Matching in RNC

1 Overview

Today, we will discuss parallel algorithms for matching, independent set, and sorting.

2 Matching

The matching problem is given a bipartite graph (L, R, E) find a perfect matching. That is, find a subset of edges $M \subseteq E$ such that each node in L and R is incident to exactly one edge (or equivalently each node in L is bijectively matched to exactly one node in R by an edge in E.)

This is a special case of the max flow problem, where all edges have unit capacity.

We will describe a randomized NC algorithm for this problem.

Given this, we look at an abstract notion of a determinate based on our bipartite graph. We consider the matrix where the rows correspond to nodes in L and the columns to nodes in R. Then, we place a variable (referred to historically in this context as an indeterminant) $x_{u,v}$ in location (u, v) if $(u, v) \in E$, and 0 in the entry otherwise.

The determinant of this matrix works out to be:

$$\det(A) = \sum_{\pi} (-1)^{\sigma(\pi)} a_{1,\pi(1)} a_{2,\pi(2)} a_{3,\pi(3)} \cdots$$

That is, it is the sum over all possible matchings of the nodes of the entries in the matrix. If we use our matrix of indeterminants (and zeros), then we see that the $det(A)$ is an expression involving indeterminants. What happens if there is no perfect matching in the graph? Well, then the $det(A)$ is zero. Moreover, since for each matching the terms are different there is no cancellation, so if there is any perfect matching the polynomial corresponding to A is nonzero.

On the other hand, computing this polynomial is not easy since there are an exponential number of terms in the expression.

2.1 Testing for a perfect matching

On the other hand, if numbers were placed in the matrix instead of variables, we could compute the determinant. Indeed, we remark that computing the determinant of a matrix can be done in NC using something called Chistov's algorithm. Basically, it reduces to a series of matrix multiplications.

This leads to our algorithm. We replace the indeterminants by random numbers and evaluate the matrix. If we ever get a nonzero result, then we know the graph had a perfect

matching. We may get zero, when it has many perfect matchings, they could conceivably cancel. We use the following lemma.

LEMMA 1
Given a nonzero polynomial $p(x_1, x_2, \ldots, x_n)$ with max degree d, if we pick $r_1, r_2, \ldots, r_n$ uniformly at random from $\{1, 2, \ldots, 2d\}$, then $Pr[p(r_1, \ldots, r_n) = 0] \leq 1/2$.

This can be seen by induction on the number of variables n as follows. We prove that the probability for a polynomial is at most d/R where the points are chosen from a domain of size R. For 1 variable, any degree d polynomial with d or more zeros is identically equal to zero. Thus, by choosing out of R possibilities, our chances of choosing a bad guy is at worst d/R. Now, we can think of a polynomial on n variables, as a polynomial on 1 variable, with coefficients formed from $n-1$ variable polynomials. Since, this is a nonzero polynomial, at least one of the coefficients is a nonzero polynomial. Consider that the highest such cooefficient has degree d_1 in this variable.

Question 1: Complete the proof.

3 What is the dang matching?

Now, we have to get fancy.

LEMMA 2
Isolation Lemma. *Given a family $\mathcal{F}$ if subsets of $\{1, \ldots n\}$, by randomly choosing element weights $w(i)$ from $\{1, \ldots 2n\}$, the weight of the smallest weight subset is unique with probability 1/2. (The weight of a subset is the sum of the weights of its elements.)*

Given that the possible weights only range from 1 to n^2, and $\mathcal{F}$ could have $\Theta(2^n/\sqrt{n})$ size, this may be suprising. We present the following proof by Joel Spenser (the lemma was used and proven previously by Mulmuley, Vazirani and Vazirani for giving this randomized NC algorithm for matching.)

PROOF:

Define the function α as

$$\alpha(x) = \min_{E \in \mathcal{F}: \S \notin \mathcal{E}} w(E) - \min_{E \in \mathcal{F}: \S \in \mathcal{E}} w(E - \{x\}),$$

for elements x, subsets E and weights $w(i)$. We note that $\alpha(x)$ does not depend on $w(x)$. That is, these are independent random variables.

Since there are $2n$ different values that $w(x)$ can take, we have

$$\Pr[w(x) = \alpha(x)] \leq \frac{1}{2n},$$

which implies that

$$Pr[w(x) = \alpha(x) \text{ for some x }] \leq \frac{1}{2}.$$

Now suppose A and B are both minimal sets from $\mathcal{F}$. Let $y \in A - B$. Then

$$\min_{E \in \mathcal{F}: \dagger \notin \mathcal{E}} \min w(E) = w(B),$$

and

$$\min_{E \in \mathcal{F}: \dagger \in \mathcal{E}} \min w(E - \{y\}) = w(A) - w(y) = w(B) - w(y).$$

This implies that

$$\alpha(y) = w(B) - (w(B) - w(y)) - w(y).$$

This occurs with probability at most 1/2.

□

Question 2: Show that the above argument works for max. (Actually, you can do this by reduction or by examining the argument.)

Question 2 (Optional): Any intuition? Let me know. (Usually, I understand why what I present works. While the proof here is lovely, the intuition is not completely clear, at least to me.)

We can make the probability arbitrarily small since the calculation of the probability came from n/R where R is the size of the weight range that we used above (i.e., $2n$.) For example, if we choose weights in the range $\{1, \ldots, 2n^2\}$ the probability of getting a unique minimal subset is at least $1 - 1/2n$.

To find a matching, we choose w_{ij} at random from $\{1, \ldots, 2n^4\}$ (the number of indeterminants is n^2.) This means that with probability at least $1 - 1/2n^2$ the minimal weight perfect matching is unique.

We form a matrix B by replace x_{ij} with $2^{w_{ij}}$. The number of bits to represent each entry is $2n^4$. (Still polynomial.)

We then compute the determinant. We see that

$$\begin{aligned} det(B) &= \sum_{\pi} (-1)^{\sigma}(\pi) \Pi 2^{w_{i\pi(i)}} && (1) \\ &= \sum_{\pi} (-1)^{\sigma}(\pi) 2^{\sum_i w_{i\pi(i)}} && (2) \end{aligned}$$

The determinant is a bit string where each summand contributes 1 bit. If we assume that the matching with minimal weight is unique, then the lowest order bit is the weight of the minimal matching, since no bit can be set lower than this (since this weight is minimal) and this bit can't be cancelled in the sum (since this matching is unique). (We will need two's complement arithmetic to deal with the fact that this bit may be negative.)

We, in parallel, compute det(B_{ij} where B_{ij} is B with the ith row and jth column removed. If x_ij was in the minimal matching then the minimal weight bit should be lower by exaclyt w_{ij}. That is, there is a matching in the resulting graph of weight $W(M) - w_{ij}$, where M was the original matching.

Here too, the probability that the matching in this subproblem is minimal is at least $1 - 1/2n^2$.

Since, we only run on n^2 subproblems. The probability that any of the matchings in the subproblems fail to be minimal is at most 1/2. Thus, this algorithm succeeds in finding a matching with probability at least 1/2.