18.434 Presentation Notes: Parallel algorithm for finding perfect matchings

Mulmuley, Vazirani and Vazirani (1987) give a randomized parallel algorithm for finding a perfect matching in a graph. The first step is to isolate a specific perfect matching, so that the processors can be coordinated to search for the same perfect matching.

Lemma 1 (Isolation Lemma)**.** *Let $S = \{x_1, x_2, ...x_n\}$ and $F = \{S_1, S_2, ...S_k\}$, where each $S_i \subset S$, $1 \leq i \leq k$. For each $x_i \in S$, we pick random weights w_i uniformly and independently from $\{1, 2, ...2n\}$. Then*

$$Pr[\textit{Minimum weight set in } F \textit{ is unique}] \geq \frac{1}{2}.$$

Proof.

- Fix weights of all elements in S except x_i.
- For each i, let $w(S_i) = \sum_{x_j \in S_i} w_j$.
- Let $\alpha_i = \min_{S_j | x_i \notin S_j} w(S_j) - \min_{S_j | x_i \in S_j} w(S_j)$, where we ignore w_i in the last term.
- If $w_i < \alpha_i$, every minimum weight subset contains x_i.
- If $w_i > \alpha_i$, no minimum weight set contains x_i.
- If $w_i = \alpha_i$, x_i is *ambiguous*. This has probability $1/2n$.
- Probability that S contains an ambiguous element is $\leq n \cdot 1/2n = 1/2$.

□

For each edge (v_i, v_j), pick a random weight $w_{i,j}$ uniformly and independently from $\{1, 2, ..., 2m\}$, m is number of edges. By the Isolation Lemma, the minimum weight perfect matching will be unique with probability at least $1/2$.

Definition 1. *The **Tutte matrix** T of G is an n by n matrix with the (i, j) entry given by*

$$t_{i,j} = \begin{cases} x_{i,j} & \textit{if } (v_i, v_j) \in E, i < j \\ -x_{j,i} & \textit{if } (v_i, v_j) \in E, i > j \\ 0 & \textit{otherwise} \end{cases}$$

where the $x_{i,j}$ are independent variables.

Obtain an integer matrix B from the Tutte matrix by substituting $2^{w_{i,j}}$ for each $x_{i,j}$.

Lemma 2. *Suppose the minimum weight perfect matching M in G is unique and has weight w. Then*

(1) $|B| \neq 0$.

(2) The highest power of 2 that divides $|B|$ is 2^{2w}.

(3) An edge (v_i, v_j) belongs to M iff $|B_{i,j}|2^{w_{i,j}}/2^{2w}$ is odd, where $B_{i,j}$ is the matrix obtained by removing the i-th row and j-th column from B.

Proof. (1), (2):

- For each permutation $\sigma \in S_n$, let $\text{value}(\sigma) = \prod_{i=1}^{n} b_{i,\sigma(i)}$.
- $|B| = \sum_{\sigma \in S_n} \text{sign}(\sigma)\,\text{value}(\sigma)$.
- $\text{value}(\sigma) \neq 0$ iff $(i, \sigma(i)) \in E$ for $1 \leq i \leq n$.
- Let *trail* of σ be subgraph consisting of directed edges $(v_i, v_{\sigma(i)})$. If $\text{value}(\sigma) \neq 0$, each vertex is in 2 edges, thus trail is a union of directed cycles.
- Contribution to the sum in $|B|$ from permutations with odd cycles cancel out.
- For permutations with only even cycles:
  - If σ corresponds to a perfect matching M', then $\text{value}(\sigma) = (-1)^{n/2} 2^{2w(M')}$.
  - Otherwise the even cycles in the trail can be partitioned into 2 perfect matchings M_1, M_2. Then $|\,\text{value}(\sigma)| = 2^{w(M_1)+w(M_2)} > 2^{2w}$, since M_1, M_2 are distinct.
- $|B|$ is a sum of powers of 2, exactly one term has exponent $2w$ and the others have strictly greater exponents. Clearly, $|B| \neq 0$ and 2^{2w} is largest power of 2 dividing $|B|$.

Proof of (3):

- $|B_{i,j}| 2^{w_{i,j}} = \sum_{\sigma\,:\,\sigma(i)=j} \text{sign}(\sigma)\,\text{value}(\sigma)$.
- Permutations with odd cycles have zero net contribution to the sum.
- If $(v_i, v_j) \in M$, smallest term in the sum is 2^{2w}, and every other permutation contributes a higher power of 2.
- If $(v_i, v_j) \notin M$, every term in the sum is a power of 2 greater than 2^{2w}.

□

Algorithm to find M:

1. Pick random weights in $\{1, 2, ..., 2n\}$ for each edge in G.
2. Compute $|B|$ and obtain w.
3. If $|B| = 0$, output that there is no perfect matching.
4. Compute $\text{adj}(B)$ to find the minors $|B_{i,j}|$.
5. For each edge (v_i, v_j), do in parallel: Compute $|B_{i,j}| 2^{w_{i,j}} / 2^{2w}$. If it is odd, include (v_i, v_j) in the matching.

With probability $\leq 1/2$, there is no unique minimum weight perfect matching, and this algorithm would fail. We can reduce the error probability by repetitions.

The most computationally intensive step is to compute $|B|$ and $\text{adj}(B)$. There is an algorithm that does this in $O(\log^2 n)$ time using $O(n^{3.5} m)$ processors (Pan, 1985).

Theorem 1. *There is an RNC^2 parallel algorithm which finds a perfect matching in a graph, using $O(n^{3.5} m)$ processors.*