Online Matching with High Probability

Thorben Tröbst (joint work with Milena Mihail)

Theory Seminar, January 13, 2022

Department of Computer Science, University of California, Irvine

Randomized Algorithms and Concentration

The Power of Randomized Algorithms

Many problemss in computer science can be solved in a more natural, efficient, or better way using randomization. You all know many examples such as:

- Quicksort
- Miller-Rabin primality test
- Hashing
- Polynomial identity testing
- Perfect matching on parallel machines
- etc...

Expectation versus Concentration

However, usually the focus is on expectation or showing non-zero probability of success. For example, let C be the total number of comparisons of Quicksort with random pivots.

Expectation versus Concentration

However, usually the focus is on expectation or showing non-zero probability of success. For example, let C be the total number of comparisons of Quicksort with random pivots.

- Most people have seen: $\mathbb{E}[C] = O(n \log n)$.

Expectation versus Concentration

However, usually the focus is on expectation or showing non-zero probability of success. For example, let C be the total number of comparisons of Quicksort with random pivots.

- Most people have seen: $\mathbb{E}[C] = O(n \log n)$.
- Fewer know: $\mathbb{P}[C > c_0 \cdot n \log n] < \frac{1}{n}$ for some c_0.

Expectation versus Concentration

However, usually the focus is on expectation or showing non-zero probability of success. For example, let C be the total number of comparisons of Quicksort with random pivots.

- Most people have seen: $\mathbb{E}[C] = O(n \log n)$.
- Fewer know: $\mathbb{P}[C > c_0 \cdot n \log n] < \frac{1}{n}$ for some c_0.
- But did you know:

$$\mathbb{P}[|C/\mathbb{E}[C] - 1| > \epsilon] < n^{-2\epsilon(\ln \ln n - \ln(1/\epsilon) + O(\ln \ln \ln n))}$$

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.
- They tell us that bad behavior is extremely unlikely!

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.
- They tell us that bad behavior is extremely unlikely!

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.
- They tell us that bad behavior is extremely unlikely!

However, outside of a few areas (e.g. graph coloring), concentration results are relatively rare because of boosting:

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.
- They tell us that bad behavior is extremely unlikely!

However, outside of a few areas (e.g. graph coloring), concentration results are relatively rare because of boosting:

- Want good performance? Simply run the algorithm $O(\log n)$ many times.

Usefulness of Concentration

- Tight concentration bounds like those for Quicksort provide useful information about behavior in practice.
- They tell us that bad behavior is extremely unlikely!

However, outside of a few areas (e.g. graph coloring), concentration results are relatively rare because of boosting:

- Want good performance? Simply run the algorithm $O(\log n)$ many times.
- Want good runtime? Simply run the algorithm $O(\log n)$ many times in parallel.

Concentration for Online Algorithms

Online Algorithms cannot be repeated and thus cannot be boosted! Still, fairly few examples of online algorithms analyzed wrt. concentration, e.g.

Concentration for Online Algorithms

Online Algorithms cannot be repeated and thus cannot be boosted! Still, fairly few examples of online algorithms analyzed wrt. concentration, e.g.

- Online Randomized Call Controll Revisited (Leonardi, Marchetti-Spaccamela, Presciutti, Rosen 2001)

Concentration for Online Algorithms

Online Algorithms cannot be repeated and thus cannot be boosted! Still, fairly few examples of online algorithms analyzed wrt. concentration, e.g.

- Online Randomized Call Controll Revisited (Leonardi, Marchetti-Spaccamela, Presciutti, Rosen 2001)
- Randomized Online Algorithms with High Probability Guarantees (Komm, Kralovic, Kralovic, Mömke 2014)

Concentration for Online Algorithms

Online Algorithms cannot be repeated and thus cannot be boosted! Still, fairly few examples of online algorithms analyzed wrt. concentration, e.g.

- Online Randomized Call Controll Revisited (Leonardi, Marchetti-Spaccamela, Presciutti, Rosen 2001)
- Randomized Online Algorithms with High Probability Guarantees (Komm, Kralovic, Kralovic, Mömke 2014)
- Online Edge Coloring Algorithms via the Nibble Method (Bhattacharya, Grandoni, Wajc 2020)

Concentration for Online Algorithms

Online Algorithms cannot be repeated and thus cannot be boosted! Still, fairly few examples of online algorithms analyzed wrt. concentration, e.g.

- Online Randomized Call Controll Revisited (Leonardi, Marchetti-Spaccamela, Presciutti, Rosen 2001)
- Randomized Online Algorithms with High Probability Guarantees (Komm, Kralovic, Kralovic, Mömke 2014)
- Online Edge Coloring Algorithms via the Nibble Method (Bhattacharya, Grandoni, Wajc 2020)

Should be more results like this!

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)
- Isoperimetric inequalities (Talagrand)

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)
- Isoperimetric inequalities (Talagrand)
- Transportation cost inequalities

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)
- Isoperimetric inequalities (Talagrand)
- Transportation cost inequalities
- log-Sobolev inequalities

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)
- Isoperimetric inequalities (Talagrand)
- Transportation cost inequalities
- log-Sobolev inequalities
- ...

Tools for Proving Concentration

The tools used for proving concentration of randomized algorithms, are nice results from probability theory:

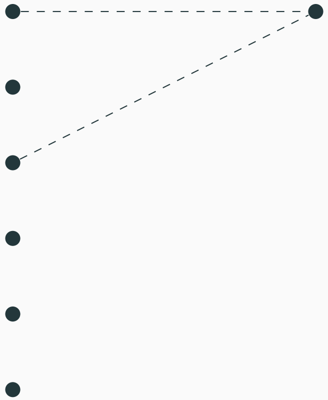
- Chernoff-Hoeffding bounds
- Martingale inequalities (Azuma, McDiarmid)
- Isoperimetric inequalities (Talagrand)
- Transportation cost inequalities
- log-Sobolev inequalities
- ...

See "Concentration of Measure for the Analysis of Randomized Algorithms" by Dubhashi and Panconesi

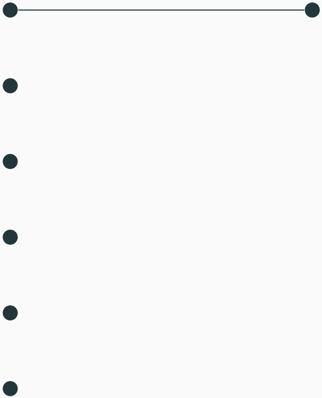
Online Bipartite Matching

Online Bipartite Matching

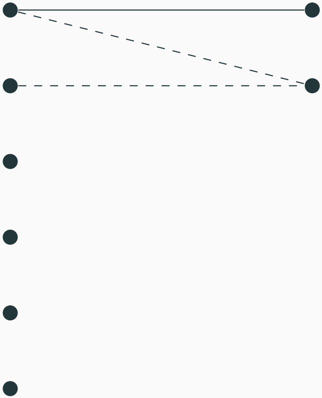
Online Bipartite Matching



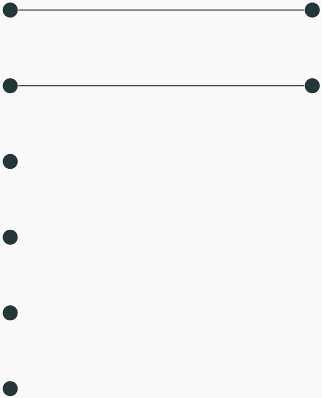
Online Bipartite Matching



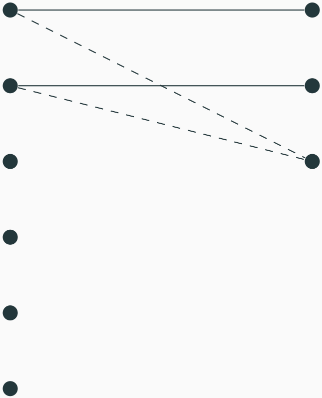
Online Bipartite Matching



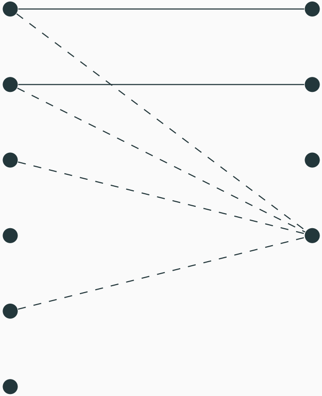
Online Bipartite Matching



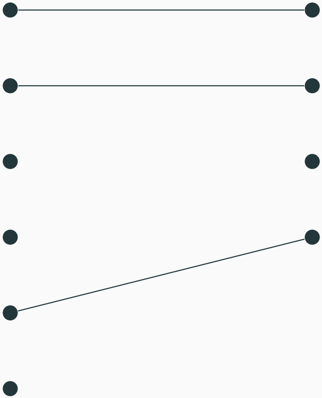
Online Bipartite Matching



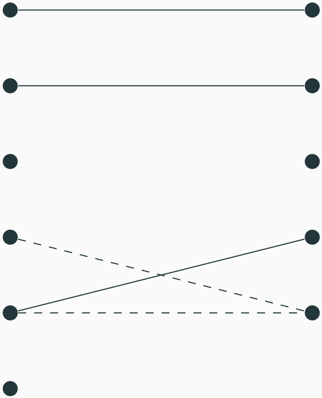
Online Bipartite Matching



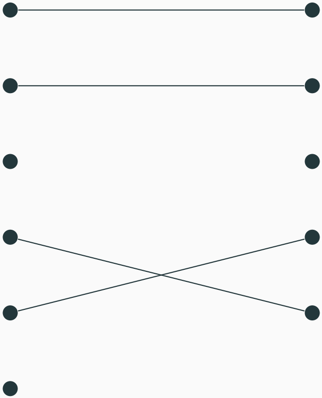
Online Bipartite Matching



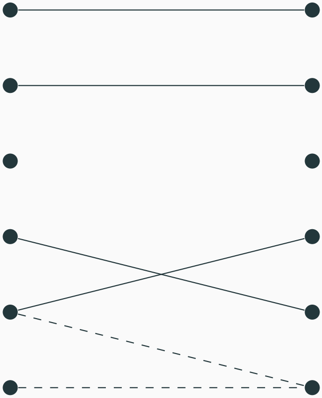
Online Bipartite Matching



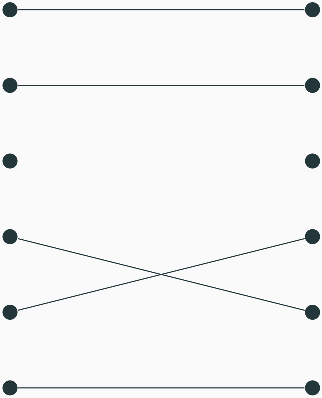
Online Bipartite Matching



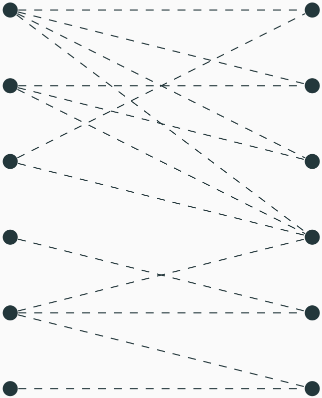
Online Bipartite Matching



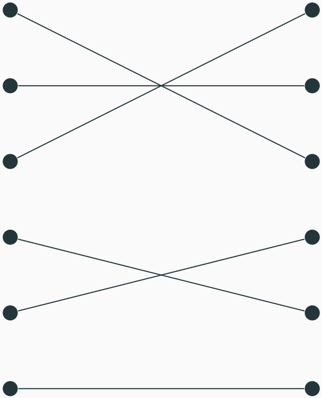
Online Bipartite Matching



Online Bipartite Matching



Online Bipartite Matching



Online Bipartite Matching II

$G = (S, B, E)$ is a bipartite graph consisting of offline vertices S and online vertices B.

Online Bipartite Matching II

$G = (S, B, E)$ 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 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.

Online Bipartite Matching II

$G = (S, B, E)$ 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.

$$\frac{|M_{\text{online}}|}{\text{OPT}_{\text{offline}}}.$$

Classic results for Online Bipartite Matching:

Classic results for Online Bipartite Matching:

- The GREEDY algorithm (match whenever possible) is 1/2-competitive.

Classic results for Online Bipartite Matching:

- The GREEDY algorithm (match whenever possible) is 1/2-competitive.
- 1/2-competitive is best possible for deterministic algorithms.

Algorithms for Online Matching Problems

Classic results for Online Bipartite Matching:

- The GREEDY algorithm (match whenever possible) is 1/2-competitive.
- 1/2-competitive is best possible for deterministic algorithms.
- The randomized RANKING algorithm is $(1 - 1/e)$-competitive in expectation.

Algorithms for Online Matching Problems

Classic results for Online Bipartite Matching:

- The GREEDY algorithm (match whenever possible) is 1/2-competitive.
- 1/2-competitive is best possible for deterministic algorithms.
- The randomized RANKING algorithm is $(1 - 1/e)$-competitive in expectation.
- $(1 - 1/e)$-competitive in expectation is best possible for randomized algorithms.

Ranking with High Probability

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

- Pick a uniformly random permutation π on the offline vertices.

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

- Pick a uniformly random permutation π on the offline vertices.
- Whenever online vertex i arrives, match it to an unmatched $j \in N(i)$ that comes first wrt. π.

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

- Pick a uniformly random permutation π on the offline vertices.
- Whenever online vertex i arrives, match it to an unmatched $j \in N(i)$ that comes first wrt. π.

is equivalent to

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

- Pick a uniformly random permutation π on the offline vertices.
- Whenever online vertex i arrives, match it to an unmatched $j \in N(i)$ that comes first wrt. π.

is equivalent to

- Pick a uniformly random real $x_j \in [0, 1]$ for each offline vertex j.

The Ranking Algorithm

There are two equivalent descriptions of Ranking:

- Pick a uniformly random permutation π on the offline vertices.
- Whenever online vertex i arrives, match it to an unmatched $j \in N(i)$ that comes first wrt. π.

is equivalent to

- Pick a uniformly random real $x_j \in [0, 1]$ for each offline vertex j.
- Whenever online vertex i arrives, match it to an umatched $j \in N(i)$ minimizing x_j.

RANKING Example

-
-
-
-
-
-

Ranking Example

0.6

0.5

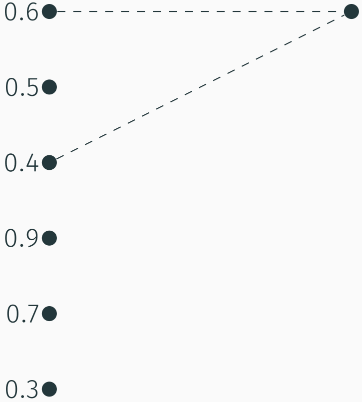
0.4

0.9

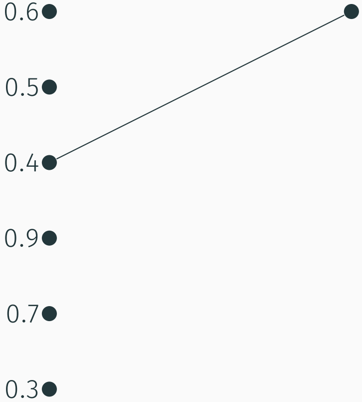
0.7

0.3

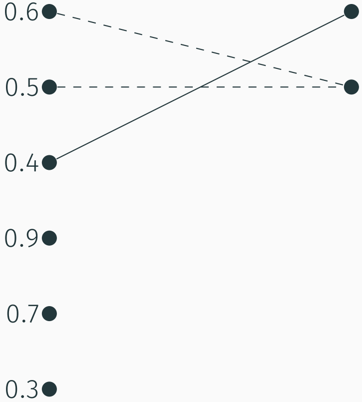
Ranking Example



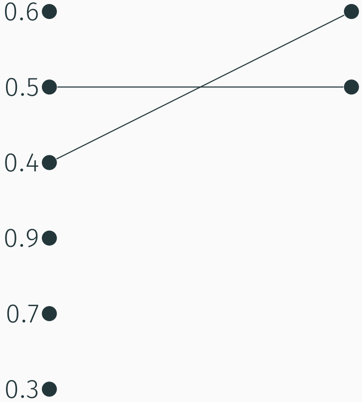
Ranking Example



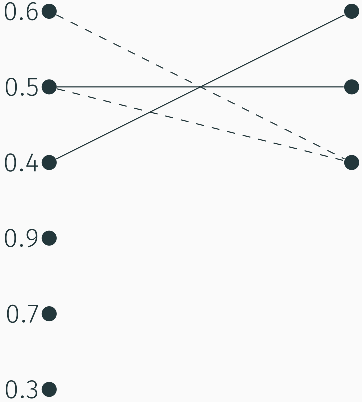
Ranking Example



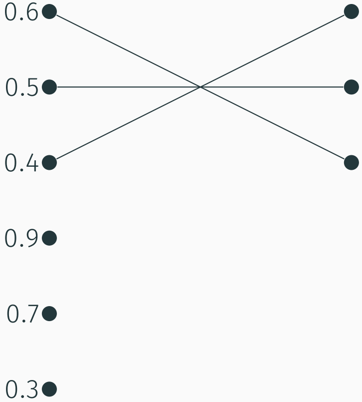
Ranking Example

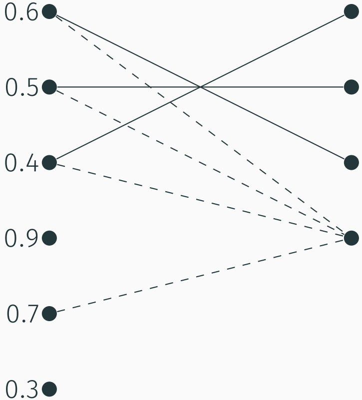


Ranking Example

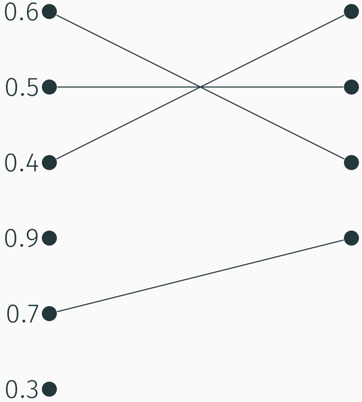


RANKING Example

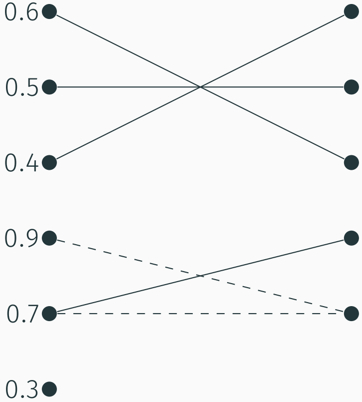



0.6
0.5
0.4
0.9
0.7
0.3

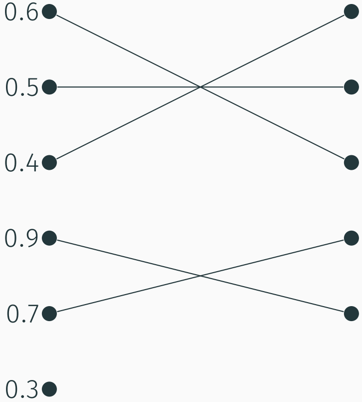
Ranking Example



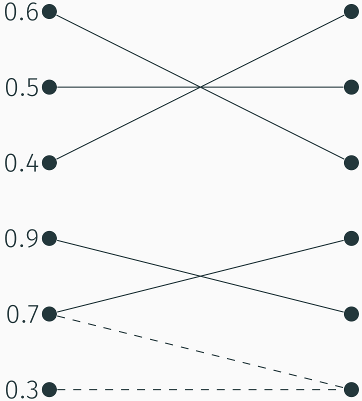
Ranking Example



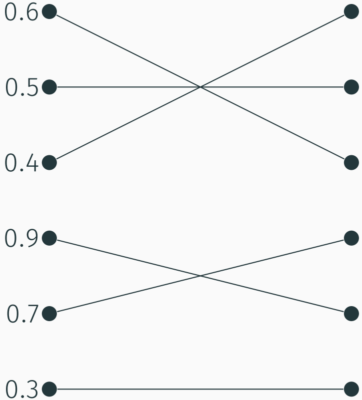
Ranking Example



Ranking Example



Ranking Example



Main Theorem

Theorem

Consider an instance (S, B, E) of the Biparite Online Matching Problem which admits a matching of size n. Then for any $\alpha > 0$ and any arrival order,

$$\mathbb{P}\left[|M| < \left(1 - \frac{1}{e} - \alpha\right) n\right] < e^{-2\alpha^2 n}$$

where M is the random variable denoting the matching generated by Ranking.

Main Theorem

Theorem

Consider an instance (S, B, E) of the Biparite Online Matching Problem which admits a matching of size n. Then for any $\alpha > 0$ and any arrival order,

$$\mathbb{P}\left[|M| < \left(1 - \frac{1}{e} - \alpha\right) n\right] < e^{-2\alpha^2 n}$$

where M is the random variable denoting the matching generated by RANKING.

For now assume that n is also the number of offline / online vertex (i.e. there is a perfect matching).

McDiarmid's Inequality

We will use the this nice consequence of Azuma's inequality:

McDiarmid's Inequality

We will use the this nice consequence of Azuma's inequality:

Lemma (McDiarmid 1989)

Let $c_1, \ldots, c_n \in \mathbb{R}_+$ and consider some function $f : [0,1]^n \to \mathbb{R}$ satisfying

$$|f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_n)| \leq c_i$$

for all $x \in [0,1]^n$, $i \in [n]$ and $x_i' \in [0,1]$. Moreover let Δ^n be the uniform distribution on $[0,1]^n$. Then for all $t > 0$, we have

$$\mathbb{P}_{x \sim \Delta^n}[f(x) < \mathbb{E}_{y \sim \Delta^n}[f(y)] - t] < e^{-\frac{2t^2}{\sum_{i=1}^n c_i^2}}.$$

Proof Strategy

The general strategy is as follows:

Proof Strategy

The general strategy is as follows:

1. Let $f(x_1, \ldots, x_n)$ be the size of matching output by Ranking with samples $x_1, \ldots, x_n$.

Proof Strategy

The general strategy is as follows:

1. Let $f(x_1, \ldots, x_n)$ be the size of matching output by Ranking with samples $x_1, \ldots, x_n$.
2. Prove the necessary bounded differences property of f.

Lemma (Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq 1$.

Proof Strategy

The general strategy is as follows:

1. Let $f(x_1, \ldots, x_n)$ be the size of matching output by RANKING with samples $x_1, \ldots, x_n$.
2. Prove the necessary bounded differences property of f.

Lemma (Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq 1$.

3. Apply McDiarmid's inequality with $t = \alpha n$ and $c_i = 1$.

Bounded Differences

Bounded differences follows directly from the following:

Lemma

Let $j \in S$, then we can define the graph G_{-j} which contains all vertices of G except for j. For some fixed values of $x \in [0,1]^S$, we let M be the matching produced by Ranking *in G and let M_{-j} be the matching produced by* Ranking *in G_{-j}. Then $|M_{-j}| \leq |M| \leq |M_{-j}| + 1$.*

Bounded Differences

Bounded differences follows directly from the following:

Lemma

Let $j \in S$, then we can define the graph G_{-j} which contains all vertices of G except for j. For some fixed values of $x \in [0,1]^S$, we let M be the matching produced by Ranking *in G and let M_{-j} be the matching produced by* Ranking *in G_{-j}. Then $|M_{-j}| \leq |M| \leq |M_{-j}| + 1$.*

Proof. Live. $\square$

Conclusion

In conclusion, for the Online Bipartite Matching Problem:

Conclusion

In conclusion, for the Online Bipartite Matching Problem:

- We get a non-trivial exponential concentration result (i.e. no boosting).

Conclusion

In conclusion, for the Online Bipartite Matching Problem:

- We get a non-trivial exponential concentration result (i.e. no boosting).
- The proof is elegant and uses a nice result from probability theory with an equally nice structural lemma about matchings.

Conclusion

In conclusion, for the Online Bipartite Matching Problem:

- We get a non-trivial exponential concentration result (i.e. no boosting).
- The proof is elegant and uses a nice result from probability theory with an equally nice structural lemma about matchings.
- This should be just as well-known as $\mathbb{E}[|M|] \geq (1 - 1/e)n$!

Can this be extended to other Online Matching Problems? Yes!

Generalizations

Fully Online Matching

In Fully Online Matching:

Fully Online Matching

In Fully Online Matching:

- We have an (in general) non-bipartite graph $G = (V, E)$.

Fully Online Matching

In Fully Online Matching:

- We have an (in general) non-bipartite graph $G = (V, E)$.
- Vertices arrive and depart in adversarial order.

Fully Online Matching

In Fully Online Matching:

- We have an (in general) non-bipartite graph $G = (V, E)$.
- Vertices arrive and depart in adverserial order.
- Vertices must be matched after they arrive and before they depart.

$\Rightarrow$ Models e.g. ride-sharing problems and is a direct generalization of Online Bipartite Matching!

The Ranking algorithm can still be used:

Ranking for Fully Online Matching

The Ranking algorithm can still be used:

- Whenever vertex i arrives, assign a uniformly random real $x_i \in [0, 1]$.

RANKING for Fully Online Matching

The RANKING algorithm can still be used:

- Whenever vertex i arrives, assign a uniformly random real $x_i \in [0, 1]$.
- Whenever vertex i departs, if it has not been matched, match it to an unmatched neighbor minimizing x_i.

Ranking for Fully Online Matching

The Ranking algorithm can still be used:

- Whenever vertex i arrives, assign a uniformly random real $x_i \in [0, 1]$.
- Whenever vertex i departs, if it has not been matched, match it to an unmatched neighbor minimizing x_i.

Huang, Kang, Tang, Wu, Zhang 2018: 0.521-competitive in general, 0.567-competitive on bipartite graphs.

Concentration for Fully Online Matching

Theorem

Let G be an instance of the Fully Online Matching Problem which admits a matching of size n. Then for any $\alpha > 0$,

$$\mathbb{P}\left[|M| < (\rho - \alpha)\, n\right] < e^{-\alpha^2 n}$$

where M is the random variable denoting the matching generated by Ranking *and ρ is the competitive ratio of* Ranking.

Concentration for Fully Online Matching

Theorem

Let G be an instance of the Fully Online Matching Problem which admits a matching of size n. Then for any $\alpha > 0$,

$$\mathbb{P}\left[|M| < (\rho - \alpha)\, n\right] < e^{-\alpha^2 n}$$

where M is the random variable denoting the matching generated by Ranking *and ρ is the competitive ratio of* Ranking.

Proof. Almost the same as for Online Bipartite Matching! ☐

Vertex-Weighted Matching

In Online Vertex-Weighted Bipartite Matching:

Vertex-Weighted Matching

In Online Vertex-Weighted Bipartite Matching:

- We have a bipartite graph $G = (S, B, E)$ but also weights $w : S \to \mathbb{R}_{\geq 0}$.

Vertex-Weighted Matching

In Online Vertex-Weighted Bipartite Matching:

- We have a bipartite graph $G = (S, B, E)$ but also weights $w : S \to \mathbb{R}_{\geq 0}$.
- Vertices from B arrive online in adverserial order and must be matched immediately.

In Online Vertex-Weighted Bipartite Matching:

- We have a bipartite graph $G = (S, B, E)$ but also weights $w : S \to \mathbb{R}_{\geq 0}$.
- Vertices from B arrive online in adverserial order and must be matched immediately.
- Goal is to maximize weight of matched vertices.

Vertex-Weighted Matching

In Online Vertex-Weighted Bipartite Matching:

- We have a bipartite graph $G = (S, B, E)$ but also weights $w : S \to \mathbb{R}_{\geq 0}$.
- Vertices from B arrive online in adverserial order and must be matched immediately.
- Goal is to maximize weight of matched vertices.

Note. Edge-weighted also exists but is much harder!

With vertex-weights, Ranking needs to bias the distribution on permutations:

With vertex-weights, Ranking needs to bias the distribution on permutations:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.

With vertex-weights, Ranking needs to bias the distribution on permutations:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1})$.

Ranking for Vertex-Weighted Matching

With vertex-weights, Ranking needs to bias the distribution on permutations:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1})$.
- When $i \in B$ arrives, match to unmatched neighbor j maximizing u_j.

With vertex-weights, Ranking needs to bias the distribution on permutations:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1})$.
- When $i \in B$ arrives, match to unmatched neighbor j maximizing u_j.

Well known that this still gives $(1 - \frac{1}{e})$-competitive!

Concentration for Vertex-Weighted Matching

Theorem

For any $\alpha > 0$, there exists a variant of RANKING *such that for any instance $G = (S, B, E)$ with weights $w : S \to \mathbb{R}_+$ of the Online Vertex-Weighted Bipartite Matching, any arrival order of B and any matching M^*,*

$$\mathbb{P}\left[w(M) < \left(1 - \frac{1}{e} - \alpha\right) w(M^\star)\right] < e^{-\frac{1}{50}\alpha^4 \frac{w(M^\star)^2}{||w||_2^2}}$$

where M denotes the matching generated by RANKING *and*

$$w(M) := \sum_{\{i,j\} \in M} w_j.$$

Proof Idea

Initial idea, show:

Lemma (Weighted Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq w_j$ where f is the weight of the Ranking *output.*

Proof Idea

Initial idea, show:

Lemma (Weighted Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq w_j$ where f is the weight of the RANKING output.

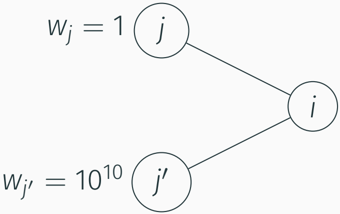
This would give

$$\mathbb{E}\left[w(M) < \left(1 - \frac{1}{e} - \alpha\right) w(M^\star)\right] < e^{-2\frac{w(M^\star)^2}{||w||_2^2}}$$

via weighted McDiarmid.

The Problem

But this does not work!



Consider $x_{j'} > 1 - 10^{-11}$. Then for some values of x_j, i picks j over j' because:

$$w_{j'}(1 - e^{x_{j'}-1}) < w_j(1 - e^{x_j-1}).$$

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

Or we can just change the algorithm to ϵ-Ranking:

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

Or we can just change the algorithm to ϵ-Ranking:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

Or we can just change the algorithm to ϵ-Ranking:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1 - \epsilon})$.

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

Or we can just change the algorithm to ϵ-Ranking:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1 - \epsilon})$.
- When $i \in B$ arrives, match to unmatched neighbor j maximizing u_j.

The Fix

We can use more complicated variants of McDiarmid that avoid bad events.

Or we can just change the algorithm to ϵ-RANKING:

- For each $j \in S$, sample a uniformly random real $x_j \in [0, 1]$.
- Assign a utility $u_j = w_j(1 - e^{x_j - 1 - \epsilon})$.
- When $i \in B$ arrives, match to unmatched neighbor j maximizing u_j.

This is still $(1 - 1/e - \epsilon)$-competitive!

The Fix II

Now we get:

Lemma (Weighted Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq \frac{2}{\epsilon} w_j$ where f is the weight of the ϵ-RANKING output.

Now we get:

Lemma (Weighted Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq \frac{2}{\epsilon} w_j$ where f is the weight of the ϵ-Ranking output.

So to get concentration above $(1 - 1/e - \alpha)w(M^\star)$:

The Fix II

Now we get:

Lemma (Weighted Bounded Differences)

Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq \frac{2}{\epsilon} w_j$ where f is the weight of the ϵ-Ranking output.

So to get concentration above $(1 - 1/e - \alpha)w(M^\star)$:

1. Run $\frac{\alpha}{2}$-Ranking to be $(1 - 1/e - \alpha/2)$-competitive.

The Fix II

Now we get:

Lemma (Weighted Bounded Differences)

*Let $x \in [0,1]^S$, $j^\star \in S$ and $\theta \in [0,1]$ be arbitrary. Define x'_j to be θ if $j = j^\star$ and x_j otherwise. Then $|f(x) - f(x')| \leq \frac{2}{\epsilon} w_j$ where f is the weight of the ϵ-*RANKING *output.*

So to get concentration above $(1 - 1/e - \alpha)w(M^\star)$:

1. Run $\frac{\alpha}{2}$-RANKING to be $(1 - 1/e - \alpha/2)$-competitive.
2. Use McDiarmid with $\alpha/2$ to get concentration above $(1 - 1/e - \alpha/2 - \alpha/2)w(M^\star)$. $\square$

Conclusion

Conclusion

Some final remarks:

Conclusion

Some final remarks:

- Concentration results for randomized algorithms are an underappreciated area!

Conclusion

Some final remarks:

- Concentration results for randomized algorithms are an underappreciated area!
- Particularly interesting for online algorithms!

Conclusion

Some final remarks:

- Concentration results for randomized algorithms are an underappreciated area!
- Particularly interesting for online algorithms!
- Open problem: is there a way to get $e^{-2\alpha^2 \frac{w(M^\star)^2}{||w||_2^2}}$ bounds for Vertex-Weighted Matching?

Conclusion

Some final remarks:

- Concentration results for randomized algorithms are an underappreciated area!
- Particularly interesting for online algorithms!
- Open problem: is there a way to get $e^{-2\alpha^2 \frac{w(M^\star)^2}{||w||_2^2}}$ bounds for Vertex-Weighted Matching?
- Open problem: is there a way to get dependence on $M^\star$ instead of $||w||_2^2$?

Conclusion

Some final remarks:

- Concentration results for randomized algorithms are an underappreciated area!
- Particularly interesting for online algorithms!
- Open problem: is there a way to get $e^{-2\alpha^2 \frac{w(M^\star)^2}{||w||_2^2}}$ bounds for Vertex-Weighted Matching?
- Open problem: is there a way to get dependence on $M^\star$ instead of $||w||_2^2$?
- Open problem: Can you show that these bounds are tight in some sense?

Thank You!