Matchings in Graphs

Lecturer: *Jose Mathew* Meeting: 8
Scribe: *Meena Mahajan* 4 March

In general, we do not know of any NC algorithm for deciding whether a graph has a perfect matching or not, although there are randomized NC algorithms. However, in special cases, determinsitic NC algorithms are possible. We consider one such case today. Assume that the input graphs come with a **promise**: there is a fixed polynomial, say n^k, such that if the input graph has n vertices, then it has at most n^k perfect matchings. In this case, we describe deterministic NC algorithms to

1. Decide whether the graph has a perfect matching,
2. Compute the number of perfect matchings, and
3. Construct all the distinct perfect matchings.

This result is due to [GK87]; we follow the presentation from [KR98].

1 Decision in NC

The main idea is to use carefully selected instantiations of the Tutte matrix A of the input graph G. There is a variable x_{ij} for each edge (i, j), and we have seen that $\mathsf{det}(A) \not\equiv 0 \Leftrightarrow$ G has a perfect matching. So checking if G has a perfect matching boils down to checking whether or not $\mathsf{det}(A)$ is identically zero. There are NC algorithms known that given a matrix over integers, or even rationals, compute the determinant of the matrix as output. However, A is a matrix with variables as entries, and $\mathsf{det}(A)$ is a multivariate polynomial, so these algorithms are not applicable. However, we know something about this polynomial:

1. $$\mathsf{det}(A) = \left(\sum_{M \text{ is a perfect matching}} a_M \cdot \mathsf{monomial}(M) \right)^2$$
   where for any set of edges $E' \subset E$, $\mathsf{monomial}(E) = \prod_{i<j,(i,j)\in E'} x_{ij}$, and the coefficients a_M are from the set $\{+1, -1\}$.
2. The degree of the polynomial $\mathsf{det}(A)$ is n if G has a perfect matching, 0 otherwise.
3. The number of terms is at most n^{2k}.

To test this polynomial, we use the Non-Zero-Sum Lemma and the Sparse Test Theorem, stated and proved below.

Lemma 1 (Non-Zero-Sum Lemma) *Let $\alpha_1, \ldots, \alpha_t$ be any pairwise distinct numbers. Let $(a_1, \ldots, a_t)$ be a non-zero vector. Then the following sums are not all zero.*

$$\begin{aligned}
\mathsf{sum}_0 &= a_1\alpha_1^0 + \ldots + a_t\alpha_t^0 \\
\mathsf{sum}_1 &= a_1\alpha_1^1 + \ldots + a_t\alpha_t^1 \\
&\vdots \quad \vdots \quad \vdots \\
\mathsf{sum}_i &= a_1\alpha_1^i + \ldots + a_t\alpha_t^i \\
&\vdots \quad \vdots \quad \vdots \\
\mathsf{sum}_{t-1} &= a_1\alpha_1^{t-1} + \ldots + a_t\alpha_t^{t-1}
\end{aligned}$$

That is, there exists an s, $0 \leq s < t$, such that $\mathsf{sum}_s = \sum_{i=1}^{t} a_i\alpha_i^s \neq 0$.

Proof: Note that

$$\begin{bmatrix} \alpha_1^0 & \alpha_2^0 & \ldots & \alpha_t^0 \\ \alpha_1^1 & \alpha_2^1 & \ldots & \alpha_t^1 \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_t^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \ldots & \alpha_t^{t-1} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_t \end{bmatrix} = \begin{bmatrix} \mathsf{sum}_0 \\ \mathsf{sum}_1 \\ \mathsf{sum}_2 \\ \vdots \\ \mathsf{sum}_{t-1} \end{bmatrix}$$

The matrix on the left is a Vandermonde matrix, and since all the α_i are distinct, it is non-singular. The vector on the left is non-zero. A non-singular transformation cannot take a non-zero vector to the zero vector, so the vector on the right is also non-zero. ■

Theorem 2 (Sparse-Test theorem) *Let P be a multivariate polynomial in N variables $y_1, \ldots, y_N$. Let S be a known upper bound on T, the number of distinct monomials in P. There is an algorithm to test if P is nonero, requiring S evaluations of P at S different input values in parallel, and checking if any of these evaluations is non-zero.*

Proof: The given polynomial P is of the form

$$P(y_1, \ldots, y_N) = \sum_{i=1}^{T} a_i \mathsf{mon}_i(y_1, \ldots, y_N)$$

where mon_i is the monomial corresponding to the ith term.

Choosing distinct primes as the values for the variables, $y_j = p_j$, each term gets a different value. So if $\alpha_i = \mathsf{mon}_i(p_1, \ldots, p_N)$, then all the α_is are distinct. Further, for each s, $P(p_1^s, \ldots, p_N^s) = \sum_{i=1}^{T} a_i\alpha_i^s$. Now, using Lemma 1, we conclude that if P is not indentically zero, then for some $0 \leq s < T \leq S$, $P(p_1^s, \ldots, p_N^s) \neq 0$. On the other hand, if P is identically 0, then $P(p_1^s, \ldots, p_N^s) = 0$ for all s. ■

Returning to the perfect matching problem, the sparse polynomial we want to test for non-zero is precisely $\mathsf{det}(A)$. Since G has at most n^k perfect matchings, $\mathsf{det}(A)$ has at most n^{2k} non-zero terms. Assign a distinct prime p_{ij} to each edge variable x_{ij}. This ensures that all the monomials of $\mathsf{det}(A)$ (corresponding to pairs of perfect matchings) get distinct values. By Theorem 2, it suffices to check if the determinant of A evalautes to a non-zero value for at least one of the n^{2k} settings, where in the rth setting, x_{ij} is instantiated with p_{ij}^r. Since the determinant of integer matrices can be evaluated in NC, and since we need at most n^2 primes and the first n^2 primes are asymptotically less than or equal to n^3, this yields an NC algorithm.

2 Construction in NC

Though the above algorithm tells us whether or not G has a perfect matching, it tells us nothing about what that matching might be. It also does not tell us the count; note that the smallest number s for which the s powers give a non-zero determinant is at most, but not necessarily equal to, the number of perfect matchings. We need a different algorithm to construct even one such perfect matching. We now describe an NC algorithm which constructs **all** the perfect matchings in the graph, and hence also computes the count. The algorithm proceeds as follows:

Let $G = (V, E)$ be the given graph, where $|V| = n$, $|E| = m$, and we are given the promise that G has at most n^k perfect matchings. Assume without loss of generality that the number of edges, m, is a power of 2 and that n is even. Recursively partition the edge set into equal halves, building a complete binary tree with m leaves and depth $\log m$. Each node in the tree is labeled with a subset W of E; its left child has subset W_L and its right child has subset W_R, where W_L and W_R partition W into equal halves. The root is labeled E, and the leaves are labeled with singletons (a single edge). The tree has $O(m)$ nodes.

The idea is to compute, for each set W in this tree, a family $\mathsf{All\text{-}Relevant}(W)$ of sets defined as follows:

$$\mathsf{All\text{-}Relevant}(W) = \{W \cap M \mid \ M \text{ is a perfect matching } \}$$

Since G has at most n^k perfect matchings, the family $\mathsf{All\text{-}Relevant}(W)$ has at most n^k sets, no matter what W is. So we need not worry about the family becoming too large. And the family $\mathsf{All\text{-}Relevant}(E)$ is precisely the set of all perfect matchings.

The computation proceeds bottom-up. At a leaf labeled by the set $W = \{e\}$ where $e = (u, v)$, there are two situations to consider.

- If some perfect matching uses e, then the set $\{e\}$ is in $\mathsf{All\text{-}Relevant}(W)$. This can be tested by using the algorithm of the previous section on the graph obtained by deleting vertices u, v. (The resulting graph, on $n' = n - 2$ vertices, still has at most $n^k = (n' + 2)^k$ perfect matchings.)

- If some perfect matching does not use e, then the empty set $\emptyset$ is in $\mathsf{All\text{-}Relevant}(W)$. This can be tested by using the algorithm of the previous section on the graph obtained

by deleting the edge (u, v). (Again, the resulting graph still has at most n^k perfect matchings.)

Now suppose we are at a node labeled W, and we know $\mathsf{All\text{-}Relevant}(W_1)$ and $\mathsf{All\text{-}Relevant}(W_2)$. We construct the family

$$\mathcal{F} = \{Y_1 \cup Y_2 \mid Y_i \in \mathsf{All\text{-}Relevant}(W_i) \text{ for } i = 1, 2\}$$

Let Y be some subset in $\mathsf{All\text{-}Relevant}(W)$. That is, for some perfect matching M, $W \cap M = Y$. Since W_1, W_2 partition W, it follows that $Y = (W_1 \cap M) \cup (W_2 \cap M)$, and so Y gets into $\mathcal{F}$ in this construction.

The problem is that some spurious sets Y can also get in; not all pairwise unions are relevant for W. So we need to prune $\mathcal{F}$. But this is easy: to test whether any given Y (not necessarily one in $\mathcal{F}$) is in $\mathsf{All\text{-}Relevant}(W)$, we need to check if

- $Y \subseteq W$,
- Y is a matching, and
- there is some perfect matching which uses all the edges in Y and no other edges of W.

The first and second checks are trivial. For the third, let G' be the graph obtained by deleting all endpoints of edges in Y and all edges in $W \setminus Y$. Then G' also has at most n^k perfect matchings, since each perfect matching in G' can be extended by Y to a perfect matching in G. If G' has n' vertices, then $n' = n - 2|Y|$, and so G' has at most $(n' + 2|Y|)^k$ perfect matchings. Use the algorithm of the previous section on G', computing and using an appropriate value k' such that $(n')^{k'} \geq (n' + 2|Y|)^k$.

References

[GK87] D Grigoriev and M Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proceedings of 28th IEEE Conference on Foundations of Computer Science*, pages 166–172. IEEE Computer Society Press, 1987.

[KR98] Marek Karpinski and Wojciech Rytter. *Fast parallel algorithms for graph matching problems*. Oxford University Press, Oxford, 1998. Oxford Lecture Series in Mathematics and its Applications 9.