Parallel Algorithms for Perfect Matchings

Prathik, Prayas & Sreenivas

17-November-2022

Table of contents I

Table of contents II

Matching is as Easy as Matrix Inversion

Brief Overview

- A new algorithm for finding a maximum matching is presented

Brief Overview

- A new algorithm for finding a maximum matching is presented
- The core feature of this algorithm is that the only non-trivial computation involved is the inversion of an integer matrix

Brief Overview

- A new algorithm for finding a maximum matching is presented
- The core feature of this algorithm is that the only non-trivial computation involved is the inversion of an integer matrix
- Since there are existing parallel algorithms for matrix inversion, this shows that maximum matching is in RNC^2

Brief Overview

- A new algorithm for finding a maximum matching is presented
- The core feature of this algorithm is that the only non-trivial computation involved is the inversion of an integer matrix
- Since there are existing parallel algorithms for matrix inversion, this shows that maximum matching is in RNC^2
- As this algorithm is designed with parallel computation in mind, the core problem we must solve is that of coordinating all the parallel processors to seek the same solution

Brief Overview

- A new algorithm for finding a maximum matching is presented
- The core feature of this algorithm is that the only non-trivial computation involved is the inversion of an integer matrix
- Since there are existing parallel algorithms for matrix inversion, this shows that maximum matching is in RNC^2
- As this algorithm is designed with parallel computation in mind, the core problem we must solve is that of coordinating all the parallel processors to seek the same solution
- This is done by using the isolating lemma, which can (with a high probability) single out one matching in the graph

The Isolating Lemma

- In its most general form, the isolating lemma holds for any set system

The Isolating Lemma

- In its most general form, the isolating lemma holds for any set system
- This provides a relation between the parallel complexity of an arbitrary search problem and the corresponding weighted decision problem

The Isolating Lemma

- In its most general form, the isolating lemma holds for any set system
- This provides a relation between the parallel complexity of an arbitrary search problem and the corresponding weighted decision problem
- In particular, we show that Exact Matching (a problem not known to be in P) is in RNC^2

The Isolating Lemma

- In its most general form, the isolating lemma holds for any set system
- This provides a relation between the parallel complexity of an arbitrary search problem and the corresponding weighted decision problem
- In particular, we show that Exact Matching (a problem not known to be in P) is in RNC^2
- The isolating lemma can also be used to provide a simpler proof of the Valiant-Vazirani theorem, which shows that SAT remains hard (under randomized reductions) even if it is known that there is at most one satisfying assignment

History

- The maximum matching problem is a natural and well-studied problem, with the very idea of considering “tractable” to mean “poly-time solvable” first arising in the context of solving the general matching problem

History

- The maximum matching problem is a natural and well-studied problem, with the very idea of considering "tractable" to mean "poly-time solvable" first arising in the context of solving the general matching problem
- Parallel algorithms to solve the problem typically require a different, often algebraic or probabilistic, method from solving them sequentially

History

- The maximum matching problem is a natural and well-studied problem, with the very idea of considering "tractable" to mean "poly-time solvable" first arising in the context of solving the general matching problem
- Parallel algorithms to solve the problem typically require a different, often algebraic or probabilistic, method from solving them sequentially
- A critical idea used is Tutte's theorem (1947), which states that a graph has a perfect matching iff a certain "Tutte" matrix of indeterminates is non-singular

First Attempts

- Lovasz was the first to use Tutte's theorem by converting the decision problem of checking if a graph has a perfect matching to that of testing if a matrix of indeterminates is non-singular

First Attempts

- Lovasz was the first to use Tutte's theorem by converting the decision problem of checking if a graph has a perfect matching to that of testing if a matrix of indeterminates is non-singular
- This yields an RNC^2 algorithm for the decision as checking if a matrix of indeterminates is non-singular can be done in NC^2

First Attempts

- Lovasz was the first to use Tutte's theorem by converting the decision problem of checking if a graph has a perfect matching to that of testing if a matrix of indeterminates is non-singular
- This yields an RNC^2 algorithm for the decision as checking if a matrix of indeterminates is non-singular can be done in NC^2
- The search problem of actually finding a perfect matching is much harder - Karp, Upfal, and Winderson were the first to provide an RNC^3 algorithm for the same, based on using the Tutte matrix to rank and prune edges from the graph

First Attempts

- Lovasz was the first to use Tutte's theorem by converting the decision problem of checking if a graph has a perfect matching to that of testing if a matrix of indeterminates is non-singular
- This yields an RNC^2 algorithm for the decision as checking if a matrix of indeterminates is non-singular can be done in NC^2
- The search problem of actually finding a perfect matching is much harder - Karp, Upfal, and Winderson were the first to provide an RNC^3 algorithm for the same, based on using the Tutte matrix to rank and prune edges from the graph

This Paper

This algorithm directly finds a perfect matching, and is faster (RNC^2) while using $O(n^{3.5}m)$ processors

The Isolating Lemma

Definition

A *set system* (S, F) consists of a finite set S of elements, $S = \{x_1, ..., x_n\}$ and a family F of subsets of S, i.e., $F = \{S_1, ..., S_k\}$, where $S_i \subseteq S$ for $1 \leq i \leq k$

The Isolating Lemma

Definition

A *set system* (S, F) consists of a finite set S of elements, $S = \{x_1, ..., x_n\}$ and a family F of subsets of S, i.e., $F = \{S_1, ..., S_k\}$, where $S_i \subseteq S$ for $1 \leq i \leq k$

We assume every x_i appears in at least one subset
We can assign a weight w_i to each element $x_i \in S$, and define the weight of a subset S_j to be $\sum_{x_i \in S_j} w_i$

The Isolating Lemma

Definition

A *set system* (S, F) consists of a finite set S of elements, $S = \{x_1, ..., x_n\}$ and a family F of subsets of S, i.e., $F = \{S_1, ..., S_k\}$, where $S_i \subseteq S$ for $1 \leq i \leq k$

We assume every x_i appears in at least one subset
We can assign a weight w_i to each element $x_i \in S$, and define the weight of a subset S_j to be $\sum_{x_i \in S_j} w_i$

Lemma

Let (S, F) be a set system with weights assigned uniformly and independently at random from $[1, 2n]$. Then:

$$\mathbb{P}[\textit{there is a unique minimum weight set in } F] \geq \frac{1}{2}$$

Proof

- Say we fix the weights for all elements except some x_i. Then define the *threshold* for x_i to be the real number (possibly negative) α_i such that if $w_i \leq \alpha_i$, then x_i is in some minimum weight subset, while if $w_i > \alpha_i$, then x_i is not in any minimum weight subset

Proof

- Say we fix the weights for all elements except some x_i. Then define the *threshold* for x_i to be the real number (possibly negative) α_i such that if $w_i \leq \alpha_i$, then x_i is in some minimum weight subset, while if $w_i > \alpha_i$, then x_i is not in any minimum weight subset
- Note that if $w_i < \alpha_i$, then x_i must be in *every* minimum weight subset – any ambiguity about x_i only occurs if $w_i = \alpha_i$, since then there is some minimum weight subset containing x_i and some other minimum weight subset not containing x_i

Proof

- Say we fix the weights for all elements except some x_i. Then define the *threshold* for x_i to be the real number (possibly negative) α_i such that if $w_i \leq \alpha_i$, then x_i is in some minimum weight subset, while if $w_i > \alpha_i$, then x_i is not in any minimum weight subset
- Note that if $w_i < \alpha_i$, then x_i must be in *every* minimum weight subset – any ambiguity about x_i only occurs if $w_i = \alpha_i$, since then there is some minimum weight subset containing x_i and some other minimum weight subset not containing x_i
- We will say that an element x_i is *singular* if $w_i = \alpha_i$

Proof

- Say we fix the weights for all elements except some x_i. Then define the *threshold* for x_i to be the real number (possibly negative) α_i such that if $w_i \leq \alpha_i$, then x_i is in some minimum weight subset, while if $w_i > \alpha_i$, then x_i is not in any minimum weight subset
- Note that if $w_i < \alpha_i$, then x_i must be in *every* minimum weight subset – any ambiguity about x_i only occurs if $w_i = \alpha_i$, since then there is some minimum weight subset containing x_i and some other minimum weight subset not containing x_i
- We will say that an element x_i is *singular* if $w_i = \alpha_i$
- Note that there can only be multiple minimum weight subsets if there is at least one singular element

Key Observation

- Notice now that α_i was defined *without reference to w_i*, so it follows that α_i and w_i are independent (as random variables)

Key Observation

- Notice now that α_i was defined *without reference to w_i*, so it follows that α_i and w_i are independent (as random variables)
- Thus, as w_i is distributed uniformly over $[1, 2n]$, we get that

$$\mathbb{P}[x_i \text{ is singular, i.e., } w_i = \alpha_i] \leq \frac{1}{2n}$$

Key Observation

- Notice now that α_i was defined *without reference to w_i*, so it follows that α_i and w_i are independent (as random variables)
- Thus, as w_i is distributed uniformly over $[1, 2n]$, we get that

$$\mathbb{P}[x_i \text{ is singular, i.e., } w_i = \alpha_i] \leq \frac{1}{2n}$$

- Then by the union bound over the n elements, we have:

$$\mathbb{P}[\text{at least one element is singular}] \leq \frac{1}{2}$$

- This completes the proof of the lemma by the previous observation

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs
- The input will be a bipartite graph $G = (U, V, E)$, and the goal is to find a perfect matching in G

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs
- The input will be a bipartite graph $G = (U, V, E)$, and the goal is to find a perfect matching in G
- We can view the set of edges E along with the collection of perfect matchings (which are subsets of E) as a set system

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs
- The input will be a bipartite graph $G = (U, V, E)$, and the goal is to find a perfect matching in G
- We can view the set of edges E along with the collection of perfect matchings (which are subsets of E) as a set system
- We randomly assign each edge with a weight in $[1, 2m]$, where $m = |E|$, so that by the isolating lemma, there is a unique perfect matching with minimum weight

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs
- The input will be a bipartite graph $G = (U, V, E)$, and the goal is to find a perfect matching in G
- We can view the set of edges E along with the collection of perfect matchings (which are subsets of E) as a set system
- We randomly assign each edge with a weight in $[1, 2m]$, where $m = |E|$, so that by the isolating lemma, there is a unique perfect matching with minimum weight
- We design our parallel algorithm to seek this perfect matching

The Matching Algorithm - Bipartite Graphs

- First, we will consider the simpler case of bipartite graphs
- The input will be a bipartite graph $G = (U, V, E)$, and the goal is to find a perfect matching in G
- We can view the set of edges E along with the collection of perfect matchings (which are subsets of E) as a set system
- We randomly assign each edge with a weight in $[1, 2m]$, where $m = |E|$, so that by the isolating lemma, there is a unique perfect matching with minimum weight
- We design our parallel algorithm to seek this perfect matching
- Let D be the biadjacency matrix of G and generate B from D by replacing every $d_{ij} = 1$ with $b_{ij} = 2^{w_{ij}}$, where w_{ij} is the weight for the edge (u_i, v_j)

The Matching Algorithm - Bipartite Graphs

Lemma

Say the minimum weight perfect matching is unique. Call it M and let its weight be w. Then $|B| \neq 0$ and moreover the highest power of 2 dividing $|B|$ is 2^w

The Matching Algorithm - Bipartite Graphs

Lemma

Say the minimum weight perfect matching is unique. Call it M and let its weight be w. Then $|B| \neq 0$ and moreover the highest power of 2 dividing $|B|$ is 2^w

- First, note that any perfect matching corresponds to some permutation $\sigma \in S_n$, so we can define the value of a perfect matching by the following:

$$value(\sigma) = \prod_{i=1}^{n} b_{i\sigma(i)}$$

The Matching Algorithm - Bipartite Graphs

Lemma

Say the minimum weight perfect matching is unique. Call it M and let its weight be w. Then $|B| \neq 0$ and moreover the highest power of 2 dividing $|B|$ is 2^w

- First, note that any perfect matching corresponds to some permutation $\sigma \in S_n$, so we can define the value of a perfect matching by the following:

$$value(\sigma) = \prod_{i=1}^{n} b_{i\sigma(i)}$$

- Then note that $value(\sigma) \neq 0$ iff σ corresponds to a perfect matching (since σ corresponds to a perfect matching iff $(u_i, v_{\sigma(i)})$ is an edge for $1 \leq i \leq n$)

Proof of the Lemma

- Now note the well-known expansion for the determinant,

$$|B| = \sum_{\sigma \in S_n} sign(\sigma) \times value(\sigma)$$

Proof of the Lemma

- Now note the well-known expansion for the determinant,

$$|B| = \sum_{\sigma \in S_n} sign(\sigma) \times value(\sigma)$$

- Let σ_M be the permutation for the minimum weight perfect matching, so $value(\sigma) = 2^w$

Proof of the Lemma

- Now note the well-known expansion for the determinant,

$$|B| = \sum_{\sigma \in S_n} sign(\sigma) \times value(\sigma)$$

- Let σ_M be the permutation for the minimum weight perfect matching, so $value(\sigma) = 2^w$
- For any other matching, the corresponding weight $w' > w$, so that the permutation corresponding to it has a value that is a higher power of 2

Proof of the Lemma

- Now note the well-known expansion for the determinant,

$$|B| = \sum_{\sigma \in S_n} sign(\sigma) \times value(\sigma)$$

- Let σ_M be the permutation for the minimum weight perfect matching, so $value(\sigma) = 2^w$
- For any other matching, the corresponding weight $w' > w$, so that the permutation corresponding to it has a value that is a higher power of 2
- Thus, 2^w must be the highest power of 2 that divides $|B|$

Another Lemma

Lemma

Let M be the unique minimum weight perfect matching and let its weight be w. Then the edge (u_i, v_j) is in M iff $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$ is odd

Another Lemma

Lemma

Let M be the unique minimum weight perfect matching and let its weight be w. Then the edge (u_i, v_j) is in M iff $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$ is odd

- Notice that (from the determinant expansion):

$$|B_{ij}|2^{w_{ij}} = \sum_{\sigma \in S_n : \sigma(i) = j} sign(\sigma) \times value(\sigma)$$

Another Lemma

Lemma

Let M be the unique minimum weight perfect matching and let its weight be w. Then the edge (u_i, v_j) is in M iff $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$ is odd

- Notice that (from the determinant expansion):

$$|B_{ij}|2^{w_{ij}} = \sum_{\sigma \in S_n : \sigma(i) = j} sign(\sigma) \times value(\sigma)$$

- If $(u_i, v_j) \in M$, then note that one permutation in the above, σ_M, has value 2^w, and everything else has a value that is 0 or a higher power of 2, so $|B_{ij}|2^{w_{ij}}$ is an odd multiple of 2^w

Another Lemma

Lemma

Let M be the unique minimum weight perfect matching and let its weight be w. Then the edge (u_i, v_j) is in M iff $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$ is odd

- Notice that (from the determinant expansion):

$$|B_{ij}|2^{w_{ij}} = \sum_{\sigma \in S_n : \sigma(i) = j} sign(\sigma) \times value(\sigma)$$

- If $(u_i, v_j) \in M$, then note that one permutation in the above, σ_M, has value 2^w, and everything else has a value that is 0 or a higher power of 2, so $|B_{ij}|2^{w_{ij}}$ is an odd multiple of 2^w
- On the other hand, if $(u_i, v_j) \notin M$, then every permutation above must have a value that is 0 or a higher power of 2, and thus 2^{w+1} must divide $|B_{ij}|2^{w_{ij}}$

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B

  1. Compute $|B|$, and thus obtain w

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B
  1. Compute $|B|$, and thus obtain w
  2. Compute $adj(B)$, since its $(i, j)^{th}$ entry is the minor $|B_{ij}|$

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B
  1. Compute $|B|$, and thus obtain w
  2. Compute $adj(B)$, since its $(i, j)^{th}$ entry is the minor $|B_{ij}|$
  3. For each edge (u_i, v_j) in parallel, compute $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$; if this quantity is odd, then include it in the matching (the output)

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B
    1. Compute $|B|$, and thus obtain w
    2. Compute $adj(B)$, since its $(i,j)^{th}$ entry is the minor $|B_{ij}|$
    3. For each edge (u_i, v_j) in parallel, compute $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$; if this quantity is odd, then include it in the matching (the output)
- This algorithm will find a perfect matching (if it exists) with a probability of at least $1/2$ (note that checking if the output is a perfect matching is trivial)

Description of the Algorithm

- Now the algorithm to find a perfect matching is fairly simple, once we have generated the random weights and created B

  1. Compute $|B|$, and thus obtain w
  2. Compute $adj(B)$, since its $(i,j)^{th}$ entry is the minor $|B_{ij}|$
  3. For each edge (u_i, v_j) in parallel, compute $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$; if this quantity is odd, then include it in the matching (the output)

- This algorithm will find a perfect matching (if it exists) with a probability of at least $1/2$ (note that checking if the output is a perfect matching is trivial)
- The only non-trivial step here is the computation of the determinant and adjoint of an integer matrix, which is equivalent to finding the inverse of the matrix

Tutte's Theorem

Generalizing this algorithm to general graphs is not very difficult; it only requires us to work with the Tutte matrix instead

Tutte's Theorem

Generalizing this algorithm to general graphs is not very difficult; it only requires us to work with the Tutte matrix instead

Definition

Note that given a graph $G = (V, E)$, its *adjacency matrix* is an $n \times n$ symmetric matrix D with $d_{ij} = 1$ if $(v_i, v_j) \in E$, and $d_{ij} = 0$ otherwise. The *Tutte matrix* is then a skew-symmetric matrix A obtained from D as follows: if $d_{ij} = d_{ji} = 1$, with $j < i$, then replace them with x_{ij} and $-x_{ij}$, so that entries above the diagonal are positive, while leaving 0s unchanged.

Tutte's Theorem

Generalizing this algorithm to general graphs is not very difficult; it only requires us to work with the Tutte matrix instead

Definition

Note that given a graph $G = (V, E)$, its *adjacency matrix* is an $n \times n$ symmetric matrix D with $d_{ij} = 1$ if $(v_i, v_j) \in E$, and $d_{ij} = 0$ otherwise. The *Tutte matrix* is then a skew-symmetric matrix A obtained from D as follows: if $d_{ij} = d_{ji} = 1$, with $j < i$, then replace them with x_{ij} and $-x_{ij}$, so that entries above the diagonal are positive, while leaving 0s unchanged.

Theorem

Let $G = (V, E)$ be a graph with Tutte matrix A. Then $|A| \neq 0$ iff G has a perfect matching

Generalizing the Algorithm

- First obtain a matrix B from the Tutte matrix by substituting the indeterminate x_{ij} with $2^{w_{ij}}$, where w_{ij} is the (random) weight assigned to the edge (v_i, v_j)

Generalizing the Algorithm

- First obtain a matrix B from the Tutte matrix by substituting the indeterminate x_{ij} with $2^{w_{ij}}$, where w_{ij} is the (random) weight assigned to the edge (v_i, v_j)
- The algorithm stated before then generates a perfect matching when applied to the matrix B (this follows from the previous lemmas which can be extended)

Generalizing the Algorithm

- First obtain a matrix B from the Tutte matrix by substituting the indeterminate x_{ij} with $2^{w_{ij}}$, where w_{ij} is the (random) weight assigned to the edge (v_i, v_j)
- The algorithm stated before then generates a perfect matching when applied to the matrix B (this follows from the previous lemmas which can be extended)
- However, these proofs (which involve looking at Tutte matrices) have been skipped here for the sake of brevity

Generalizing the Algorithm

- First obtain a matrix B from the Tutte matrix by substituting the indeterminate x_{ij} with $2^{w_{ij}}$, where w_{ij} is the (random) weight assigned to the edge (v_i, v_j)
- The algorithm stated before then generates a perfect matching when applied to the matrix B (this follows from the previous lemmas which can be extended)
- However, these proofs (which involve looking at Tutte matrices) have been skipped here for the sake of brevity
- The only non-trivial step is equivalent to matrix inversion, for which Pan's randomized algorithm can be used which uses $O(n^{3.5}m)$ processors and $O(\log^2 n)$ time to invert a $n \times n$ matrix with m-bit integer entries

Minimum Weight Perfect Matching

- Given a graph $G = (V, E)$ along with edge weights $w(e)$ given in **unary**, this algorithm can be extended to find a minimum weight perfect matching

Minimum Weight Perfect Matching

- Given a graph $G = (V, E)$ along with edge weights $w(e)$ given in **unary**, this algorithm can be extended to find a minimum weight perfect matching
- First, by scaling up every weight by a factor of mn, we ensure that the minimum weight perfect matching is lighter by at least mn

Minimum Weight Perfect Matching

- Given a graph $G = (V, E)$ along with edge weights $w(e)$ given in **unary**, this algorithm can be extended to find a minimum weight perfect matching
- First, by scaling up every weight by a factor of mn, we ensure that the minimum weight perfect matching is lighter by at least mn
- Then we use a generalization of the isolating lemma by assigning a weight of $mnw(e) + r(e)$, where $r(e)$ is chosen uniformly and independently at random from $[1, 2m]$

Minimum Weight Perfect Matching

- Given a graph $G = (V, E)$ along with edge weights $w(e)$ given in **unary**, this algorithm can be extended to find a minimum weight perfect matching
- First, by scaling up every weight by a factor of mn, we ensure that the minimum weight perfect matching is lighter by at least mn
- Then we use a generalization of the isolating lemma by assigning a weight of $mnw(e) + r(e)$, where $r(e)$ is chosen uniformly and independently at random from $[1, 2m]$
- We can then apply the algorithm stated above, requiring $O(n^{3.5}mW)$ processors, where W is the weight of the heaviest edge - so this problem is in RNC^2 as well if the weights are given in unary

Vertex Weighted Matching

- Note that the problem of finding a maximum matching is easily converted to one of finding a minimum weight perfect matching by extending the graph to a complete graph and providing a weight of 1 all new edges and 0 to existing edges

Vertex Weighted Matching

- Note that the problem of finding a maximum matching is easily converted to one of finding a minimum weight perfect matching by extending the graph to a complete graph and providing a weight of 1 all new edges and 0 to existing edges
- In the vertex-weighted matching problem, a graph $G = (V, E)$ is given along with (positive) vertex weights, and the goal is to find a matching with the highest total vertex weight

Vertex Weighted Matching

- Note that the problem of finding a maximum matching is easily converted to one of finding a minimum weight perfect matching by extending the graph to a complete graph and providing a weight of 1 all new edges and 0 to existing edges
- In the vertex-weighted matching problem, a graph $G = (V, E)$ is given along with (positive) vertex weights, and the goal is to find a matching with the highest total vertex weight
- Note that the desired matching must be a maximum matching

Vertex Weighted Matching

- Note that the problem of finding a maximum matching is easily converted to one of finding a minimum weight perfect matching by extending the graph to a complete graph and providing a weight of 1 all new edges and 0 to existing edges
- In the vertex-weighted matching problem, a graph $G = (V, E)$ is given along with (positive) vertex weights, and the goal is to find a matching with the highest total vertex weight
- Note that the desired matching must be a maximum matching
- Define $V' \subseteq V$ to be a matching set if V' is the set of vertices matched by any maximum matching

Vertex Weighted Matching

- Note that the problem of finding a maximum matching is easily converted to one of finding a minimum weight perfect matching by extending the graph to a complete graph and providing a weight of 1 all new edges and 0 to existing edges
- In the vertex-weighted matching problem, a graph $G = (V, E)$ is given along with (positive) vertex weights, and the goal is to find a matching with the highest total vertex weight
- Note that the desired matching must be a maximum matching
- Define $V' \subseteq V$ to be a matching set if V' is the set of vertices matched by any maximum matching
- The problem then reduces to one of finding the heaviest matching set, along with finding a perfect matching in the set of vertices within the matching set

Vertex Weighted Matching

- For this, sort the vertices by decreasing weight, and use this to induce a lexicographic order on the matching sets

Vertex Weighted Matching

- For this, sort the vertices by decreasing weight, and use this to induce a lexicographic order on the matching sets
- Any matching set is represented by a binary string, where the i^{th} entry is 1 iff the i^{th} heaviest vertex is in the matching set

Vertex Weighted Matching

- For this, sort the vertices by decreasing weight, and use this to induce a lexicographic order on the matching sets
- Any matching set is represented by a binary string, where the i^{th} entry is 1 iff the i^{th} heaviest vertex is in the matching set

Lemma

The heaviest matching set is the lexicographically largest one

Vertex Weighted Matching

- For this, sort the vertices by decreasing weight, and use this to induce a lexicographic order on the matching sets
- Any matching set is represented by a binary string, where the i^{th} entry is 1 iff the i^{th} heaviest vertex is in the matching set

Lemma

The heaviest matching set is the lexicographically largest one

- Say this were not the case, so L (the lexicographically largest matching set) and H (the heaviest matching set) differ at some point, the earliest of which is, say, u

Vertex Weighted Matching

- For this, sort the vertices by decreasing weight, and use this to induce a lexicographic order on the matching sets
- Any matching set is represented by a binary string, where the i^{th} entry is 1 iff the i^{th} heaviest vertex is in the matching set

Lemma

The heaviest matching set is the lexicographically largest one

- Say this were not the case, so L (the lexicographically largest matching set) and H (the heaviest matching set) differ at some point, the earliest of which is, say, u
- Clearly u is matched in L but not H, so consider the symmetric difference of L and H, which must have an alternating even length path to some vertex v

Vertex Weighted Matching

- The symmetric difference of H with this path then contains u instead of v and remains a matching, but this new matching must be heavier as u is heavier than v - a contradiction

Vertex Weighted Matching

- The symmetric difference of H with this path then contains u instead of v and remains a matching, but this new matching must be heavier as u is heavier than v - a contradiction
- Finding the lexicographically largest matching set is known to be in RNC^2, and since we have shown that finding a perfect matching within this matching set is also in RNC^2, it follows that the vertex weighted matching problem is also in RNC^2

Parallel Complexity of Search vs Decision Problems

- Many search problems are reducible to their corresponding decision problem via self-reducibility, but this reduction is not easy to paralellize

Parallel Complexity of Search vs Decision Problems

- Many search problems are reducible to their corresponding decision problem via self-reducibility, but this reduction is not easy to paralellize
- Self-reduction typically leads to the lexicographically first solution, and for many problems finding such a solution is known to be P-complete in spite of efficient parallel algorithms existing for the unrestricted search problem

Parallel Complexity of Search vs Decision Problems

- Many search problems are reducible to their corresponding decision problem via self-reducibility, but this reduction is not easy to paralellize
- Self-reduction typically leads to the lexicographically first solution, and for many problems finding such a solution is known to be P-complete in spite of efficient parallel algorithms existing for the unrestricted search problem
- This was first studied by Karp, Upfal and Wigdersion, who gave an RNC^2 procedure for the search problem by using oracle access to the ranking function

Parallel Complexity of Search vs Decision Problems

- Many search problems are reducible to their corresponding decision problem via self-reducibility, but this reduction is not easy to paralellize
- Self-reduction typically leads to the lexicographically first solution, and for many problems finding such a solution is known to be P-complete in spite of efficient parallel algorithms existing for the unrestricted search problem
- This was first studied by Karp, Upfal and Wigderson, who gave an RNC^2 procedure for the search problem by using oracle access to the ranking function
- Here, we reduce the general search problem to a *weighted decision problem*, with polynomially bounded weights

Parallel Complexity of Search vs Decision Problems

Theorem

Let (S, F) be a set system and O be an oracle for the weighted decision problem of checking if there a set in F with a weight less than k when every element of S is assigned a polynomially bounded weight. Then there is an RNC^1 procedure using the oracle O to find a set in F.

Parallel Complexity of Search vs Decision Problems

Theorem

Let (S, F) be a set system and O be an oracle for the weighted decision problem of checking if there a set in F with a weight less than k when every element of S is assigned a polynomially bounded weight. Then there is an RNC^1 procedure using the oracle O to find a set in F.

- The procedure is similar to the perfect matching algorithm

Parallel Complexity of Search vs Decision Problems

Theorem

Let (S, F) be a set system and O be an oracle for the weighted decision problem of checking if there a set in F with a weight less than k when every element of S is assigned a polynomially bounded weight. Then there is an RNC^1 procedure using the oracle O to find a set in F.

- The procedure is similar to the perfect matching algorithm
- First, we find the minimum weight for any set in F via binary search, which takes $O(\log n)$ calls to O

Parallel Complexity of Search vs Decision Problems

Theorem

Let (S, F) be a set system and O be an oracle for the weighted decision problem of checking if there a set in F with a weight less than k when every element of S is assigned a polynomially bounded weight. Then there is an RNC^1 procedure using the oracle O to find a set in F.

- The procedure is similar to the perfect matching algorithm
- First, we find the minimum weight for any set in F via binary search, which takes $O(\log n)$ calls to O
- We can then determine if some element is in the minimum weight set by increasing its weight and checking if the minimum weight set increased in weight (with one call to O)

Parallel Complexity of Search vs Decision Problems

- The above process can clearly be done in parallel

Parallel Complexity of Search vs Decision Problems

- The above process can clearly be done in parallel
- Note that the isolating lemma would be used to ensure (with high probability) that the minimum weight set in F is unique

Parallel Complexity of Search vs Decision Problems

- The above process can clearly be done in parallel
- Note that the isolating lemma would be used to ensure (with high probability) that the minimum weight set in F is unique
- Using this, we find an RNC^2 procedure for the **exact matching** problem, which is not known to be solvable in deterministic polynomial time

Parallel Complexity of Search vs Decision Problems

- The above process can clearly be done in parallel
- Note that the isolating lemma would be used to ensure (with high probability) that the minimum weight set in F is unique
- Using this, we find an RNC^2 procedure for the **exact matching** problem, which is not known to be solvable in deterministic polynomial time
- In the **exact matching** problem, a graph $G = (V, E)$ is given, with a subset $E' \subseteq E$ of **red edges**, and the goal is to find a perfect matching with exactly k red edges, for some positive integer k

Exact Matching

- The set system contains all the perfect matchings with exactly k red edges

Exact Matching

- The set system contains all the perfect matchings with exactly k red edges
- If we assign weights randomly to the edges and assume that there is a unique minimum weight perfect matching with exactly k red edges, then we can use the following NC^2 procedure (due to Lovasz) to find that perfect matching

Exact Matching

- The set system contains all the perfect matchings with exactly k red edges
- If we assign weights randomly to the edges and assume that there is a unique minimum weight perfect matching with exactly k red edges, then we can use the following NC^2 procedure (due to Lovasz) to find that perfect matching
- Take the Tutte matrix and replace the indeterminates of the non-red edges with 2^w, where w is the weight of the corresponding edge, and the indeterminates of the red edges with $2^w y$, where y is another indeterminate

Exact Matching

- The set system contains all the perfect matchings with exactly k red edges
- If we assign weights randomly to the edges and assume that there is a unique minimum weight perfect matching with exactly k red edges, then we can use the following NC^2 procedure (due to Lovasz) to find that perfect matching
- Take the Tutte matrix and replace the indeterminates of the non-red edges with 2^w, where w is the weight of the corresponding edge, and the indeterminates of the red edges with $2^w y$, where y is another indeterminate
- The resulting matrix, say B, is then skew symmetric, so $|B| = (pf(B))^2$, where $pf(B)$ represents the Pfaffian of B

Exact Matching

- Note that the Pfaffian is given by the formula:

$$pf(B) = \frac{1}{2^{\frac{n}{2}}\frac{n}{2}!}\sum_{\sigma\in S_n} sign(\sigma)\prod_{i=1}^{\frac{n}{2}} a_{\sigma(2i-1)\sigma(2i)}$$

Exact Matching

- Note that the Pfaffian is given by the formula:

$$pf(B) = \frac{1}{2^{\frac{n}{2}}\frac{n}{2}!}\sum_{\sigma \in S_n} sign(\sigma)\prod_{i=1}^{\frac{n}{2}} a_{\sigma(2i-1)\sigma(2i)}$$

- Compute the determinant using the parallel determinant algorithm, and then compute its square root via interpolation

Exact Matching

- Note that the Pfaffian is given by the formula:

$$pf(B) = \frac{1}{2^{\frac{n}{2}}\frac{n}{2}!}\sum_{\sigma \in S_n} sign(\sigma)\prod_{i=1}^{\frac{n}{2}} a_{\sigma(2i-1)\sigma(2i)}$$

- Compute the determinant using the parallel determinant algorithm, and then compute its square root via interpolation
- Then the power of 2 in the coefficient of y^k will be the weight of the minimum weight perfect matching with exactly k edges

Exact Matching

- Note that the Pfaffian is given by the formula:

$$pf(B) = \frac{1}{2^{\frac{n}{2}}\frac{n}{2}!}\sum_{\sigma\in S_n} sign(\sigma)\prod_{i=1}^{\frac{n}{2}} a_{\sigma(2i-1)\sigma(2i)}$$

- Compute the determinant using the parallel determinant algorithm, and then compute its square root via interpolation
- Then the power of 2 in the coefficient of y^k will be the weight of the minimum weight perfect matching with exactly k edges
- This yields an RNC^2 procedure for the exact matching problem

Valiant Vazirani Theorem

- The isolating lemma can also be used to provide a much simpler proof for (a variant of) the Valiant Vazirani theorem, which showed that USAT is NP-Hard under randomized reductions (i.e., USAT can be reduced to SAT)

Valiant Vazirani Theorem

- The isolating lemma can also be used to provide a much simpler proof for (a variant of) the Valiant Vazirani theorem, which showed that USAT is NP-Hard under randomized reductions (i.e., USAT can be reduced to SAT)
- We instead consider the CLIQUE, which is very closely related to SAT (i.e., they are easily inter-reducible)

Valiant Vazirani Theorem

- The isolating lemma can also be used to provide a much simpler proof for (a variant of) the Valiant Vazirani theorem, which showed that USAT is NP-Hard under randomized reductions (i.e., USAT can be reduced to SAT)
- We instead consider the CLIQUE, which is very closely related to SAT (i.e., they are easily inter-reducible)
- CLIQUE is the problem if finding if there is a clique of size k given a graph $G = (V, E)$, while UCLIQUE is the same with the knowledge that there is at most one such clique

Valiant Vazirani Theorem

- The isolating lemma can also be used to provide a much simpler proof for (a variant of) the Valiant Vazirani theorem, which showed that USAT is NP-Hard under randomized reductions (i.e., USAT can be reduced to SAT)
- We instead consider the CLIQUE, which is very closely related to SAT (i.e., they are easily inter-reducible)
- CLIQUE is the problem if finding if there is a clique of size k given a graph $G = (V, E)$, while UCLIQUE is the same with the knowledge that there is at most one such clique
- UCLIQUE is similarly related to USAT

Valiant Vazirani Theorem

The reduction from CLIQUE to UCLIQUE is as follows:

- First, we assign random weights $w(v)$ to every vertex $v \in V$ uniformly and independently from $[1, 2n]$, so that by the isolating lemma, the maximum weight clique (of size k) will be unique with a probability of at least $1/2$

Valiant Vazirani Theorem

The reduction from CLIQUE to UCLIQUE is as follows:

- First, we assign random weights $w(v)$ to every vertex $v \in V$ uniformly and independently from $[1, 2n]$, so that by the isolating lemma, the maximum weight clique (of size k) will be unique with a probability of at least $1/2$
- The transformed graph G' is generated by transforming every vertex $v \in V$ into a clique of $2nk + w(v)$, and then for every edge $(u, v) \in E$, every vertex in the clique corresponding to u is joined to every vertex in the clique corresponding to v

Valiant Vazirani Theorem

The reduction from CLIQUE to UCLIQUE is as follows:

- First, we assign random weights $w(v)$ to every vertex $v \in V$ uniformly and independently from $[1, 2n]$, so that by the isolating lemma, the maximum weight clique (of size k) will be unique with a probability of at least $1/2$
- The transformed graph G' is generated by transforming every vertex $v \in V$ into a clique of $2nk + w(v)$, and then for every edge $(u, v) \in E$, every vertex in the clique corresponding to u is joined to every vertex in the clique corresponding to v
- Finally, choose a random integer $r \in [1, 2nk]$, and let $k' = 2nk^2 + r$, giving us the transformed problem (G', k')

Valiant Vazirani Theorem

- Then the following will hold:
  1. $(G, k) \notin \text{CLIQUE} \implies (G', k') \notin \text{UCLIQUE}$
  2. $(G, k) \in \text{CLIQUE} \implies \mathbb{P}[(G', k') \in \text{UCLIQUE}] \geq \frac{1}{4nk}$

Valiant Vazirani Theorem

- Then the following will hold:
  1. $(G, k) \notin \text{CLIQUE} \implies (G', k') \notin \text{UCLIQUE}$
  2. $(G, k) \in \text{CLIQUE} \implies \mathbb{P}[(G', k') \in \text{UCLIQUE}] \geq \frac{1}{4nk}$
- Note that any clique of size m in G with weight w becomes a clique (in G') of size $2nkm + w$, and $w \in [m, 2nm]$. Thus, if $m < k$ then the resulting clique has size at most $2n(k^2 - 1) < 2nk^2$, and the first claim follows

Valiant Vazirani Theorem

- Then the following will hold:
  1. $(G, k) \notin \text{CLIQUE} \implies (G', k') \notin \text{UCLIQUE}$
  2. $(G, k) \in \text{CLIQUE} \implies \mathbb{P}[(G', k') \in \text{UCLIQUE}] \geq \frac{1}{4nk}$
- Note that any clique of size m in G with weight w becomes a clique (in G') of size $2nkm + w$, and $w \in [m, 2nm]$. Thus, if $m < k$ then the resulting clique has size at most $2n(k^2 - 1) < 2nk^2$, and the first claim follows
- With a probability of at least $1/2$, the maximum weight clique (of size k) is unique, with a weight w^* (say). Then exactly one clique in G' can have a size of $2nk^2 + w^*$, since cliques in G' corresponding to sizes other than k are transformed into cliques of sizes less than $2nk^2$ or more than $2nk(k+1)$, and only one clique in G has weight w^*

Discussion

- The main difficulty in solving combinatorial problems in parallel is that one must coordinate all the processors to find the same solution, while dealing with the fact that searching for solutions with special properties is often P-complete

Discussion

- The main difficulty in solving combinatorial problems in parallel is that one must coordinate all the processors to find the same solution, while dealing with the fact that searching for solutions with special properties is often P-complete
- The isolating lemma uses randomization to get around this, but it remains unclear if such an approach could be used to find a *random* perfect matching, which can help resolve problems such as finding the permanent of a $0/1$ matrix

Discussion

- The main difficulty in solving combinatorial problems in parallel is that one must coordinate all the processors to find the same solution, while dealing with the fact that searching for solutions with special properties is often P-complete
- The isolating lemma uses randomization to get around this, but it remains unclear if such an approach could be used to find a *random* perfect matching, which can help resolve problems such as finding the permanent of a $0/1$ matrix
- An interesting problem to study is how imperfectness in the random source would affect our results; this is usually modelled via an adversarial random source that is effectively a random die with $m+k$ faces, where the first m faces are labelled 1 to m while the adversary chooses labels for the rest of the faces

Discussion

- The main difficulty in solving combinatorial problems in parallel is that one must coordinate all the processors to find the same solution, while dealing with the fact that searching for solutions with special properties is often P-complete
- The isolating lemma uses randomization to get around this, but it remains unclear if such an approach could be used to find a *random* perfect matching, which can help resolve problems such as finding the permanent of a 0/1 matrix
- An interesting problem to study is how imperfectness in the random source would affect our results; this is usually modelled via an adversarial random source that is effectively a random die with $m + k$ faces, where the first m faces are labelled 1 to m while the adversary chooses labels for the rest of the faces
- Finally, it is as yet unknown if the perfect matching problem is in NC

Bipartite Perfect Matching is in Quasi-NC

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])
- Planar bipartite graphs (Datta, Kulkarni, &Roy [DKR10] and Tewari & Vinodchandran [TV12])

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])
- Planar bipartite graphs (Datta, Kulkarni, &Roy [DKR10] and Tewari & Vinodchandran [TV12])

This Paper

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])
- Planar bipartite graphs (Datta, Kulkarni, &Roy [DKR10] and Tewari & Vinodchandran [TV12])

This Paper

Bipartite PM and SEARCH-PM are in quasi-NC.

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])
- Planar bipartite graphs (Datta, Kulkarni, &Roy [DKR10] and Tewari & Vinodchandran [TV12])

This Paper

Bipartite PM and SEARCH-PM are in quasi-NC.
Uniform circuits of depth $O(\log^2 n)$ and size $2^{O(\log^2 n)}$.

Results So Far

There are NC algorithms for special classes of graphs:

- $K_{3,3}-$ free graphs(Vazirani[Vaz89])
- Graphs with poly-many PM's(Grigoriev & Karpinski[GK87], Agrawal, Hoang, & Thierauf[AHT07])
- Bipartite d-regular graphs (Lav, Pippenger, & Valiant [LPV81])
- Planar bipartite graphs (Datta, Kulkarni, &Roy [DKR10] and Tewari & Vinodchandran [TV12])

This Paper

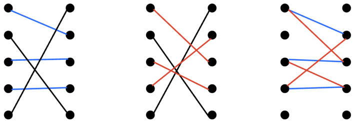
Bipartite PM and SEARCH-PM are in quasi-NC.
Uniform circuits of depth $O(\log^2 n)$ and size $2^{O(\log^2 n)}$.
RNC^2 algorithm for bipartite PM using $O(\log^2 n)$ bits.

Short Overview of Main Ideas

- For any two PM of G, the edges where they differ form disjoint cycles.

Short Overview of Main Ideas

- For any two PM of G, the edges where they differ form disjoint cycles.

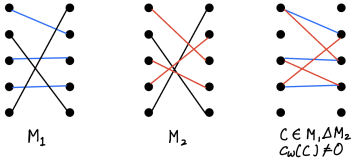


Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.

Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.



Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.

Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- Not obvious if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.

Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- Not obvious if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.
- Subgraph $G' \subseteq G$ which is the union of minimum weight PMs in G. In the bipartite case, graph G' is significantly smaller than G.

Short Overview of Main Ideas

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- Not obvious if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.
- Subgraph $G' \subseteq G$ which is the union of minimum weight PMs in G. In the bipartite case, graph G' is significantly smaller than G.
- We show that G' does not contain any cycle with a nonzero circulation. This means that G' does not contain any small cycles.

Short Overview of Main Ideas

Next, we show that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivates the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.

Short Overview of Main Ideas

Next, we show that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivates the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.

Short Overview of Main Ideas

Next, we show that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivates the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.
- In $\log n$ rounds, we get a unique minimum weight perfect matching.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight *PM* with respect to w.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight PM with respect to w.

If G has a PM & w is isolating, then $det(A) \neq 0$.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight PM with respect to w.

If G has a PM & w is isolating, then $det(A) \neq 0$.As, $2^{w(M)}$ corresponding to the minimum weight PM cannot be canceled with other terms, which are strictly higher powers of 2.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight PM with respect to w.

If G has a PM & w is isolating, then $det(A) \neq 0$.As, $2^{w(M)}$ corresponding to the minimum weight PM cannot be canceled with other terms, which are strictly higher powers of 2.

Isolation Lemma [MVV87]

Let w be a weight assignment chosen uniformly and independently at random from $[2|E|]$. Then w is isolating with probability $\geq 1/2$.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight PM with respect to w.

If G has a PM & w is isolating, then $det(A) \neq 0$.As, $2^{w(M)}$ corresponding to the minimum weight PM cannot be canceled with other terms, which are strictly higher powers of 2.

Isolation Lemma [MVV87]

Let w be a weight assignment chosen uniformly and independently at random from $[2|E|]$. Then w is isolating with probability $\geq 1/2$.

If w is isolating, then computing $det(A_w)$ gives the answer.

An *RNC* Algorithm for SEARCH-*PM*

Isolating Weight Function

A weight function w is isolating if G has a unique minimum weight PM with respect to w.

If G has a PM & w is isolating, then $det(A) \neq 0$.As, $2^{w(M)}$ corresponding to the minimum weight PM cannot be canceled with other terms, which are strictly higher powers of 2.

Isolation Lemma [MVV87]

Let w be a weight assignment chosen uniformly and independently at random from $[2|E|]$. Then w is isolating with probability $\geq 1/2$.

If w is isolating, then computing $det(A_w)$ gives the answer.This can be done in NC^2 (Berkowiz[Ber84]).

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.
For PM M of G, incidence vector $\mathbf{x}^M \in \mathbb{R}^m$ is given by

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.
For PM M of G, incidence vector $\mathbf{x}^M \in \mathbb{R}^m$ is given by

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

$$PM(G) := conv\{\mathbf{x}^M | M \text{ is a } PM \text{ in } G\}$$

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.
For PM M of G, incidence vector $\mathbf{x}^M \in \mathbb{R}^m$ is given by

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

$$PM(G) := conv\{\mathbf{x}^M | M \text{ is a } PM \text{ in } G\}$$

Natural extension of weight function w to $\mathbb{R}^m$:

$$w(\mathbf{x}) = \sum_{e \in E} w(e) x_e$$

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.
For PM M of G, incidence vector $\mathbf{x}^M \in \mathbb{R}^m$ is given by

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

$$PM(G) := conv\{\mathbf{x}^M | M \text{ is a } PM \text{ in } G\}$$

For any matching M, we have $w(M) = w(\mathbf{x}^M)$

Perfect Matching Polytope

$PM(G)$ of $G(V, E)$, $|E| = m$ is a polytope in the edge space, i.e., $PM(G) \subseteq \mathbb{R}^m$.
For PM M of G, incidence vector $\mathbf{x}^M \in \mathbb{R}^m$ is given by

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

$$PM(G) := conv\{\mathbf{x}^M | M \text{ is a } PM \text{ in } G\}$$

Let M^* be a PM in G of minimum weight. Then,

$$w(M^*) = \min\{w(\mathbf{x}) | \mathbf{x} \in PM(G)\}$$

Perfect Matching Polytope

$$x_e^M = \begin{cases} 1, \text{if } e \in M, \\ 0, \text{otherwise.} \end{cases}$$

$$PM(G) := conv\{\mathbf{x}^M | M \text{ is a } PM \text{ in } G\}$$

Lemma

G be a bipartite graph & $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{x} \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \tag{1}$$

$$x_e \geq 0 \quad e \in E, \tag{2}$$

where $\delta(v)$ denotes the set of edges incident on the vertex v.

Derandomise This

$G(V, E)$ **has a PM**.

Derandomise This

$G(V, E)$ **has a PM**.
Cycle C in G is a *nice cycle*, if the subgraph $G - C$ still has a PM.

Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Note that a nice cycle is always an even cycle.

Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Circulation $c_w(C)$

For an even length cycle $C = (v_1, v_2, \cdots, v_k)$

$$c_w(C) := |w(v_1, v_2) - w(v_2, v_3) + w(v_3, v_4) - \cdots - w(v_k, v_1)|$$

Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Lemma

([DKR10]).Let G be a graph with a PM, & let w be a weight function such that all nice cycles in G have nonzero circulation. Then the minimum PM is unique. That is, w is isolating.

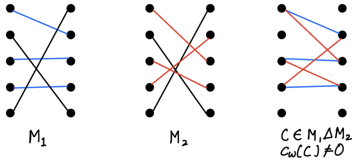
Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Lemma

([DKR10]).Let G be a graph with a PM, & let w be a weight function such that all nice cycles in G have nonzero circulation. Then the minimum PM is unique. That is, w is isolating.



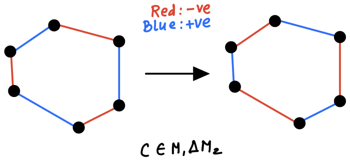
Derandomise This

$G(V, E)$ **has a PM**.

Nice cycle can be obtained from symmetric difference of 2 PMs.

Lemma

([DKR10]).Let G be a graph with a PM, & let w be a weight function such that all nice cycles in G have nonzero circulation. Then the minimum PM is unique. That is, w is isolating.



Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

Let $\{e_i\}_{i \in [m]}$ enumerates E. Define, $w(e_i) = 2^{i-1} \forall i \in [m]$.

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

Let $\{e_i\}_{i \in [m]}$ enumerates E. Define, $w(e_i) = 2^{i-1} \forall i \in [m]$.
Then clearly every cycle has a nonzero circulation. However, we want to achieve this with small weights.

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ *one can construct a set of* $O(n^2 s)$ *weight assignments with weights bounded by* $O(n^2 s)$, *such that for any set of* s *cycles, one of the weight assignments gives nonzero circulation to each of the* s *cycles.* link

Proof.

Let $\{e_i\}_{i \in [m]}$ enumerates E. Define, $w(e_i) = 2^{i-1} \forall i \in [m]$.
Consider weight functions $\{w \pmod{j} | 2 \leq j \leq t\}$.

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

Let $\{e_i\}_{i \in [m]}$ enumerates E. Define, $w(e_i) = 2^{i-1} \forall i \in [m]$.
Consider weight functions $\{w \pmod{j} | 2 \leq j \leq t\}$.
We want to show that for any fixed set of s cycles $\{C_1, C_2, \cdots, C_s\}$, one of these assignments will work, when t is chosen large enough.

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

$$\exists j \leq t \forall i \leq s : c_{w \pmod{j}}(C_j) \neq 0$$

□

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

$$\exists j \leq t \forall i \leq s : c_{w \pmod{j}}(C_j) \neq 0$$

$$\exists j \leq t : \prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{j}$$

□

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

$$\exists j \leq t \forall i \leq s : c_{w \pmod{j}}(C_i) \neq 0$$

$$lcm(2, 3, \cdots, t) \nmid \prod_{i=1}^{s} c_w(C_i)$$

□

Forcing Nonzero Circulation

Lemma

([CRS95]).For any $s \in \mathbb{N}$ one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of s cycles, one of the weight assignments gives nonzero circulation to each of the s cycles. link

Proof.

$$lcm(2, 3, \cdots, t) \nmid \prod_{i=1}^{s} c_w(C_i)$$

Set $lcm(2, 3, \cdots, t) > \prod_{i=1}^{s} c_w(C_i)$. We know $\prod_{i=1}^{s} c_w(C_i) < 2^{n^2 s} \& lcm(2, 3, \cdots, t) > 2^t$ for $t \geq 7$. Choosing $t = n^2 s$ suffices. □

Union of Minimum Weight Perfect Matchings

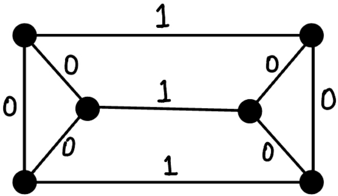
Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.

Union of Minimum Weight Perfect Matchings

Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.



Union of Minimum Weight Perfect Matchings

Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.

Proof.

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be all the minimum weight PMs,i.e., corners of $PM(G)$ corresponding to weight q.

Union of Minimum Weight Perfect Matchings

Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.

Proof.

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be all the minimum weight PMs,i.e., corners of $PM(G)$ corresponding to weight q.

$$\mathbf{x} = \frac{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k}{k}$$

Union of Minimum Weight Perfect Matchings

Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.

Proof.

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be all the minimum weight PMs,i.e., corners of $PM(G)$ corresponding to weight q.

$$\mathbf{x} = \frac{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k}{k}$$

$w(\mathbf{x}) = q$. Consider cycle $C(e_1, e_2, \cdots, e_p)$ with $c_w(C) \neq 0$. Suppose to the contrary $\{e_1, e_2, \cdots, e_p\} \in E_1$.

Union of Minimum Weight Perfect Matchings

Lemma

Let C be a cycle in G such that $c_w(C) \neq 0$. E_1 be the union of all minimum weight PMs in G. $G_1(V, E_1)$ does not contain C.

Proof.

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ be all the minimum weight PMs,i.e., corners of $PM(G)$ corresponding to weight q.

$$\mathbf{x} = \frac{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k}{k}$$

$w(\mathbf{x}) = q$. Consider cycle $C(e_1, e_2, \cdots, e_p)$ with $c_w(C) \neq 0$. Suppose to the contrary $\{e_1, e_2, \cdots, e_p\} \in E_1$.
We show that when we move from point $\mathbf{x}$ along the cycle C, we reach a point in the PM polytope with a weight $< q$. □

Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

Define $\mathbf{y} \in \mathbb{R}^m$:

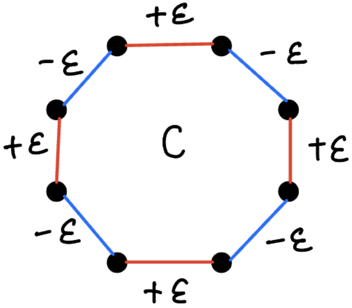
$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.

Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.



Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.

$$w(\mathbf{x}\text{-}\mathbf{y}) = w(\mathbf{x}) - w(\mathbf{y}) = \pm\varepsilon \cdot c_w(C) \neq 0$$

Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.

$$w(\mathbf{x}\text{-}\mathbf{y}) = w(\mathbf{x}) - w(\mathbf{y}) = \pm\varepsilon \cdot c_w(C) \neq 0$$

Choose $\varepsilon > 0$ such that

- $w(\mathbf{y}) < w(\mathbf{x}) = q$
- $y_e \geq 0 \forall e \in E$

Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.

$$w(\mathbf{x}\text{-}\mathbf{y}) = w(\mathbf{x}) - w(\mathbf{y}) = \pm\varepsilon \cdot c_w(C) \neq 0$$

Choose $\varepsilon > 0$ such that

- $w(\mathbf{y}) < w(\mathbf{x}) = q$
- $y_e \geq 0 \forall e \in E$

Now we show that $\mathbf{y} \in PM(G)$.

Define $\mathbf{y} \in \mathbb{R}^m$:

$$y_e = \begin{cases} x_e + (-1)^i \varepsilon, & \text{if} e = e_i, \text{for some } i \in [p], \\ x_e, \text{otherwise} \end{cases}$$

x-y has nonzero coordinates only on C, where its entries are alternating ε and $-\varepsilon$.

$$w(\mathbf{x}\text{-}\mathbf{y}) = w(\mathbf{x}) - w(\mathbf{y}) = \pm\varepsilon \cdot c_w(C) \neq 0$$

Choose $\varepsilon > 0$ such that

- $w(\mathbf{y}) < w(\mathbf{x}) = q$
- $y_e \geq 0 \forall e \in E$

Now we show that $\mathbf{y} \in PM(G)$.But, as $w(\mathbf{y}) < q$ there must be a corner point of $PM(G)$, corresponding to a PM in G with weight $< q$. This would give us a contradiction, completing the proof.

Lemma

G be a bipartite graph & $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{x} \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \tag{3}$$

$$x_e \geq 0 \quad e \in E, \tag{4}$$

where $\delta(v)$ denotes the set of edges incident on the vertex v.

Lemma

G be a bipartite graph & $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{x} \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \tag{3}$$

$$x_e \geq 0 \quad e \in E, \tag{4}$$

where $\delta(v)$ denotes the set of edges incident on the vertex v.

As $y_e \geq 0 \forall e \in E$, the inequality (4) is satisfied. To show (3) is also satisfied, consider $v \in V$:

1. $v \notin C$. Then $y_e = x_e \forall e \in \delta(v)$.

Lemma

G be a bipartite graph & $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{x} \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \tag{3}$$

$$x_e \geq 0 \quad e \in E, \tag{4}$$

where $\delta(v)$ denotes the set of edges incident on the vertex v.

As $y_e \geq 0 \forall e \in E$, the inequality (4) is satisfied. To show (3) is also satisfied, consider $v \in V$:

1. $v \notin C$. Then $y_e = x_e \forall e \in \delta(v)$.
2. $v \in C$. Let $e_j, e_{j+1} \in C$ incident on v.

Lemma

G be a bipartite graph & $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{x} \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \tag{3}$$

$$x_e \geq 0 \quad e \in E, \tag{4}$$

where $\delta(v)$ denotes the set of edges incident on the vertex v.

As $y_e \geq 0 \forall e \in E$, the inequality (4) is satisfied. To show (3) is also satisfied, consider $v \in V$:

1. $v \notin C$. Then $y_e = x_e \forall e \in \delta(v)$.
2. $v \in C$. Let $e_j, e_{j+1} \in C$ incident on $v . y_{e_j} = x_{e_j} + (-1)^j \varepsilon$ and $y_{e_{j+1}} = x_{e_{j+1}} + (-1)^{j+1} \varepsilon$. $\forall$ other edges $e \in \delta(v), y_e = x_e$.

Corollary

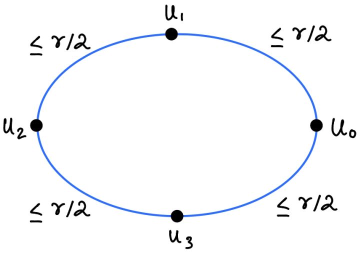
Every PM in $G(V, E_1)$ has the same weight – the minimum weight of any PM in G.

Corollary

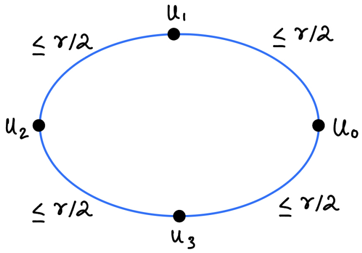
Every PM in $G(V, E_1)$ has the same weight – the minimum weight of any PM in G.

Lemma

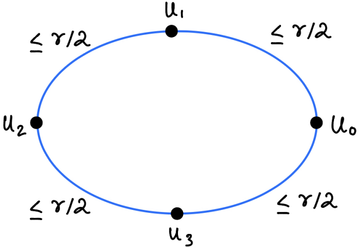
Let H be a graph with n nodes that has no cycles of length $\leq r$. Let $r' = 2r$ when r is even, and $r' = 2r - 2$ otherwise. Then H has $\leq n^4$ cycles of length $\leq r'$.



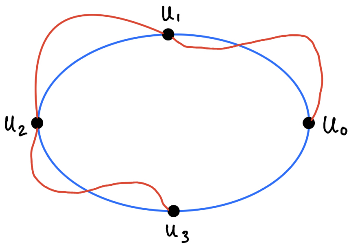
- Let $C = (v_0, v_1, \cdots, v_{l-1})$ be a cycle of length $l \leq r'$.



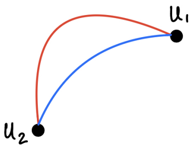
- Let $C = (v_0, v_1, \cdots, v_{l-1})$ be a cycle of length $l \leq r'$. Given a cycle of length $\leq 2r$, choose 4 vertices as shown.



- Let $C = (v_0, v_1, \cdots, v_{l-1})$ be a cycle of length $l \le r'$. Given a cycle of length $\le 2r$, choose 4 vertices as shown.
- This is the only such cycle given $u_0, \cdots, u_3$. If there is another such cycle C' then



- Let $C = (v_0, v_1, \cdots, v_{l-1})$ be a cycle of length $l \leq r'$. Given a cycle of length $\leq 2r$, choose 4 vertices as shown.
- This is the only such cycle given $u_0, \cdots, u_3$. If there is another such cycle C' then
- C' forms a cycle of length at most r. Contradiction.



- Let $C = (v_0, v_1, \cdots, v_{l-1})$ be a cycle of length $l \leq r'$. Given a cycle of length $\leq 2r$, choose 4 vertices as shown.
- This is the only such cycle given $u_0, \cdots, u_3$. If there is another such cycle C' then
- C' forms a cycle of length at most r. Contradiction.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.

G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

- $\forall i \in [k]$ any two PM in G_i have the same weight according to $w_j \forall j \in [i]$.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

- $\forall i \in [k]$ any two PM in G_i have the same weight according to $w_j \forall j \in [i]$.
- By Lemma 3.2 $\forall i \in [k]$ G_i does not have cycles of length $\leq 2^{i+1}$.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

- $\forall i \in [k]$ any two PM in G_i have the same weight according to $w_j \forall j \in [i]$.
- By Lemma 3.2 $\forall i \in [k]$ G_i does not have cycles of length $\leq 2^{i+1}$.
- G_k does not have any cycles as $2^{k+1} \geq n$. Hence, G_k has a unique PM.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

- $\forall i \in [k]$ any two PM in G_i have the same weight according to $w_j \forall j \in [i]$.
- By Lemma 3.2 $\forall i \in [k]$ G_i does not have cycles of length $\leq 2^{i+1}$.
- G_k does not have any cycles as $2^{k+1} \geq n$. Hence, G_k has a unique PM.

Define,

$$w = w_0 B^{k-1} + w_1 B^{k-2} + \cdots + w_{k-1} B_0$$

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

Define,

$$w = w_0 B^{k-1} + w_1 B^{k-2} + \cdots + w_{k-1} B_0$$

- The precedence decreases from w_0 to w_{k1}.

Constructing the Weight assignment

Definition

w_i : weight function such that cycles in G_i of length $\leq 2^{i+2}$ have > 0 circulations.
G_{i+1} : union of minimum weight PMs in G_i due to weight w_i.

Define,

$$w = w_0 B^{k-1} + w_1 B^{k-2} + \cdots + w_{k-1} B_0$$

- The precedence decreases from w_0 to w_{k1}.
- As a consequence, the PMs left in G_i have a strictly smaller weight with respect to w than the ones in G_{i1} that did not make it to G_i.

Constructing the Weight assignment

Lemma

$\forall i \in [k]$ let M_1 be PM's in G_i and M_2 be a PM G_{i-1} which is not in G_i. Then $w(M_1) < w(M_2)$.

Constructing the Weight assignment

Lemma

$\forall i \in [k]$ let M_1 be PM's in G_i and M_2 be a PM G_{i-1} which is not in G_i. Then $w(M_1) < w(M_2)$.

Proof.

M_1, M_2 are PM in G_{i-1}. Hence, $\forall j < i-1$, $w_j(M_1) = w_j(M_2)$. From Corollary 3.3 it follows that $w_{i-1}(M_1) < w_{i-1}(M_2)$. Hence, $w(M_1) < w(M_2)$. □

Constructing the Weight assignment

Lemma

$\forall i \in [k]$ let M_1 be PM's in G_i and M_2 be a PM G_{i-1} which is not in G_i. Then $w(M_1) < w(M_2)$.

Proof.

M_1, M_2 are PM in G_{i-1}. Hence, $\forall j < i-1$, $w_j(M_1) = w_j(M_2)$. From Corollary 3.3 it follows that $w_{i-1}(M_1) < w_{i-1}(M_2)$. Hence, $w(M_1) < w(M_2)$. □

Corollary

The weight assignment w is isolating for G_0.

Constructing the Weight assignment

Lemma

$\forall i \in [k]$ let M_1 be PM's in G_i and M_2 be a PM G_{i-1} which is not in G_i. Then $w(M_1) < w(M_2)$.

Proof.

M_1, M_2 are PM in G_{i-1}. Hence, $\forall j < i-1$, $w_j(M_1) = w_j(M_2)$. From Corollary 3.3 it follows that $w_{i-1}(M_1) < w_{i-1}(M_2)$. Hence, $w(M_1) < w(M_2)$. □

Corollary

The weight assignment w is isolating for G_0.

Decision Problem

Lemma

In quasi-NC^1, one can construct a set of $O(n^6 \log n)$ integer weight functions on $[n/2] \times [n/2]$, where weights have $O(\log^2 n)$ bits, such that for any bipartite graph with n nodes, one of the weight functions is isolating.

One can decide the existence of a perfect matching in a bipartite graph in quasi-NC_2 as follows:

- If the graph has a PM, then one of the weight functions isolates a PM. For this weight function $det(A)$ will be nonzero. When there is no PM, then $det(A)$ will be zero for any weight function.

Decision Problem

Lemma

In quasi-NC^1, one can construct a set of $O(n^6 \log n)$ integer weight functions on $[n/2] \times [n/2]$, where weights have $O(\log^2 n)$ bits, such that for any bipartite graph with n nodes, one of the weight functions is isolating.

One can decide the existence of a perfect matching in a bipartite graph in quasi-NC_2 as follows:

- If the graph has a PM, then one of the weight functions isolates a PM. For this weight function $det(A)$ will be nonzero. When there is no PM, then $det(A)$ will be zero for any weight function.
- Weights have $O(\log^2 n)$ bits, determinant entries have quasi-polynomial bits.

Decision Problem

Lemma

In quasi-NC^1, one can construct a set of $O(n^6 \log n)$ integer weight functions on $[n/2] \times [n/2]$, where weights have $O(\log^2 n)$ bits, such that for any bipartite graph with n nodes, one of the weight functions is isolating.

One can decide the existence of a perfect matching in a bipartite graph in quasi-NC_2 as follows:

- If the graph has a PM, then one of the weight functions isolates a PM. For this weight function $det(A)$ will be nonzero. When there is no PM, then $det(A)$ will be zero for any weight function.
- Weights have $O(\log^2 n)$ bits, determinant entries have quasi-polynomial bits.
- As we need to compute $2^{O(\log^2 n)}$-many determinants in parallel, our algorithm is in quasi-NC^2 with circuit size $2^{O(\log^2 n)}$.

Search Problem

- For a weight function w which is isolating, the algorithm outputs the unique minimum weight PM M. If we have a weight function w_0 which is not isolating, still $det(A)$ might be non-zero with respect to w_0.

Search Problem

- For a weight function w which is isolating, the algorithm outputs the unique minimum weight PM M. If we have a weight function w_0 which is not isolating, still $det(A)$ might be non-zero with respect to w_0.
- In this case, the algorithm computes a set of edges M_0 that might or might not be a perfect matching. However, it is easy to verify if M_0 is indeed a perfect matching, and in this case,we will output M_0.

Search Problem

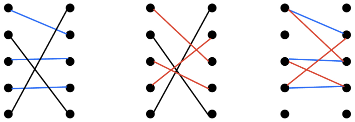
- For a weight function w which is isolating, the algorithm outputs the unique minimum weight PM M. If we have a weight function w_0 which is not isolating, still $det(A)$ might be non-zero with respect to w_0.
- In this case, the algorithm computes a set of edges M_0 that might or might not be a perfect matching. However, it is easy to verify if M_0 is indeed a perfect matching, and in this case,we will output M_0.
- As the algorithm involves computation of similar determinants as in the decision algorithm, it is in quasi-NC_2 with circuit size $2^{O(\log^2 n)}$.

Recap

- For any two PM of G, the edges where they differ form disjoint cycles.

Recap

- For any two PM of G, the edges where they differ form disjoint cycles.

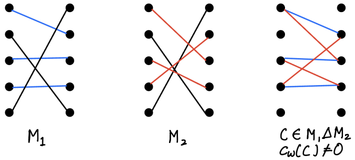


Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.

Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.



Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.

Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- It is not clear if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.

Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- It is not clear if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.
- Then, we consider the subgraph G' of G which is the union of minimum weight PMs in G. In the bipartite case, graph G' is significantly smaller than the original graph G.

Recap

- For a cycle C, its circulation is defined to be the difference of weights of two PMs whose symmetric difference is C.
- Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating.
- It is not clear if $\exists$ such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles.
- Then, we consider the subgraph G' of G which is the union of minimum weight PMs in G. In the bipartite case, graph G' is significantly smaller than the original graph G.
- In particular, we showed that G' does not contain any cycle with a nonzero circulation. This meant that G' does not contain any small cycles.

Recap

Next, we showed that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivated the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.

Recap

Next, we showed that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivated the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.

Recap

Next, we showed that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivated the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.
- In $\log n$ rounds, we get a unique minimum weight perfect matching.

Recap

Next, we showed that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivated the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.
- In $\log n$ rounds, we get a unique minimum weight perfect matching.
- *Instead of quasi-NC, we can get an RNC-circuit but with only* $O(\log^2 n)$ *random bits.*

Recap

Next, we showed that for a graph which has no cycles of length $< r$, the number of cycles of length $< 2r$ is polynomially bounded. This motivated the following strategy which works in $\log n$ rounds:

- in the $i-$th round, assign weights which ensure nonzero circulations for all cycles with length $< 2^i$.
- Since the graph obtained after $(i-1)-$ th rounds has no cycles of length 2^{i-1}, the number of cycles of length 2^i is small.
- In $\log n$ rounds, we get a unique minimum weight perfect matching.
- *Instead of quasi-NC, we can get an RNC-circuit but with only* $O(\log^2 n)$ *random bits.*
- *For complete derandomization, it would suffice to bring the number of random bits down to* $O(\log n)$*. Then there are only polynomially many random strings which can all be tested in NC.*

RNC^2 Algorithms with Few Random Bits

- Decision Version
- Search Version

Decision Version

Theorem

For bipartite graphs, there is an RNC^2-algorithm for PM which uses $O(\log^2 n)$ random bits.

Decision Version

Theorem

For bipartite graphs, there is an RNC^2-algorithm for PM which uses $O(\log^2 n)$ random bits.

We needed quasi-polynomially large circuits for two reasons:

1. we need to try quasi-polynomially many different weight assignments.

Decision Version

Theorem

For bipartite graphs, there is an RNC^2-algorithm for PM which uses $O(\log^2 n)$ random bits.

We needed quasi-polynomially large circuits for two reasons:

1. we need to try quasi-polynomially many different weight assignments.
2. each weight assignment has quasi-polynomially large weights.

Decision Version

Theorem

For bipartite graphs, there is an RNC^2-algorithm for PM which uses $O(\log^2 n)$ random bits.

We needed quasi-polynomially large circuits for two reasons:

1. we need to try quasi-polynomially many different weight assignments.
2. each weight assignment has quasi-polynomially large weights.

We show how to come down to polynomial bounds in both cases by using randomization.

Decision Version

Theorem

For bipartite graphs, there is an RNC^2-algorithm for PM which uses $O(\log^2 n)$ random bits.

We needed quasi-polynomially large circuits for two reasons:

1. we need to try quasi-polynomially many different weight assignments.
2. each weight assignment has quasi-polynomially large weights.

We show how to come down to polynomial bounds in both cases by using randomization.

To solve the first problem, we modify Lemma 2.3 to get a random weight assignment which works with high probability. link

The Modified Lemma

Let G be a graph with n nodes and $s \geq 1$.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers.
Choose a random number p among the first t primes.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers. Choose a random number p among the first t primes.We want to show that with high probability $\prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{p}$.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers. Choose a random number p among the first t primes.We want to show that with high probability $\prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{p}$.
Product is bounded by $2^{n^2 s}$, so it has at most $n^2 s$ prime factors. Choose $t = n^3 s$.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers. Choose a random number p among the first t primes.We want to show that with high probability $\prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{p}$.
Product is bounded by $2^{n^2 s}$, so it has at most $n^2 s$ prime factors. Choose $t = n^3 s$.Then, the random prime works with probability $\geq (1 - 1/n)$.

The Modified Lemma

Lemma

There is a random weight assignment w which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of s cycles, w gives nonzero circulation to each of the s cycles with probability at least $1 - 1/n$.

Proof.

$w(e_i) = 2^{i-1}$.Give exponential weights modulo small prime numbers. Choose a random number p among the first t primes.We want to show that with high probability $\prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{p}$.
Product is bounded by $2^{n^2 s}$, so it has at most $n^2 s$ prime factors. Choose $t = n^3 s$.Then, the random prime works with probability $\geq (1 - 1/n)$.
Now t^{th} prime $\leq 2t \log t = O(n^3 s \log ns)$, by which the weights are bounded by, hence have $O(\log ns)$ bits. □

- We choose each of the weight functions $w_0, w_1, \cdots, w_{k1}$ independently. Probability that all of them provide non-zero circulation on their respective cycles $\geq 1 - k/n \geq 1 - \log n/n$ by union bound.

- We choose each of the weight functions $w_0, w_1, \cdots, w_{k1}$ independently. Probability that all of them provide non-zero circulation on their respective cycles $\geq 1 - k/n \geq 1 - \log n/n$ by union bound.
- Define $A \in \mathbb{R}^{n/2 \times n/2}$ as

$$A(i,j) = \begin{cases} \prod_{i=0}^{k-1} x_i^{w_i(e)}, & \text{if } e = (u_i, v_j) \in E \\ 0, & \textit{otherwise} \end{cases}$$

- We choose each of the weight functions $w_0, w_1, \cdots, w_{k1}$ independently. Probability that all of them provide non-zero circulation on their respective cycles $\geq 1 - k/n \geq 1 - \log n/n$ by union bound.
- Define $A \in \mathbb{R}^{n/2 \times n/2}$ as

$$A(i,j) = \begin{cases} \prod_{i=0}^{k-1} x_i^{w_i(e)}, & \text{if } e = (u_i, v_j) \in E \\ 0, & otherwise \end{cases}$$

- $$det(A) = \sum_{M \text{ PM in } G} sgn(M) \prod_{i=0}^{k-1} x_i^{w_i(M)}$$

- We choose each of the weight functions $w_0, w_1, \cdots, w_{k1}$ independently. Probability that all of them provide non-zero circulation on their respective cycles $\geq 1 - k/n \geq 1 - \log n/n$ by union bound.
- Define $A \in \mathbb{R}^{n/2 \times n/2}$ as

$$A(i,j) = \begin{cases} \prod_{i=0}^{k-1} x_i^{w_i(e)}, & \text{if } e = (u_i, v_j) \in E \\ 0, & otherwise \end{cases}$$

- $$det(A) = \sum_{M \text{ PM in } G} sgn(M) \prod_{i=0}^{k-1} x_i^{w_i(M)}$$

Lemma

$det(A) \neq 0$ if and only if G has a PM.

Complexity

Number of random bits:

For a weight assignment w_i, we need $O(\log ns)$ random bits, where $s = n^4$. Number of random bits required for all w_i's together is $O(k \log n) = O(\log^2 n)$. Finally, we need to plug in $O(\log n)$ random bits for each x_i. This again requires $O(\log^2 n)$ random bits.

Complexity

Number of random bits:

For a weight assignment w_i, we need $O(\log ns)$ random bits, where $s = n^4$. Number of random bits required for all w_i's together is $O(k \log n) = O(\log^2 n)$. Finally, we need to plug in $O(\log n)$ random bits for each x_i. This again requires $O(\log^2 n)$ random bits.

Complexity

Weight construction involves taking exponential weights modulo small primes. Primality testing can be done by the brute force algorithm in NC^2, as the numbers involved have $O(logn)$ bits. Thus, the weight assignments can be constructed in NC^2. Moreover, the determinant with polynomially bounded entries can be computed in NC^2[Ber84].

Complexity

Number of random bits:

For a weight assignment w_i, we need $O(\log ns)$ random bits, where $s = n^4$. Number of random bits required for all w_i's together is $O(k \log n) = O(\log^2 n)$. Finally, we need to plug in $O(\log n)$ random bits for each x_i. This again requires $O(\log^2 n)$ random bits.

Complexity

Weight construction involves taking exponential weights modulo small primes. Primality testing can be done by the brute force algorithm in NC^2, as the numbers involved have $O(logn)$ bits. Thus, the weight assignments can be constructed in NC^2. Moreover, the determinant with polynomially bounded entries can be computed in NC^2[Ber84].

In summary, we get an RNC^2-algorithm that uses $O(\log^2 n)$ random bits.

Search Version

Theorem

For bipartite graphs, there is an RNC^3-algorithm for Search-PM which uses $O(\log^2 n)$ random bits.

- Let $G(V, E)$ be bipartite. Construct weight assignments as before. Let M^* be the unique minimum weight PM in G with respect to the combined weight function w.

Search Version

Theorem

For bipartite graphs, there is an RNC^3-algorithm for Search-PM which uses $O(\log^2 n)$ random bits.

- Let $G(V, E)$ be bipartite. Construct weight assignments as before. Let M^* be the unique minimum weight PM in G with respect to the combined weight function w.
- The bottleneck is its not obvious how to create the graphs $G_1, G_2, \cdots, G_k$ using $O(\log n^2)$ random bits.

Search Version

Theorem

For bipartite graphs, there is an RNC^3-algorithm for Search-PM which uses $O(\log^2 n)$ random bits.

- Let $G(V, E)$ be bipartite. Construct weight assignments as before. Let M^* be the unique minimum weight PM in G with respect to the combined weight function w.
- The bottleneck is its not obvious how to create the graphs $G_1, G_2, \cdots, G_k$ using $O(\log n^2)$ random bits.
- We construct sequence of graphs $\{H_r\}_{i\in[k]}$ such that $H_r \subseteq G_r$ and contains $M^*, \forall r \in [k]$.

Search Version

Theorem

For bipartite graphs, there is an RNC^3-algorithm for Search-PM which uses $O(\log^2 n)$ random bits.

- Let $G(V, E)$ be bipartite. Construct weight assignments as before. Let M^* be the unique minimum weight PM in G with respect to the combined weight function w.
- The bottleneck is its not obvious how to create the graphs $G_1, G_2, \cdots, G_k$ using $O(\log n^2)$ random bits.
- We construct sequence of graphs $\{H_r\}_{i \in [k]}$ such that $H_r \subseteq G_r$ and contains $M^*, \forall r \in [k]$.
- Once we have $H_k = G_k$ we are done.

Search Version

$H_0 = G$. Suppose we have constructed $H_r(V, E_r)$.

Search Version

$H_0 = G$. Suppose we have constructed $H_r(V, E_r)$. An edge will appear in H_{r+1} only if participates in a matching M with $w_r(M) = w_r(M^*)$. Thus $H_{r+1} \subseteq G_{r+1}$.

Search Version

$H_0 = G$. Suppose we have constructed $H_r(V, E_r)$. An edge will appear in H_{r+1} only if participates in a matching M with $w_r(M) = w_r(M^*)$. Thus $H_{r+1} \subseteq G_{r+1}$.

For $e \in E_r$, define $A_e \in \mathbb{R}^{n/2 \times n/2}$ as

$$A_e(i,j) = \begin{cases} \mathbf{X}_r^{w(e')}, & \text{if } e' = (u_i, v_j) \in E_r - N(e) \\ 0, & \textit{otherwise} \end{cases}$$

where

$$\mathbf{X}_r^{w(e)} = \prod_{i=r}^{k-1} x_i^{w_i(e)}$$

Search Version

$H_0 = G$. Suppose we have constructed $H_r(V, E_r)$. An edge will appear in H_{r+1} only if participates in a matching M with $w_r(M) = w_r(M^*)$. Thus $H_{r+1} \subseteq G_{r+1}$.
For $e \in E_r$, define $A_e \in \mathbb{R}^{n/2 \times n/2}$ as

$$A_e(i,j) = \begin{cases} \mathbf{X}_r^{w(e')}, & \text{if } e' = (u_i, v_j) \in E_r - N(e) \\ 0, & otherwise \end{cases}$$

where

$$\mathbf{X}_r^{w(e)} = \prod_{i=r}^{k-1} x_i^{w_i(e)}$$

$$det(A_e) = \sum_{\substack{M \text{ PM in } H_r \\ e \in M}} sgn(M)\mathbf{X}_r^{w(e)}$$

Search Version

Consider the coefficient c_e of $x_r^{w_r(M^*)}$ in $det(A_e)$,

$$c_e = \sum_{\substack{M \text{ PM in } H_r \\ w_r(M)=w_r(M^*), e \in M}} sgn(M)\mathbf{X}_{r+1}^{w(e)}$$

Search Version

Consider the coefficient c_e of $x_r^{w_r(M^*)}$ in $det(A_e)$,

$$c_e = \sum_{\substack{M \text{ PM in } H_r \\ w_r(M)=w_r(M^*), e\in M}} sgn(M)\mathbf{X}_{r+1}^{w(e)}$$

- Define H_{r+1} to be the union of all edges e for which the polynomial $c_e \neq 0$.

Search Version

Consider the coefficient c_e of $x_r^{w_r(M^*)}$ in $det(A_e)$,

$$c_e = \sum_{\substack{M \text{ PM in } H_r \\ w_r(M)=w_r(M^*), e\in M}} sgn(M)\mathbf{X}_{r+1}^{w(e)}$$

- Define H_{r+1} to be the union of all edges e for which the polynomial $c_e \neq 0$.
- For any edge $e \in M$, the polynomial c_e will contain the term $\mathbf{X}_{r+1}^{w(e)}$. As the matching M is isolated in H_r with respect to the weight vector $(w_{r+1}, \cdots, w_{k1})$, the polynomial c_e is nonzero.

Search Version

Consider the coefficient c_e of $x_r^{w_r(M^*)}$ in $det(A_e)$,

$$c_e = \sum_{\substack{M \text{ PM in } H_r \\ w_r(M)=w_r(M^*), e\in M}} sgn(M)\mathbf{X}_{r+1}^{w(e)}$$

- Define H_{r+1} to be the union of all edges e for which the polynomial $c_e \neq 0$.
- For any edge $e \in M$, the polynomial c_e will contain the term $\mathbf{X}_{r+1}^{w(e)}$. As the matching M is isolated in H_r with respect to the weight vector $(w_{r+1}, \cdots, w_{k1})$, the polynomial c_e is nonzero.

In NC^2 , we can construct the weight assignments and compute the determinants in each round. As we have $k = O(\log n)$ rounds, the overall complexity becomes NC^3.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10] $\rightarrow NC^2$, [MV,00] $\rightarrow NC^3$

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow$ NC^2, [MV,00]$\rightarrow$ NC^3

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}
  - For each edge e, check if deletion reduces the above number.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}
  - For each edge e, check if deletion reduces the above number.
  - Run the above step for n^6 possibilities of w_{i-1};

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}
  - For each edge e, check if deletion reduces the above number.
  - Run the above step for n^6 possibilities of w_{i-1}; One of them ensures that there are no cycles of length $\leq 2^{i+1}$ in G_i.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10]$\rightarrow NC^2$, [MV,00]$\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}
  - For each edge e, check if deletion reduces the above number.
  - Run the above step for n^6 possibilities of w_{i-1}; One of them ensures that there are no cycles of length $\leq 2^{i+1}$ in G_i.
- Finding shortest cycle is in NC^2.

Bipartite Planar Graphs

- State-of-art: [MN,95],[DKR,10] $\rightarrow NC^2$, [MV,00] $\rightarrow NC^3$

Using above techniques, we can devise an alternate NC^3 algorithm.

Key Idea: #PerfectMatchings of minimum weight can be counted using Pfaffian orientation.

Algorithm:

- In ith round, compute G_i explicitly.
  - Count #MinWtPerfectMatchings in G_{i-1}
  - For each edge e, check if deletion reduces the above number.
  - Run the above step for n^6 possibilities of w_{i-1}; One of them ensures that there are no cycles of length $\leq 2^{i+1}$ in G_i.
- Finding shortest cycle is in NC^2.
- Total Complexity: NC^3 for $\log n$ rounds.

Conclusion

- The major open question remains whether one can do isolation with polynomially bounded weights.

Conclusion

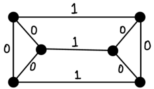
- The major open question remains whether one can do isolation with polynomially bounded weights.
- Our construction requires quasi-polynomial weights because it takes $\log n$ rounds to reach a unique PM and the graphs obtained in the successive rounds cannot be constructed.

Conclusion

- The major open question remains whether one can do isolation with polynomially bounded weights.
- Our construction requires quasi-polynomial weights because it takes $\log n$ rounds to reach a unique PM and the graphs obtained in the successive rounds cannot be constructed.
- For non-bipartite graphs, our approach fails in its first step: There can be matchings other than min-weight matchings when we go from G_i to G_{i+1}.

Conclusion

- The major open question remains whether one can do isolation with polynomially bounded weights.
- Our construction requires quasi-polynomial weights because it takes $\log n$ rounds to reach a unique PM and the graphs obtained in the successive rounds cannot be constructed.
- For non-bipartite graphs, our approach fails in its first step: There can be matchings other than min-weight matchings when we go from G_i to G_{i+1}.



Conclusion

- The major open question remains whether one can do isolation with polynomially bounded weights.
- Our construction requires quasi-polynomial weights because it takes $\log n$ rounds to reach a unique PM and the graphs obtained in the successive rounds cannot be constructed.
- For non-bipartite graphs, our approach fails in its first step: There can be matchings other than min-weight matchings when we go from G_i to G_{i+1}.
- Svensson and Tarnawski [ST,17] generalized Quasi-NC^3 for General graphs.