CSE 632: Analysis of Algorithms II: Randomized Algorithms **Spring 2024**

Lecture 12: Bipartite Perfect Matching

Lecturer: Zongchen Chen

1 Finding a Perfect Matching in Bipartite Graphs

We consider the search problem for bipartite perfect matching: Given a bipartite graph $G = (U \cup V, E)$ where $|U| = |V| = n$, find a perfect matching of G if one exists.

Recall that we have an algorithm to test if G has a perfect matching via polynomial identity testing. One simple idea for finding a perfect matching in G is to use the following sequential procedure: Try an edge $e = uv \in E$ and check if $G \setminus \{u, v\}$ has a perfecting matching or not; if yes, add e to the output perfect matching and repeat on the smaller graph $G \setminus \{u, v\}$; if no, discard e and repeat on $G \setminus e$. Here, $G \setminus \{u, v\}$ is a subgraph of G where we remove vertices u, v and their adjacent edges, and $G \setminus e$ is a subgraph where we remove a single edge e.

Algorithm 1 A sequential algorithm for bipartite perfect matching

```
1: $M \leftarrow \emptyset$
2: while $G$ has an edge do
3:     Choose an arbitrary edge $e = uv$
4:     Check if $G' = G \setminus \{u, v\}$ has a perfect matching or not     ▷ Need failure probability $\leq \frac{1}{2m}$
5:     if YES then
6:         $M \leftarrow M \cup \{e\}$
7:         $G \leftarrow G'$
8:     else
9:         $G \leftarrow G \setminus e$
10:    end if
11: end while
       return $M$
```

Algorithm 1 takes at most m rounds to find a perfect matching. Each round requires $O(\log m)$ trials of bipartite perfect matching testing (with failure probability $\leq 1/2$), so that the failure probability of each round is $\leq \frac{1}{2m}$, and by the union bound Algorithm 1 fails with probability at most $1/2$.

2 Sequential and Parallel Algorithms

In a sequential algorithm, a single processor computes sequentially, and we want the algorithm to run in $\mathrm{poly}(n)$ time. In a parallel algorithm, multiple processors compute simultaneously, and we want the algorithm to run in $\mathrm{polylog}(n)$ time with $\mathrm{poly}(n)$ processors.

Example 1. Summation: Compute $\sum_{i=1}^{n} x_i$. A sequential algorithm takes $O(n)$ time, while a parallel algorithm runs in $O(\log n)$ time with $O(n)$ processors.

Example 2. Matrix multiplication: Compute AB where $A, B \in \mathbb{Z}^{n \times n}$. A sequential algorithm runs in $O(n^{\omega})$ time where $\omega \approx 2.37$ (currently) is the matrix multiplication exponent. A parallel algorithm runs in $O(\log n)$ time with $O(n^3)$ processors.

Example 3. Matrix determinant: Compute $\det(A)$ where $A \in \mathbb{Z}^{n \times n}$. A sequential algorithm runs in $O(n^{\omega})$ time where $\omega \approx 2.37$ (currently) is the matrix multiplication exponent. There is a parallel algorithm which computes the determinant in $O(\log^2 n)$ time with $O(n^{3.5})$ processors, see this paper. This in particular gives an efficient parallel algorithm for testing if a bipartite graph G has a perfect matching or not.

3 Parallel Algorithm for Bipartite Perfect Matching

Our goal is to find a perfect matching in parallel in polylog(n) time. A first idea is to run the sequential Algorithm 1 in parallel. Namely, for each edge $u_i v_j \in E$, do the following in parallel: Check if $G \setminus \{u_i, v_j\}$ contains a perfect matching or not by checking if $\det(A_{ij}) \neq 0$ or not, where A_{ij} is the Tutte matrix A with the i'th row and j'th column removed; If $\det(A_{ij}) \neq 0$ then output $u_i v_j$. Observe that such an algorithm would output a subset of edges $\bigcup_{M \in \mathcal{PM}} M$ where $\mathcal{PM}$ denotes the set of all perfect matchings of G. In particular, if G has a *unique* perfect matching, then this parallel algorithm indeed works.

When there are multiple perfect matchings, we would try to isolate one of them. Our plan is to assign random weights to edges so that G has a unique minimum weight perfect matching M^* and we can check if an edge $e \in M^*$ or not in parallel. The following lemma is helpful to us.

Lemma 4 (Isolation Lemma)**.** *Let $S_1, \dots, S_k \subseteq S$ be subsets of a ground set S where $|S| = m$. Each element $x \in S$ is assigned a random weight $w(x)$ chosen independently and uniformly at random from $\{1, \dots, \ell\}$. For each subset S_i, the weight of S_i is defined as $w(S_i) = \sum_{x \in S_i} w(x)$. Then, we have*

$$\Pr\left(\exists \text{ a unique } S_i \text{ of min weight}\right) \geq 1 - \frac{m}{\ell}.$$

Proof. For $x \in S$, we say x is *tied* if

$$\min_{j \colon x \in S_j} w(S_j) = \min_{j \colon x \notin S_j} w(S_j).$$

Observe that, there exist multiple minimum weight subsets if and only if there exists $x \in S$ which is tied. Fix $x \in S$ and $w(y)$ for all $y \in S \setminus \{x\}$. Define

$$w^+ = \min_{j \colon x \in S_j} w(S_j) - w(x) \quad \text{and} \quad w^- = \min_{j \colon x \notin S_j} w(S_j).$$

Then, x is tied iff $w(x) = w^- - w^+$. Therefore,

$$\Pr\left(x \text{ is tied} \mid w(y),\, y \in S \setminus \{x\}\right) \leq \frac{1}{\ell}.$$

This implies that

$$\Pr\left(x \text{ is tied}\right) \leq \frac{1}{\ell}.$$

Hence, we deduce from the union bound that

$$\Pr\left(\exists \text{ tied } x \in S\right) \leq \frac{m}{\ell}.$$

The lemma then follows. □

For each edge e, choose a weight $w(e)$ independently and uniformly at random from $\{1, \dots, 2m\}$. The weight of a perfect matching $M \in \mathcal{PM}$ is defined as $w(M) = \sum_{e \in M} w(e)$. By the isolation lemma, if G contains a perfect matching, then with probability at least $1/2$ there is a unique perfect matching M^* of minimum weight. Define an $n \times n$ matrix $D = (d_{ij})_{i,j=1}^n$ with entries

$$d_{ij} = \begin{cases} 2^{w(e)}, & \text{if } e = u_i v_j \in E \\ 0, & \text{otherwise} \end{cases}$$

That is, D is the Tutte matrix $A_G = A_G(x)$ evaluated at $x_e = 2^{w(e)}$, $e \in E$. Recall that

$$\det(A_G) = \sum_{M \in \mathcal{PM}} \operatorname{sgn}(\sigma_M) \prod_{e \in M} x_e.$$

Therefore, we obtain

$$\det(D) = \sum_{M \in \mathcal{PM}} \operatorname{sgn}(\sigma_M) \prod_{e \in M} 2^{w(e)} = \sum_{M \in \mathcal{PM}} \operatorname{sgn}(\sigma_M)\, 2^{w(M)}. \tag{1}$$

Claim 5. *1. If G does not contain a perfect matching, then $\det(D) = 0$;*

2. If G has a unique minimum weight perfect matching M^*, then $\det(D) \neq 0$ and the maximum power of 2 dividing $\det(D)$ is $w(M^*)$, i.e.,

$$2^{w(M^*)} \mid \det(D) \quad \textit{and} \quad 2^{w(M^*)+1} \nmid \det(D).$$

Proof. The first claim follows immediately from Eq. (1). For the second, observe that

$$\det(D) = \operatorname{sgn}(\sigma_{M^*})\, 2^{w(M^*)} + 2^{w(M^*)+1} \sum_{\substack{M \in \mathcal{PM} \\ M \neq M^*}} \operatorname{sgn}(\sigma_M)\, 2^{w(M)-w(M^*)-1}.$$

Since $\operatorname{sgn}(\sigma_{M^*}) \in \{\pm 1\}$ is odd and $w(M) - w(M^*) - 1 \geq 0$ for all $M \neq M^*$, the claim follows. ☐

Algorithm 2 A parallel algorithm for bipartite perfect matching

```
 1: For each $e \in E$, choose $w(e)$ independently and u.a.r. from $\{1, \dots, 2m\}$   ▷ $G$ has a unique min weight
    PM $M^*$ with prob. $\geq 1/2$
 2: Compute $\det(D)$ in parallel
 3: if $\det(D) = 0$ then
        return No perfect matching
 4: end if
 5: $w^* \leftarrow$ maximum $k$ such that $2^k \mid \det(D)$   ▷ $w^* = w(M^*)$
 6: for each $e = u_i v_j \in E$ in parallel do
 7:     Compute $\det(D_{ij})$ in parallel
 8:     if $\det(D_{ij}) = 0$ then
 9:         Stop considering $e$
10:     end if
11:     $w_e^* \leftarrow$ maximum $k$ such that $2^k \mid \det(D_{ij})$   ▷ $w_e^* \geq w(M_0^*)$ where $M_0^*$ is a min weight PM in
    $G \setminus \{u_i, v_j\}$; if $e \in M^*$, then $M_0^* = M^* \setminus e$ is the unique min weight PM in $G \setminus \{u_i, v_j\}$
12:     if $w_e^* + w(e) = w^*$ then   ▷ We always have $w_e^* + w(e) \geq w(M_0^*) + w(e) = w(M_0^* \cup \{e\}) \geq w^*$
        return $e$
13:     end if
14: end for
```

As long as the random weights of edges satisfy that G has a unique minimum weight perfect matching, Algorithm 2 successfully finds it in parallel in $\operatorname{polylog}(n)$ time. Thus, the success probability of Algorithm 2 is at least $1/2$ by the isolation lemma.