Matchings in Graphs

Lecturer: *Prajakta Nimbhorkar* Meeting: 7
Scribe: *Karteek* 18/02/2010

In this lecture we look at an "efficient" randomized parallel algorithm for each of the following:

- Checking the existence of perfect matching in a bipartite graph.
- Checking the existence of perfect matching in a general graph.
- Constructing a perfect matching for a given graph.
- Finding a min-weight perfect matching in the given weighted graph.
- Finding a Maximum matching in the given graph.

where "efficient" means an NC-algorithm or an RNC-algorithm.

1 Definitions

Definition 1 NC*: The set of problems that can be solved within poly-log time by a polynomial number of processors working in parallel.*

Definition 2 RNC*: The set of problems for which there is a poly-log time algorithm which uses polynomially many processors in parallel and in addition is allowed polynomially many random bits and the following properties hold:*

- *If the correct answer is YES, the algorithm returns YES with probabilty* $\geq \frac{1}{2}$.
- *If the correct answer is NO, the algorithm always returns NO.*

2 Existence of a perfect matching

2.1 Bipartite Graphs

Here we consider the decision problem of checking if a bipartite graph has a perfect matching. Let the given graph be $G = (V, E)$. Let $V = A \cup B$. Since we are looking for perfect matchings, we only look at graphs where $|A| = |B| = n$, because otherwise, there cannot be a perfect matching.

Fact 3 *In a bipartite graph as above, every perfect matching can be seen as a permutation from* S_n *and every permutation from* S_n *can be seen as perfect matching.*

Let the adjacency matrix of G be A. The Edmonds Matrix D of A is defined as follows:
if $A_{i,j} = 1$ then $D_{i,j} = x_{i,j}$
else $D_{i,j} = 0$
where each $x_{i,j}$ is a variable.
Note: dim(D)=dim(A)=$n \times n$

Let det(D) denote the determinant polynomial of D.

det(D)=$\sum_{\sigma \in S_n} Sign(\sigma) \cdot Value(\sigma)$
where $Value(\sigma) = \prod_{i=1}^{n} D_{i,\sigma(i)}$

Claim 4 *G has a perfect matching $\iff$ det(D) is not identically zero.*

Proof:
($\Rightarrow$): If det(D) is not identically zero, then there exists at least one permutation σ that has value(σ)$\neq 0$.
$M = \{(i,j) \in E | \sigma(i) = j\}$ will form a perfect matching of G (from fact).

($\Leftarrow$): if G has a perfect matching M, then M corresponds to a permutation $\sigma \in S_n$. Value(σ) $\neq 0$ since $\forall i$ $A_{i,\sigma(i)} \neq 0$ ■

Fact 5 *Problem of computing the determinant of a matrix is in NC*

Lemma 6 *(Schwartz Zippel):*
Let p be a polynomial of degree n on m variables.
Let Ω be a set of integers, $|\Omega| = N$
then,
$Pr_{\bar{a} \in_R \Omega}[p(\bar{a}) \neq 0] \geq 1 - \frac{n}{N}$
where $\bar{a} = a_1, a_2, ..., a_m$

Claim 7 *Checking if the determinant polynomial p of degree n on m variables is not identically zero can be done in RNC*

Proof: The algorithm is as follows:
Input: Polynomial p on m variables with degree n.
Goal: If p is not identically 0, output 1 with probability $\geq \frac{1}{2}$. If p is indeed identically 0, output 0 with probability 1

Take Ω such that $|\Omega| \geq 2 \times n$

- Choose a random assignment ρ for the m variables from Ω.

- Run the NC algorithm for determinant on D with the assignment ρ.
- If the above step returns zero, then output 0
- Else output 1.

We can observe the following:

- The above algorithm satisfies the goals by virtue of the Schwartz-Zippel Lemma.
- The above algorithm uses polynomially many random bits.

Hence, we conclude that the above algorithm is indeed in RNC. ■

Theorem 8 *The problem of checking if a given bipartite graph $G = (V, E)$ has a perfect matching is in RNC*

Proof: Follows from Claim 4 and Claim 7. ■

2.2 General Graphs

Here we consider the decision problem of checking if a graph has a perfect matching in the more general setting where the graph need not be bipartite.
Let $G = (V, E)$ be the given graph. Let $|V| = 2 \cdot n$. Assume that V is ordered.

Fact 9 *Permutations from S_n without an odd cycle correspond to a perfect matching in G and vice-versa.*

Let the adjacency matrix of G be A.
The Tutte Matrix T of A is defined as follows:

$$\begin{aligned} t_{i,j} &= x_{i,j} \text{ if } a_{i,j} = 1 \text{ and } i < j \\ t_{i,j} &= -x_{j,i} \text{ if } a_{i,j} = 1 \text{ and } j < i \\ t_{i,j} &= 0 \text{ Otherwise .} \end{aligned}$$

where each $x_{i,j}$ is a variable.
Note: T is a skew symmetric matrix and $\dim(T)$=$\dim(A)$=$n \times n$.

Let $\det(T)$ denote the determinant polynomial of T.

Lemma 10 *$\forall \sigma \in S_n$ which have an odd cycle, $\exists \sigma' \in S_n$ such that:*
Value(σ)=− Value(σ')

Proof: Let $P = \{\sigma \in S_n | \sigma$ has an odd cycle$\}$ We can establish a bijection $f : P \to P$ as follows: For each $\sigma \in P$.
Choose the odd cycle in σ which has the least numbered vertex, call it C. Reverse C to get a permutation σ'.
Define $f(\sigma) = \sigma'$
We can observe that f is a bijection by definition and that Value(σ)=−Value($f(\sigma)$). ■

Remark 11 *In the determinant polynomial, all the $\sigma \in P$ get killed by $f(\sigma)$ due the the above lemma. Hence, only the permutations $\sigma \in S_n \setminus P$ survive.*

Claim 12 *G has a perfect matching $\iff$ det(T) is not identically zero.*

Proof:
($\Rightarrow$): If det(D) is not identically zero, then there exists at least one permutation $\sigma \in S_n \setminus P$ that has value(σ)$\neq 0$.
$M = \{(i, j) \in E | \sigma(i) = j\}$ will form a perfect matching of G (from fact).

($\Leftarrow$): if G has a perfect matching M, then M corresponds to a permutation $\sigma \in S_n \setminus P$. σ survives in the polynomial $det(T)$ and Value(σ) $\neq 0$ since $\forall i$ $A_{i,\sigma(i)} \neq 0$ ■

Claim 13 *Checking if the determinant polynomial p of degree n on m variables is not identically zero can be done in RNC*

Proof: Same as claim 7 ■

Theorem 14 *The problem of checking if a given graph $G = (V, E)$ has a perfect matching is in RNC*

Proof: Follows from Claim 12 and Claim 13. ■

3 Constructing a Perfect Matching

Now we give an algorithm to construct a perfect matching in RNC.

3.1 Exactly one perfect matching

Suppose the given graph G has exactly one perfect matching M, then For any $(i,j) \in |E|$, remove vertices i and j along with all their incident edges from G to get G'.

Fact 15 *G' has a perfect matching $\iff$ $(i,j) \in M$*

Consider the following algorithm for constructing the unique perfect matching in G:
The algorithm uses $|E|$ number of processors. i.e., one processor $P_{(i,j)}$ for each edge $(i,j) \in E$.
The algorithm is as follows:
Each processor $P_{(i,j)}$ checks if removal of vertices i and j along with all their incident edges results in a graph that has a perfect matching.
If yes, then $P_{(i,j)}$ outputs 1. Else outputs 0
The edges $(i,j) \in E$ corresponding to processors that output 1 are the edges in the perfect matching.

Claim 16 *(Correctness): The above algorithm indeed outputs 1 for all edges which belong to the perfect matching.*

Proof: Follows from fact 15. ■

Claim 17 *The above algorithm works in RNC.*

Proof: Clearly there are polynomial number of processors. And each processor tests the existence of a perfect matching in a smaller graph. Existence of a perfect matching can be done in RNC by Theorem 14. Hence the above algorithm works in RNC. ■

Theorem 18 *Constructing the perfect matching for a graph G that is known to have exactly one perfect matching can be done in RNC.*

Proof: Follows from the above algorithm, Claim 16, and Claim 17. ■

3.2 More than one perfect matching

Now, we consider the case where the graph G could possibly have more than one perfect matchings. We will need the following lemma:

Lemma 19 *(Isolation Lemma): Let $S = x_1, x_2, ..., x_m$. Let F be a family of subsets of S. Let $w : S \to 1, 2, ..., N$ be a weight function which assigns weights independently at random to each $x \in S$. Define weight of a set $w(Y)$ to be $\sum_{y \in Y} w(y)$. Call w isolating for F if the subset in F that gets minimum weight is unique. Then,*

$$Pr[w \text{ is isolating for } S] \geq 1 - \frac{n}{N}$$

We can think of E as S and the set of all perfect matchings of G as F. We assign a weight $w_{i,j}$ for each edge $(i,j) \in E$ uniformly at random from the range $1, 2, ..., 2|E|$. By Isolation Lemma, $Pr[minweightperfectmatchingisunique] \geq \frac{1}{2}$. We now create a modified Tutte Matrix A of G as follows:

$$a_{i,j} = 2^{w_{i,j}} \text{ if } (i,j) \in E$$

Let the min weight perfect matching of G be M and let $w(M) = W$ Let σ be the permutation corresponding to M.
Then,

$$\begin{aligned} value(\sigma) = & \prod_{i=1}^{n} a_{i,\sigma(i)} \\ = & \quad 2^{2W} \end{aligned}$$

Since each edge weight is being accounted for twice.

Claim 20 *The largest power of 2 that divides $det(A)$ is $2W$.*

Proof: This follows from the fact that the permutation corresponding to the unique minimum weight matching has value 2^{2W}. And all other permutations have a value greater than 2^{2W}. ■

Claim 21 *Let M be the unique minimum weight perfect matching, and let W be weight of M. Then*

$$(i,j) \in M \iff \frac{det(A_{i,j})}{2^{2W}} \cdot 2^{w_{i,j}} \text{ is odd}$$

where $A_{i,j}$ stands for the minor of A obtained by removing the i^th and the j^th vertex along with all the edges incident on them from G.

Proof:
If $(i,j) \in M$, then the permutation that corresponds to M has value $\pm 2^{2W}$ depending on its sign, while every other permutation will have value as a higher power of 2 (or will cancel out if the permutation has an odd cycle). And so, $\frac{det(A_{i,j})}{2^{2W}} \cdot 2^{w_{i,j}}$ would be an odd number by claim 20.
On the other hand, if $(i,j) \notin M$, then every permutation has value 2^k, where $k > 2W$. And hence $\frac{det(A_{i,j})}{2^{2W}} \cdot 2^{w_{i,j}}$ would be an even number. ■

We can now give an efficient radomized parallel algorithm for constructing a perfect matching when the input graph G has a perfect matching as follows:

- Compute the values $det(A)$ and W.
- Compute $adjoint(A)$. $det(A_{i,j}$ is now the entry (j,i) in $adjoint(A)$.

- For each edge (i, j), compute $\frac{det(A_{i,j})}{2^{2W}} \cdot 2^{w_{i,j}}$ in parallel
- Output edge (i, j) iff the above computed value is odd.

Fact 22 *Matrix inversion can be done in RNC using $O(n^{3.5}|E|)$ processors.*

Theorem 23 *Constructing a perfect matching for a graph G that has a perfect matching can be done in RNC.*

Proof: Follows from the algorithm. Randomness is needed to assign weights. And the only non-trivial step is that of computing the adjoint. This can be achieved in NC by fact 22. ■

4 Minimum weight Perfect Matching

In this section we consider the problem of finding a minimum weight perfect matching in a given weighted graph G with edge weights $w(e)$. The edge weights are given as input in unary.
We follow a similar strategy as the previous case - We isolate exactly one perfect matching and then construct that matching like before. Note that if we scale up all edge weights by a multiplicative factor of $|E||V|$, then, the minimum weight perfect matchings will be lighter than the remaining matchings by at least $|E||V|$. After scaling up, we use the Isolation Lemma to isolate one of the possibly many minimum weight perfect matchings.The algorithm is as follows:
Let $|E| = m$ and $|V| = n$

- For each edge $e \in E$, with edge weight w_e, assign new edge weight $w(e) = m \cdot n \cdot w_e + r(e)$, where $r(e)$ is chosen unifromly at random from the range $1, 2, ..., 2m$.
- Run the algorithm for constructing a perfect matching described in previous section.

Since the isolation lemma works in this setting as well, the algorithm would output the minimum weight perfect matching of G similar to the previous section. The reason we need to scale up the weights by a factor of mn is so as to maintain a large gap between the minimum weight perfect matchings and the remaining ones. We need to have a large gap so that the random weights that we add to each weight to isolate a perfect matching should not make the weight of the minimum weight perfect matching become larger than that of some other matching.

5 Maximum Cardinality Matching

Given a graph G with weights on edges $w(e)$, the goal is to find a maximum cardinality matching in G. The idea is to give an algorithm that finds whether G has a matching of cardinality atleast k. Using this algorithm as a subroutine, we can do binary search on k.
The algorithm for finding if G has a matching of cardinality atleast k is as follows:
Let $|V| = n$

- Add $(n-k)$ new vertices to the graph.
- Connect the newly added vertices to every vertex which was already present in G to get G'.
- Run the algorithm for checking existence of perfect matching.
- If the algorithm returns yes, then return Yes. Else return No

Its easy to see that the G' will have a perfect matching if and only if G has a matching of at least size k. Now do binary search on k to obtain the k which is maximal.