Recent Advances in Online Matching: Edge-Weighted

Thorben Tröbst
Theory Seminar, October 15, 2020

Department of Computer Science, University of California, Irvine
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
Online Bipartite Matching
$G = (S, B, E)$ is a bipartite graph consisting of offline vertices $S$ and online vertices $B$. 
$G = (S, B, E)$ is a bipartite graph consisting of offline vertices $S$ and online vertices $B$.

Online vertices arrive one by one in adverserial order.
Online Bipartite Matching II

\[ G = (S, B, E) \] is a bipartite graph consisting of \textit{offline} vertices \( S \) and \textit{online} vertices \( B \).

Online vertices arrive one by one in \textit{adverserial order}. The algorithm must \textit{irrevocably} and \textit{immediately} match revealed online vertices.
$G = (S, B, E)$ is a bipartite graph consisting of offline vertices $S$ and online vertices $B$.

Online vertices arrive one by one in adverserial order.

The algorithm must irrevocably and immediately match revealed online vertices.

The goal is to maximize the competitive ratio, i.e.

$$\frac{|M_{\text{online}}|}{\text{OPT}_{\text{offline}}}.$$
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.

- **WATER-FILLING / BALANCING**
  - Idea: continuously allocate online vertices to the least-matched offline vertices.
  - Provides fractional solution in a deterministic algorithm.
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.

- **WATER-FILLING / BALANCING**
  - Idea: continuously allocate online vertices to the least-matched offline vertices.
  - Provides fractional solution in a deterministic algorithm.
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.

- **WATER-FILLING / BALANCING**
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.

- **WATER-FILLING / BALANCING**
  - Idea: continuously allocate online vertices to the least-matched offline vertices.
There are two main algorithmic ideas for online matching problems:

- **RANKING**
  - Idea: randomly permute offline vertices and then match online vertices to the first available offline vertex.
  - Provides integral solution in a randomized algorithm.

- **WATER-FILLING / BALANCING**
  - Idea: continuously allocate online vertices to the least-matched offline vertices.
  - Provides fractional solution in a deterministic algorithm.
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$. 
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a \textit{value} $v_{ji}$.

The goal is to \textit{maximize the value} of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In the Edge-Weighted Online Bipartite Matching Problem, every edge comes with a value $v_{ji}$.

The goal is to maximize the value of matched edges.

Unfortunately, no algorithm has constant competitive ratio:
In order to obtain a meaningful setting, we need an extra condition:

**Definition**

An online matching problem allows *free disposal* if the offline vertices are allowed *drop* previously matched online vertices.
In order to obtain a meaningful setting, we need an extra condition:

**Definition**

An online matching problem allows **free disposal** if the offline vertices are allowed **drop** previously matched online vertices.

⇒ **GREEDY** algorithm is now $\frac{1}{2}$-competitive!
In order to obtain a meaningful setting, we need an extra condition:

**Definition**

An online matching problem allows *free disposal* if the offline vertices are allowed *drop* previously matched online vertices.

⇒ **GREEDY** algorithm is now $\frac{1}{2}$-competitive!

⇒ This was best known until a recent breakthrough by Zadimoghaddam!
A New Algorithm for Unweighted
People tried for a long time to extend RANKING to no avail.

Consider the $1 \over 2$-BALANCE algorithm:

- Whenever an online vertex $i$ arrives, let $j_1$, $j_2$ be the two neighbors which are currently least matched.
- Fractionally match $i$ to $j_1$ and $j_2$ with a value of $1 \over 2$ each.

Yields $5 \over 9$-competitive fractional matching!
People tried for a long time to extend RANKING to no avail.
⇒ Let's find a new algorithm for the unweighted case and then try to extend that!
People tried for a long time to extend RANKING to no avail.

⇒ Let's find a new algorithm for the unweighted case and then try to extend that!

Consider the $\frac{1}{2}$-BALANCE algorithm:
People tried for a long time to extend RANKING to no avail.

⇒ Let's find a new algorithm for the unweighted case and then try to extend that!

Consider the $\frac{1}{2}$-BALANCE algorithm:

- Whenever an online vertex $i$ arrives, let $j_1, j_2$ be the two neighbors which are currently least matched.
People tried for a long time to extend RANKING to no avail.
⇒ Let's find a new algorithm for the unweighted case and then try to extend that!

Consider the $\frac{1}{2}$-BALANCE algorithm:

- Whenever an online vertex $i$ arrives, let $j_1, j_2$ be the two neighbors which are currently least matched.
- Fractionally match $i$ to $j_1$ and $j_2$ with a value of $\frac{1}{2}$ each.
People tried for a long time to extend RANKING to no avail.

⇒ Let’s find a new algorithm for the unweighted case and then try to extend that!

Consider the $\frac{1}{2}$-BALANCE algorithm:

- Whenever an online vertex $i$ arrives, let $j_1, j_2$ be the two neighbors which are currently least matched.
- Fractionally match $i$ to $j_1$ and $j_2$ with a value of $\frac{1}{2}$ each.

⇒ Yields $\frac{5}{9}$-competitive fractional matching!
- BALANCE Example
$\frac{1}{2}$-BALANCE Example
$\frac{1}{2}$-BALANCE Example
$\frac{1}{2}$-BALANCE Example
$\frac{1}{2}$-BALANCE Example
-BALANCE Example
$\frac{1}{2}$-BALANCE Example
\( \frac{1}{2} \)-BALANCE Example
$\frac{1}{2}$-BALANCE Example
$\frac{1}{2}$-BALANCE Example
1/2 - BALANCE Example
$\frac{1}{2}$-BALANCE Example
We get a half-integral matching that is guaranteed to be $\frac{5}{9}$-competitive.
We get a half-integral matching that is guaranteed to be $\frac{5}{9}$-competitive.

⇒ Can we round it somehow without too much loss?
Rounding

We get a half-integral matching that is guaranteed to be $\frac{5}{9}$-competitive.

$\Rightarrow$ Can we round it somehow without too much loss?

**Note:** Rounding after the fact is very easy with zero loss!
Rounding

We get a half-integral matching that is guaranteed to be \( \frac{5}{9} \)-competitive.

⇒ Can we round it somehow without too much loss?

**Note:** Rounding after the fact is very easy with zero loss!

**Core issue:** How to round \( \frac{1}{2} \)-BALANCE online?
Rounding Problem

•

•

•

•

•
Rounding Problem
Rounding Problem
Rounding Problem
Rounding Problem
Rounding Problem
Rounding Problem
Rounding Problem

⇒ No rounding strategy can do better than $7/8$-approx!
Obvious rounding strategy: pick uniformly at random among the two choices.
Obvious rounding strategy: pick uniformly at random among the two choices.

**Problem:** Does not beat $\frac{1}{2}$ due to collisions!
Obvious rounding strategy: pick uniformly at random among the two choices.

**Problem:** Does not beat $\frac{1}{2}$ due to collisions!

⇒ Pick uniformly if both neighbors are unpicked, otherwise try to avoid collisions.
Rounding Problem II

Just avoiding collisions is still not quite optimal:

- 
- 
- 

- 
- 
- 

- 
- 
- 

- 
- 
- 

-
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:
Just avoiding collisions is still not quite optimal:

⇒ If a vertex was not chosen previously, choose it!
Recall that we started with a $\frac{5}{9}$-competitive fractional matching...
Recall that we started with a $\frac{5}{9}$-competitive fractional matching...then we rounded and (in the worst case) get $\frac{7}{8}$ of the fractional matching.
Recall that we started with a $\frac{5}{9}$-competitive fractional matching...then we rounded and (in the worst case) get $\frac{7}{8}$ of the fractional matching.

But $\frac{5}{9} \times \frac{7}{8} = \frac{35}{72} < \frac{1}{2}$ which is not good enough!
Recall that we started with a $\frac{5}{9}$-competitive fractional matching...then we rounded and (in the worst case) get $\frac{7}{8}$ of the fractional matching.

But $\frac{5}{9} \times \frac{7}{8} = \frac{35}{72} < \frac{1}{2}$ which is not good enough!

$\Rightarrow$ A fine-grained analysis can show that in order to get close to really only get a $\frac{5}{9}$-competitive fractional matching, we need many $C_4$ or $C_6$ in the support. But small cycles are easy to round!
Recall that we started with a $\frac{5}{9}$-competitive fractional matching...then we rounded and (in the worst case) get $\frac{7}{8}$ of the fractional matching.

But $\frac{5}{9} \times \frac{7}{8} = \frac{35}{72} < \frac{1}{2}$ which is not good enough!

$\Rightarrow$ A fine-grained analysis can show that in order to get close to really only get a $\frac{5}{9}$-competitive fractional matching, we need many $C_4$ or $C_6$ in the support. But small cycles are easy to round!

So we can beat $\frac{1}{2}$ with rounded BALANCE!
The Crux of Weighted: Online Correlated Selection
Extending the Unweighted Algorithm

To extend the unweighted idea to weighted, two ingredients are needed:

• Weighted version of $1 - \frac{1}{2}$-BALANCE
• Weighted rounding

Good news: weighted $1 - \frac{1}{2}$-BALANCE can still be done with factor $\frac{5}{9}$.
See my talk last year.
To extend the unweighted idea to weighted, two ingredients are needed:

- Weighted version of $\frac{1}{2}$-BALANCE
Extending the Unweighted Algorithm

To extend the unweighted idea to weighted, two ingredients are needed:

- Weighted version of $\frac{1}{2}$-BALANCE
- Weighted rounding
Extending the Unweighted Algorithm

To extend the unweighted idea to weighted, two ingredients are needed:

- Weighted version of $\frac{1}{2}$-BALANCE
- Weighted rounding

Good news: weighted $\frac{1}{2}$-BALANCE can still be done with factor $\frac{5}{9}$. See my talk last year.
But how do we adapt the rounding strategy?
But how do we adapt the rounding strategy?
But how do we adapt the rounding strategy?
Weighted Rounding Problem

But how do we adapt the rounding strategy?

Diagram:

- Two points connected by a line labeled with 1.
- An additional point connected to the line by a line labeled with 100.
But how do we adapt the rounding strategy?
Weighted Rounding Problem

But how do we adapt the rounding strategy?

![Diagram with two points and a line segment](image)

1 + \( \epsilon \)
We can avoid weights entirely if we show a uniform bound:
We can avoid weights entirely if we show a uniform bound:

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Give an online, randomized rounding strategy such that each edge is picked with probability $\frac{1}{2}$ and there is some amount of negative correlation among vertices.</em></td>
</tr>
</tbody>
</table>
We can avoid weights entirely if we show a uniform bound:

**Problem**

*Give an online, randomized rounding strategy such that each edge is picked with probability $\frac{1}{2}$ and there is some amount of negative correlation among vertices.*

This is the problem of Online Correlated Selection, the key ingredient of the weighted matching breakthrough!
We are given a set $A$ (known in advance) and pairs $\{a, b\} \subseteq A$ arrive online adverserially.
We are given a set $A$ (known in advance) and pairs $\{a, b\} \subseteq A$ arrive online adverserially.

**Task:** For each pair $P = \{a, b\}$ pick a winner $w \in P$ online and irrecovably such that

- $P[w] = P[w'] = 1/2$.
- For any fixed $c \in A$, the probability that $c$ has not been matched after appearing in $k$ pairs is $2^{-k} (1 - \gamma)^{k-1}$ for some $\gamma > 0$. 


We are given a set $A$ (known in advance) and pairs $\{a, b\} \subseteq A$ arrive online adverserially.

**Task:** For each pair $P = \{a, b\}$ pick a winner $w \in P$ online and irrecovably such that

- $\mathbb{P}[w = a] = \mathbb{P}[w = b] = \frac{1}{2}$. 
The OCS Problem II

We are given a set $A$ (known in advance) and pairs $\{a, b\} \subseteq A$ arrive online adverserially.

Task: For each pair $P = \{a, b\}$ pick a winner $w \in P$ online and irrecovably such that

- $\mathbb{P}[w = a] = \mathbb{P}[w = b] = \frac{1}{2}$.
- For any fixed $c \in A$, the probability that $c$ has not been matched after appearing in $k$ pairs is $2^{-k}(1 - \gamma)^{k-1}$ for some $\gamma > 0$. 
The algorithm for weighted online matching is now clear:

- As vertices arrive, use weighted $1 - \text{BALANCE}$ to determine (at most) two offline vertices $j_1, j_2$ to connect to.
- Use an OCS to decide the winner of $\{j_1, j_2\}$.
- If the negative correlation is large enough (e.g. $\gamma = \frac{1}{16}$), we beat $1 - 2$ just like the unweighted case!
The algorithm for weighted online matching is now clear:

- As vertices arrive, use weighted $\frac{1}{2}$-BALANCE to determine (at most) two offline vertices $j_1, j_2$ to connect to.
The algorithm for weighted online matching is now clear:

- As vertices arrive, use weighted \( \frac{1}{2} \)-BALANCE to determine (at most) two offline vertices \( j_1, j_2 \) to connect to.
- Use an OCS to decide the winner of \( \{j_1, j_2\} \).
The algorithm for weighted online matching is now clear:

- As vertices arrive, use weighted $\frac{1}{2}$-BALANCE to determine (at most) two offline vertices $j_1, j_2$ to connect to.
- Use an OCS to decide the winner of $\{j_1, j_2\}$.
- If the negative correlation is large enough (e.g. $\gamma = \frac{1}{16}$), we beat $\frac{1}{2}$ just like the unweighted case!
Let us construct a remarkably simple OCS for $\gamma = \frac{1}{16}$:

$$\{b, e\} \; \{c, a\} \; \{c, d\} \; \{a, b\} \; \{e, c\} \; \{a, c\} \; \{a, f\} \; \{b, e\}$$
Let us construct a remarkably simple OCS for $\gamma = \frac{1}{16}$:

\[
\{b, e\} \xrightarrow{\text{c, a}} \{c, d\} \xrightarrow{\text{a, b}} \{e, c\} \xrightarrow{\text{a, c}} \{a, f\} \xrightarrow{\text{b, e}}
\]
Let us construct a remarkably simple OCS for $\gamma = \frac{1}{16}$:
Let us construct a remarkably simple OCS for $\gamma = \frac{1}{16}$:
With this construction one can see:

• Winners are always picked with probability $\frac{1}{2}$.

• Probability that two pairs which contain consecutive occurrences of some $c \in A$ are matched is (at least) $\frac{1}{16}$.

• Whenever such a pair is matched, perfect negative correlation happens for $c$.

• Implies $2^{-k}(1 - \frac{1}{16})^k - 1$ chance of not getting matched after $k$ occurrences!
Constructing an OCS

With this construction one can see:

- Winners are always picked with probability $\frac{1}{2}$.

- Probability that two pairs which contain consecutive occurrences of some $c \in A$ are matched is at least $\frac{1}{16}$.

- Whenever such a pair is matched, perfect negative correlation happens for $c$.

- Implies $2^{-k}(1 - \frac{1}{16})^{k-1}$ chance of not getting matched after $k$ occurrences!
With this construction one can see:

- Winners are always picked with probability $\frac{1}{2}$.
- Probability that two pairs which contain consecutive occurrences of some $c \in A$ are matched is (at least) $\frac{1}{16}$.
Constructing an OCS

With this construction one can see:

- Winners are always picked with probability $\frac{1}{2}$.
- Probability that two pairs which contain consecutive occurrences of some $c \in A$ are matched is (at least) $\frac{1}{16}$.
- Whenever such a pair is matched, perfect negative correlation happens for $c$. 
With this construction one can see:

- Winners are always picked with probability $\frac{1}{2}$.
- Probability that two pairs which contain consecutive occurrences of some $c \in A$ are matched is (at least) $\frac{1}{16}$.
- Whenever such a pair is matched, perfect negative correlation happens for $c$.
- Implies $2^{-k}(1 - \frac{1}{16})^{k-1}$ chance of not getting matched after $k$ occurrences!
Conclusion
Impact and Recent Works

• The OCS has already been improved by Charikar and Blanc by looking at $\frac{1}{k}$-BALANCE or even the continuous variant called WATER-FILLING.
• The OCS has already been improved by Charikar and Blanc by looking at $\frac{1}{k}$-BALANCE or even the continuous variant called WATER-FILLING.

• This leads to a 0.5368-competitive algorithm, much better than the 0.505 achieved via the $\frac{1}{2}$-BALANCE approach!
• The OCS has already been improved by Charikar and Blanc by looking at $\frac{1}{k}$-BALANCE or even the continuous variant called WATER-FILLING.

• This leads to a 0.5368-competitive algorithm, much better than the 0.505 achieved via the $\frac{1}{2}$-BALANCE approach!

• OCS has also been used to give an algorithm for the general AdWords problem that beats $\frac{1}{2}$. 
Impact and Recent Works

- The OCS has already been improved by Charikar and Blanc by looking at $\frac{1}{k}$-BALANCE or even the continuous variant called WATER-FILLING.
- This leads to a 0.5368-competitive algorithm, much better than the 0.505 achieved via the $\frac{1}{2}$-BALANCE approach!
- OCS has also been used to give an algorithm for the general AdWords problem that beats $\frac{1}{2}$.
- Several other online matching problems have edge weighted variants that could be tackled by this new tool.
Thank You!