Let $G = (V, E)$ be an undirected graph. A subset of edges $M \subseteq E$ is a matching in G if no two edges in M share a vertex. Matchings in graphs have a number of important applications and many interesting mathematical and structural properties. There is extensive literature on the topic.

Defn: A matching M is perfect if every vertex is incident to an edge of M.

Algorithmic questions of interest are:

- does G have a perfect matching?
- what is a maximum cardinality matching in G?
- given weights on edges what is a max weight matching?
- min cost perfect matching
- ...

In bipartite graphs most of the above questions can be answered via reductions to max-flow and min-cost flow.

Non-bipartite graphs require different techniques based on "blossoms"

that were first identified in the seminal work of Jack Edmonds in the 60's. Following earlier work of Tutte and Berge on structural aspects.

Here we will consider linear algebraic approach to matchings and showcase the effectiveness in coming up with parallel algorithms.

Some mathematical preliminaries

For an integer $n \geq 1$ we let S_n be the set of $n!$ permutations over $\{1, 2, \ldots, n\}$. These form a group called the symmetric group.

For $\sigma \in S_n$ we let $\sigma(i)$ denote the element that i is mapped to by σ.

For each σ one can assign a sign, denoted by $\mathrm{sgn}(\sigma)$, which is either $+1$ or -1. $\mathrm{sgn}(\sigma)$ is 1 iff the number of inversions in σ is even and -1 if it is odd. Other ways of defining the sgn turn out to be the same.

Let A be a $n \times n$ matrix.
The determinant of A can be defined in several ways. We will need the formula

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i\sigma(i)}$$

Ex: $\begin{bmatrix} a & b \\ c & d \end{bmatrix} = A$

$$\det(A) = ad - bc$$

$$S_2 = \{\underset{\sigma_1}{12}, \underset{\sigma_2}{21}\}$$

Closely related to the determinant is another quantity called the permanent of a square matrix denoted by $per(A)$.

$$per(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

So the only difference is no $sgn(\sigma)$ and this makes all the difference because while $det(A)$ can be efficiently computed, $per(A)$ is hard!

Schwartz-Zippel Lemma

An important tool in randomized algebraic algorithms is the following lemma.

Suppose $p(x_1, \ldots, x_n)$ is a multivariate polynomial in n variables.

Ex: $p(x_1, x_2, x_3) = 3x_1x_2 + 10x_1^2 - x_1x_2x_3^2 + x_3^2 - x_1x_2^3$

The degree of a term such as $x_1x_2x_3^2$ is simply the sum of the degrees of the variables

We say $p(x_1, \ldots, x_n)$ has degree d if the degree of each term is at most d and there is some term with degree equal to d.

Lemma/Theorem: Suppose $p(x_1, \ldots, x_n)$ is a non-zero polynomial of degree at most d whose coefficients of p are all in a field $\mathbb{F}$.

Suppose we pick $a_1, \ldots, a_n$ independently from a set S of distinct elements of $\mathbb{F}$.

Then $\Pr[p(a_1, \ldots, a_n) = 0] \leq \frac{d}{|S|}$.

Easy proof by induction on n.

Base case $n=1$ univariate case

$p(a_1) = 0$ iff a_1 is a root of p.

p has at most d roots over $\mathbb{F}$.

Hence $\Pr[p(a_1)=0] \leq \frac{d}{|S|}$.

Inductive step

$p(x_1, \ldots, x_n) \neq 0$, all n variables occur in some term of p.

Hence $p = \sum_{j=0}^{t} x_1^j \, p_j(x_2, \ldots, x_n)$

where t is the max degree of x_1 in any term.

Since $a_1, \ldots, a_n$ are picked independently we imagine first picking $a_2, \ldots, a_n$ and then picking a_1

$$\Pr\left[p(a_1, \ldots, a_n) \neq 0\right]$$

$$\geq \Pr\left[p(x_1, a_2, \ldots, a_n) \neq 0\right] \cdot \Pr\left[p(a_1, \ldots, a_n) \neq 0 \mid p(x_1, a_2, \ldots, a_n) \neq 0\right]$$

$$\geq \Pr\left[p_t(x_1, a_2, \ldots, a_n) \neq 0\right] \cdot \Pr\left[\quad\right]$$

$$\geq \left(1 - \frac{d-t}{|S|}\right)\left(1 - \frac{t}{|S|}\right)$$

(by induction, by induction)

$$\geq \left(1 - \frac{d}{|S|}\right)$$

$$\Rightarrow \Pr\left[p(a_1, \ldots, a_n) = 0\right] \leq \frac{d}{|S|}$$

□

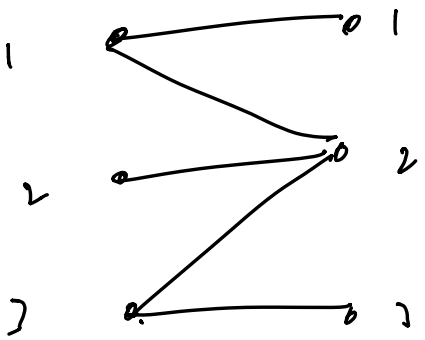
Bipartite Graphs

Suppose $G = (A \cup B, E)$ is a bipartite graph that is balanced i.e. $|A| = |B|$. We want to know if G has a perfect matching. Let $n = |A| = |B|$.

Edmonds considered the following symbolic $n \times n$ matrix A where $A_{ij} = 0$ if $ij \notin E$ and $A_{ij} = x_{ij}$ if $ij \in E$

Here x_{ij} is a variable.

Ex.



$$\begin{bmatrix} X_{11} & X_{12} & 0 \\ 0 & X_{22} & 0 \\ 0 & X_{32} & X_{33} \end{bmatrix}$$

Consider $\det(A)$. This is a polynomial in the m variables. In fact a multilinear polynomial.

Claim: $\det(A) \neq 0$ iff G has a perfect matching.

Proof: Recall

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i\sigma(i)}$$

It is easy to see that for any $\sigma \in S_n$ $\prod_{i=1}^{n} A_{i\sigma(i)} = 0$ if there is some i s.t. $(i, \sigma(i)) \notin E$

Similarly $\prod_{i=1}^{n} A_{i\sigma(i)} \neq 0$ if

$(i, \sigma(i)) \in E \quad \forall \; i = 1, 2, \ldots, n$

But in that case σ defines a perfect matching in G.
Moreover the term $\prod_{i=1}^{n} A_{i\sigma(i)}$

cannot be cancelled symbolically because the other terms do not have the same variables exactly.

□.

We cannot hope to evaluate $\det(A)$ symbolically but we can use Schwartz-Zippel lemma since $\det(A)$ is a polynomial of degree at most n.

Suppose we plug in random integers in the range $[1, 2n]$ into each $x_{ij} \neq 0$ and evaluate

Let $\det(A, \bar{a})$ be the value of $\det(A)$ when $\bar{a}$ is substituted into variables.

$$\Pr\left[\det(A, \bar{a}) = 0 \mid \det(A) = 0\right] \leq \frac{1}{2}.$$

By repeating several times we can get reduce the error probability.

The idea of using randomization in this fashion for matchings is due to Lovasz.

Non-bipartite graphs

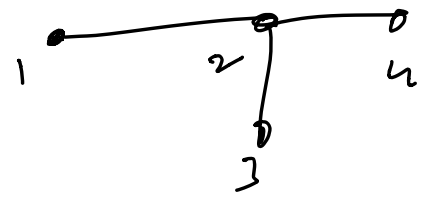
Bill Tutte generalized the approach of Edmonds to general graphs.

The Tutte matrix of a graph $G = (V, E)$ is defined as follows. It is a $n \times n$ matrix where $T_{ij} = x_{ij}$ if $i < j$ and $ij \in E$

and $T_{ij} = -x_{ji}$ if $i > j$

T is a skew-symmetric matrix.

Ex:



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & x_{12} & 0 & 0 \\ -x_{12} & 0 & x_{23} & x_{24} \\ 0 & -x_{23} & 0 & 0 \\ 0 & -x_{24} & 0 & 0 \end{bmatrix} \end{array}$$

Lemma: $\det(T) \neq 0$ iff G has a perfect matching.

The proof is a little more tricky in this case. Again we will use the formula for the determinant:

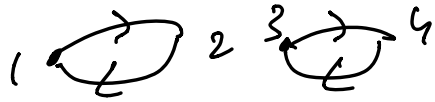
$$\det(T) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i\sigma(i)}$$

We need to understand the structure of permutations in terms of their cycle decomposition. I find it useful to do this via a graph. Suppose $\sigma \in S_n$. We can create a directed graph H_σ on the vertex set $\{1, 2, \ldots, n\}$ as follows. For each i add an arc $(i, \sigma(i))$. Note that if $\sigma(i) = i$ then this creates a self loop. We think of a self loop as an arc leaving i and entering i and as a cycle of length 1. Every i has one arc leaving and one arc entering since σ is a permutation (bijection).

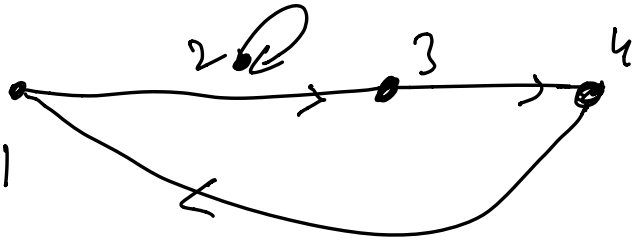
Thus H_σ is a partition of $\{1, 2, \ldots, n\}$ into a collection of disjoint cycles. Often this is called a cycle cover though the terminology can be misleading because the cycles are required to be disjoint. Observe that knowing the cycle cover we can reconstruct σ and hence permutations can be represented via their cycle decomposition.

Ex:

σ	1	2	3	4
	2	1	4	3



σ	1	2	3	4
	3	2	4	1



Note that the orientation of the cycles is important.

σ	1	2	3	4
	4	2	1	3

is what is obtained if we reverse the arcs in the previous graph.

Claim: Suppose H_σ has a self loop then $\prod_{i=1}^{n} T_{i,\sigma(i)} = 0$.

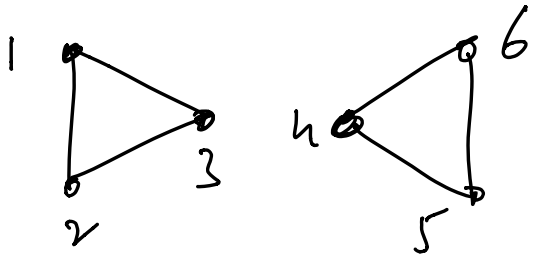
Claim: For $\sigma \in S_n$ $\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0$

$\iff$ $\forall (i,j) \in H_\sigma$ there is an undirected edge $ij \in E(G)$. In particular there are no self-loops in T.

Claim: Suppose H_σ has only even cycles and $\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0$. Then G has a perfect matching.

Claim: Suppose H_σ has a cycle C and σ' is the permutation obtained by reversing the arcs of C. Then $\operatorname{sgn}(\sigma) = \operatorname{sgn}(\sigma')$.

Consider the following graph



Clearly it does not have a perfect matching but there is a permutation $(1\ 2\ 3)(4\ 5\ 6)$ that has the property that

$$\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0.$$

The point is that the skew symmetry in T implies that if reverse one odd cycle in σ to obtain σ' then

$$\prod_{i=1}^{n} T_{i\sigma(i)} = -\prod_{i=1}^{n} T_{i\sigma'(i)}$$

while $\operatorname{sgn}(\sigma) = \operatorname{sgn}(\sigma')$.

Hence the two terms corresponding to σ and σ' will cancel out.

If all cycles in σ are even and $\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0$ then this term cannot be canceled out by any other permutation σ'.

Claim: Let $S' = \{\sigma \in S_n \mid H_\sigma$ has a self loop or σ has an odd length cycle$\}$. Then $\sum_{\sigma \in S'} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i\sigma(i)} = 0$.

Proof: If H_σ has self loop we have $\prod_{i=1}^{n} T_{i\sigma(i)} = 0$ since diagonal entries of T are 0.

Let $S'' = \{\sigma \in S' \mid H_\sigma$ has no self loops$\}$.

We can pair up permutations in S'' as follows. For each $\sigma \in S''$ find the odd cycle with least numbered vertex and reverse the cycle. We get another permutation σ' and it is easy to see that σ''s mate is σ. Further, as we saw, skew symmetry of T

implies that $\prod_{i=1}^{n} T_{i\sigma(i)} = -\prod_{i=1}^{n} T_{i\sigma'(i)}$

Also $\operatorname{sgn}(\sigma) = \operatorname{sgn}(\sigma')$.

Hence the sum is 0.

□.

Claim: Let $\sigma \in S_n \setminus S'$.

If $\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0$ then

$\det(T) \neq 0$.

Proof: Recall all cycles are even in any $\sigma \in S_n \setminus S'$.

Consider the monomial $\prod_{i=1}^{n} T_{i\sigma(i)}$

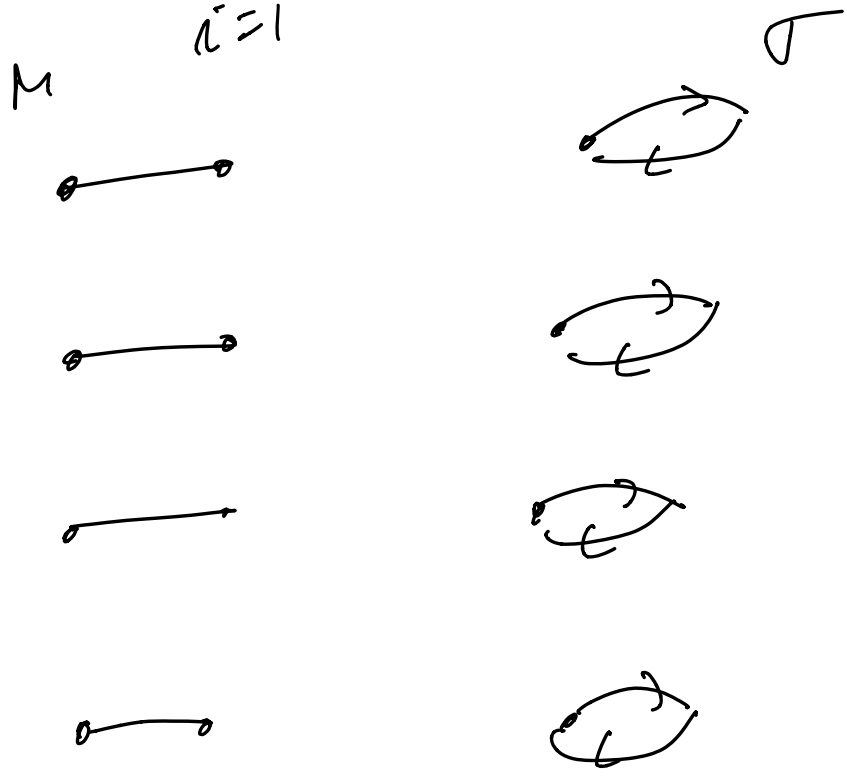
Ignoring the sign we have
n variables that form an
undirected cycle C in G.
The only other terms that
have this same monomial
are obtained by taking an
(even) cycle C' in σ and
reversing the arcs but the
sign remains the same and
hence terms do not cancel. □

!

Now we are ready to complete
the proof. If there is a
perfect matching M in G

then M defines a permutation σ consisting of even cycles

s.t. $\prod_{i=1}^{n} X_{i\sigma(i)} \neq 0$

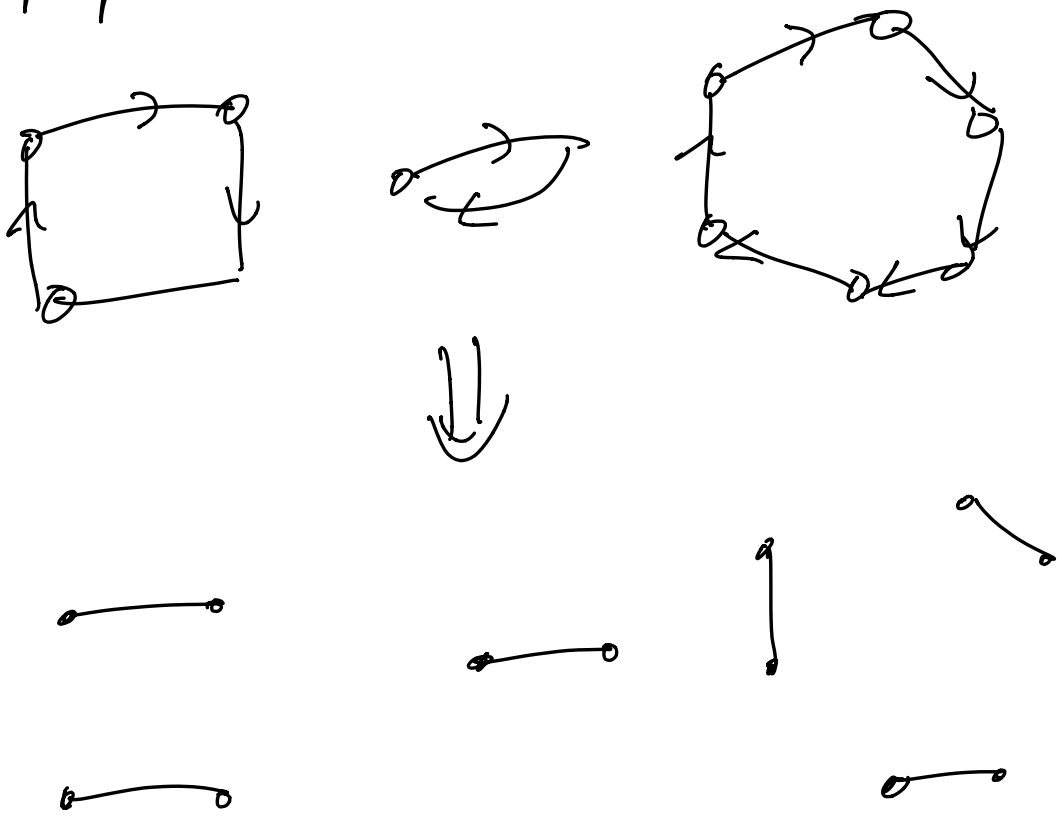


Conversely suppose $\det(T) \neq 0$

$\Rightarrow \exists\ a\ \sigma$ s.t. $\prod_{i=1}^{n} T_{i\sigma(i)} \neq 0$

$\Rightarrow$ there is a subgraph of G

that is a cycle cover with all even cycles. But then there is a perfect matching in G.



□

We can apply the Schwartz-Zippel lemma and deterministic computation

to prove the following

Theorem. There is an $O(n^\omega)$ time randomized algorithm that given a general graph G decides whether G has a perfect matching or not with prob $\geq 1/2$. It is a 1-sided error guarantee. If G has no perfect matching always correct.

Exercise: Reduce the problem of checking whether G has a matching of size $\geq k$ to the perfect matching problem.

Self-reduction to find a matching

We have a randomized algorithm to check if G has a perfect matching. Can we find one?

Yes, via the obvious self-reduction.

Fix an edge e. We can check if there exists a perfect matching containing $e = uv$ by checking whether $G - u - v$ has a PM.

If $G - u - v$ has a PM we output e plus a PM in $G - u - v$.

Otherwise we compute a PM in $G - e$.

Total seen time is $O(m n^\omega)$.

This can be improved to $O(n^\omega)$ by some clever ideas. We may discuss this later.

Parallel algorithm

Is there any advantage of the linear algebraic approach to matchings? Although we may get theoretically faster seen times it depends on impractical (so far) fast matrix multiplication and the seen times only improve for dense graphs over combinatorial methods.

However one of the key advantages of this approach is that it results in the only known fast parallel algorithm.

We assume some basic familiarity with parallel model such as PRAM. In this model we have some $p(n)$ identical processors that have access to shared memory. Time is synchronized. In each step each process can access (read/write) $O(1)$ amount of memory and do $O(1)$ amount of work.

The "depth" of parallel run time is the number of steps to finish the computation. Input is assumed to be written in memory at the start.

NC (for Nick's class named after Nick Pippenger) is the class of problems that can be solved in $O(\log n)$ depth using $poly(n)$ processors. NC^k is those that can be solved in $O(\log^k n)$ depth. Various memory access models are studied.

- EREW is the most restrictive and hence most realistic. Exclusive read exclusive write which means that in each step any memory location can only be accessed by 1 process.

- CREW concurrent read exclusive write.

- CRCW concurrent read/write.

  quite powerful and can result in counter intuitively fast algorithms.

See refs for more on PRAM algorithms. Not only is the depth important

but also total "work".

A problem is considered "parallelizable" efficiently
if it is in NC^k for some fixed k.

RNC: randomized version of NC.

Q: Is perfect matching in NC^k?

This is a big open problem but
the determinant algorithm
yields a RNC^2 algorithm.

This is because $\det(A)$ can be computed in depth $O(\log^2 n)$.
This is easy to see for matrix multiplication and det calculation can be reduced to matrix multiplication.

Can we also find a perfect matching in parallel? This is not as straightforward because the self reduction we saw to reduce the search for a matching to decision is inherently sequential.

Mulmuley, Vazirani & Vazirani in 1987 gave an elegant solution.

Their solution relied on an important observation called the Isolation lemma.

Lemma: Suppose $N = \{1, 2, \ldots, n\}$ is a finite set of n elements and Let $\mathcal{F} \subseteq 2^N$ be any non-empty family. Suppose we pick for each $i \in N$ a weight w_i uniformly and independently from $\{1, 2, \ldots, d\}$. Then $Pr[\exists$ a unique min weight set S in $\mathcal{F}] \geq \frac{n}{d}$

In particular for $d = 2n$ we have $\geq \frac{1}{2}$ prob.

Note that the lemma doesn't impose any structure on $\mathcal{F}$. They could be matchings, spanning trees or any collection of feasible sets. Hence it is applicable to many discrete optimization problem.

Proof: Let $\mathcal{F} = \{S_1, S_2, \ldots, S_K\}$ for some $K \geq 1$. Let $w(S_j) = \sum_{i \in S_j} w_i$.

Define: $\alpha_i = \min_{S_j : i \notin S_j} w(S_j) - \min_{S_j : i \in S_j} (w(S_j) - w_i)$

If $\alpha_i = w_i$ we call i ambiguous.

$$P_\ell[i \text{ is ambiguous}] \leq \frac{1}{d}.$$

because if α_i is an integer and i is picked independently of the elements (can be picked last).

$$\Rightarrow P_\ell[\exists \text{ ambiguous element}] \leq \frac{n}{d}$$

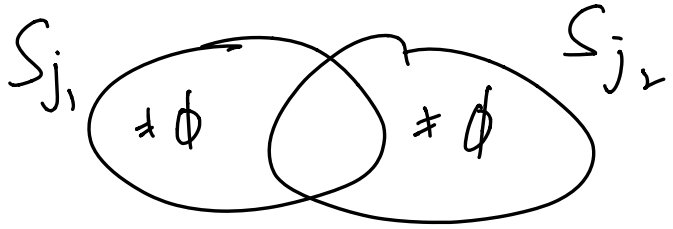
by union bound.

Non-unique min weight set

$\Rightarrow$ there is an ambiguous element. why?

$$w(S_{j_1}) = w(S_{j_2}) = \min_{S_j} w(S_j)$$

with $S_{j_1} \neq S_{j_2}$ say.



Pick i with max weight in

$$(S_{j_1} - S_{j_2}) \cup (S_{j_2} - S_{j_1}).$$

Say $i \in S_{j_1} - S_{j_2}$ w.l.o.g.

Clearly S_{j_1} is min cost set not containing i.

And min $w(S_j) = w(S_{j_1})$
$S_j : i \in S_j$

$\Rightarrow$ $\alpha_i = w_i$ since $w(S_{j_1}) = w(S_{j_2})$.

□.

Idea: Suppose min weight perfect matching is unique. And we know the value W of this min weight matching. Then we can run the self reduction in parallel as follows. First we compute min weight perfect matching weight W. Then each edge e in parallel computes min weight PM in $G-e$ with weight W_e say. Then e is in a min wt matching if $w_e + W_e = W$.

Since min wt matching is unique all edges which pass the test

from the matching!

We only saw how to test whether a graph G has a PM by using random numbers in the Tutte matrix T. We didn't see how to find a min-wt matching.

However the determinant idea combined with the Isolation lemma and the trick of using exponentials to create scores gives an algorithm.

Lemma: Suppose $w_1, \ldots, w_m$ are non-negative integer weights in a graph

$G = (V, E)$ s.t. G has a unique min wt PM. with weight W.

Consider the Tutte matrix T_G where $T_{ij} = 2^{w_{ij}}$ if $i < j$ and $T_{ij} = -2^{w_{ji}}$ if $i > j$.

Then 2^W is the highest power of 2 that divides $|\det(T)|$

Proof: Recall $\det(T) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} T_{i\sigma(i)}$.

All permutations with odd cycles cancel out. Only terms that remain correspond to ~~even~~ cycle covers with even cycles.

$\prod_{i=1}^{n} T_{i\sigma(i)} = \pm 2^{2W}$ for the unique σ that corresponds to the unique min wt PM.

For every other σ with even cycles

$\prod_{i=1}^{n} T_{i\sigma(i)} = \pm 2^{2a}$ a value $a > W$

$\Rightarrow |\det(T)|$ is divisible by 2^{2W} and not by any higher power of 2.

□.

Note that if M is a unique min wt PM in G then $M-e$ is a unique

min wt PM in G-e.

This gives rise to the following parallel algorithm.

1. Pick for each edge independently a random weight $w_e \in \{1, 2, \dots, 2m\}$.

2. Compute $|\det(T_A)|$ with entries $T_{ij} = 2^{w_{ij}}$ for $i<j$ and $T_{ij} = -2^{w_{ji}}$ $i>j$

   and let 2^W be highest power of 2 that divides $|\det(T)|$.

3. Compute for each $e=uv$ in parallel $\det(T_{-e})$ where T_{-e} is the matrix obtained by removing u and v.
   If $\det(T_{-e}) \neq 0$ let 2^{2a} be highest-

power of 2 that divides $|\det(T_e)|$.

If $a + w_e = W$ then add e to M. in parallel.

4. Check that M is a PM.
   If Yes output M. Else reject no PM.

Claim: Alg correct with prob $\geq \frac{1}{2}$.

Time: Size of numbers.
Each has $O(n^2)$ bits. Hence arithmetic operations cannot be assumed to take $O(1)$ time.

But arithmetic operations can be done in parallel in $O(\log n)$ time.
Work increases.

So $\text{poly}(n)$ work but $O(\log^2 n)$ depth.

Recent breakthroughs: Quasi polynomial work but $\text{polylog}(n)$ depth.
See references.