15-859E: Advanced Algorithms **CMU, Spring 2015**
Lecture #11: Matrix-based Algorithms for Matchings via Polynomial Identity Testing
February 6, 2015
Lecturer: Anupam Gupta Scribe: David Kurokawa

1 Outline and Background

We look at matrix-based algorithms for computing a perfect matching (PM) or reporting its non-existence. We cover an $\tilde{O}(m \cdot n^\omega)$ approach for general graphs, and an $\tilde{O}(n \cdot n^\omega)$ approach for bipartite graphs.

1.1 State of the Art Algorithms

State of the art algorithms on the perfect matching problem include:

- $O(m\sqrt{n})$ time for bipartite graphs (Hopcroft-Karp algorithm) and general graphs.
- $\tilde{O}(n^{10/7})$ time for sparse bipartite graphs.
- $O(n^\omega)$ time for general graphs.

1.2 The Field $\mathbb{F}_q$

Throughout this write-up, assume whenever we consider polynomials or matrices that these are over some finite field $\mathbb{F}_q$ (where q is the order of the field). It is known that basic arithmetic operations over this field can be done in polylog(q) time and that the determinant and inverse of a matrix in $\mathbb{F}_q^{n\times n}$ can be computed in $O(n^\omega \text{polylog}(q))$ time. Importantly, if q is polynomial in n, then this reduces to $\tilde{O}(n^\omega)$ time.

2 Schwartz-Zippel and Polynomial Identity Testing

Before we begin, let us first establish an important tool: how to check if a (potentially multi-variate) polynomial is zero quickly.

Theorem 11.1 (Schwartz-Zippel)**.** *Let $p(x_1, ..., x_n)$ be a nonzero polynomial of degree at most d and suppose we choose values $Y_1, ..., Y_n$ independently and uniformly at random from $S \subseteq \mathbb{F}_q$. Then*

$$\mathbb{P}\left[p(Y_1, ..., Y_n) = 0\right] \leq d/|S|.$$

Proof. We proceed by induction on n.

Base case ($n = 1$): It is well-known that any single variable polynomial with degree d has at most d roots. Thus we have that $\mathbb{P}\left[p(Y_1) = 0\right] \leq d/|S|$ as desired.

Inductive step: Let k be the highest power of x_n that appears in p and let $q(x_1, ..., x_{n-1})$ and $r(x_1, ..., x_n)$ be the (unique) polynomials such that

$$p(x_1, ..., x_n) = x_n^k q(x_1, ..., x_{n-1}) + r(x_1, ..., x_n)$$

and the highest power of x_n that appears in r is less than k. That is, when dividing p by x_n^k, q is the quotient and r is the remainder.

Now letting E be the event that $q(Y_1, ..., Y_{n-1})$ is zero we find

$$\begin{aligned}\mathbb{P}\left[p(Y_1, ..., Y_n) = 0\right] &= \mathbb{P}\left[p(Y_1, ..., Y_n) = 0 \mid E\right]\mathbb{P}\left[E\right] + \mathbb{P}\left[p(Y_1, ..., Y_n) = 0 \mid \bar{E}\right]\mathbb{P}\left[\bar{E}\right] \\ &\leq \mathbb{P}\left[E\right] + \mathbb{P}\left[p(Y_1, ..., Y_n) = 0 \mid \bar{E}\right]\end{aligned}$$

By the inductive assumption, and noting that q has degree at most $d - k$, we know

$$\mathbb{P}\left[E\right] = \mathbb{P}\left[q(Y_1, ..., Y_{n-1}) = 0\right] \leq (d-k)/|S|$$

and similarly, viewing p as a polynomial only on x_n (with degree k), we know

$$\mathbb{P}\left[p(Y_1, ..., Y_n) = 0 \mid \bar{E}\right] \leq k/|S|.$$

Thus we find

$$\mathbb{P}\left[p(Y_1, ..., Y_n) = 0\right] \leq (d-k)/|S| + k/|S| = d/|S|.$$

□

From this theorem, we are therefore lead to the following algorithm to test whether a polynomial is zero.

Algorithm 1 ZeroPolynomialTest($p(x_1, ..., x_n)$, S)

```
Randomly select a value for x_i from S
if p with the instatiated variables evaluates to 0 then
    return p is the zero polynomial
else
    return p is not the zero polynomial
end if
```

Note that if p is indeed the zero polynomial this algorithm will always correctly classify p. If on the other hand, p is nonzero then by the theorem we have that this algorithm correctly states p is nonzero with probability at least $1 - d/|S|$. Therefore, this algorithm is correct with probability at least $1 - d/|S|$ overall.

3 An $\tilde{O}(m \cdot n^\omega)$ Algorithm for General Graphs

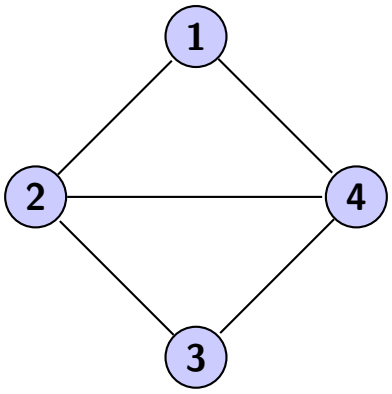
In this section we show how to find a perfect matching in a general graph (or report its non-existence) in $\tilde{O}(m \cdot n^\omega)$ time. To do so, we will largely consider the following matrix.

Definition 11.2. Suppose we have a graph general graph $G = (V, E)$ such that $|V| = n$. Then the Tutte matrix T of G is the $n \times n$ skew-symmetric matrix[1] given by

$$T_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E \text{ or } i = j \\ x_{i,j} & \text{if } (i,j) \in E \text{ and } i < j \\ -x_{j,i} & \text{if } (i,j) \in E \text{ and } i > j. \end{cases}$$

For example, the following graph

[1] A matrix A is said to be skew-symmetric if $A^T = -A$.



has a Tutte matrix of

$$\begin{bmatrix} 0 & x_{1,2} & 0 & x_{1,4} \\ -x_{1,2} & 0 & x_{2,3} & x_{2,4} \\ 0 & -x_{2,3} & 0 & x_{3,4} \\ -x_{1,4} & -x_{2,4} & -x_{3,4} & 0 \end{bmatrix}$$

An important property of this matrix is the following (whose proof we omit).

Theorem 11.3. *The Tutte matrix T of G has nonzero determinant if and only if there exists a perfect matching of G.*

Thus, to check whether a perfect matching exists we need only check if the determinant of the Tutte matrix is zero. Unfortunately, this is quite expensive to check naively and so we therefore utilize Theorem 11.1 to construct a randomized algorithm.

Algorithm 2 PMDecision($G = (V, E)$, S)

$\forall i, j$ where $(i, j) \in E$, choose $x_{i,j}$ (for the Tutte matrix T) randomly from S
if determinant of T with the random entries is zero **then**
    return G does not have a PM
else
    return G has a PM
end if

Now suppose $S = |n^4|$ and the order of $\mathbb{F}$ is approximately $|S|$ (say $\leq 4|S| = 4n^4$), then we find that this algorithm takes $\tilde{O}(n^\omega)$ time and has a probability of error of at most $d/|S| = n/n^4 = 1/n^3$.

To actually compute a perfect matching we do the following.

Algorithm 3 PMSearch($G = (V, E)$, S)

```
if PMDecision(G, S) = no PM then
    return G does not have a PM
end if
M = ∅
while take e ∈ E do
    if PMDecision(G \ {e}, S) = no PM then
        M ← M ∪ {e}
        G ← G \ {u, v}
    else
        G ← G \ {e}
    end if
end while
return M
```

As the loop in this algorithm can run at most m times, we have that this algorithm runs in $\tilde{O}(m \cdot n^\omega)$ time. Furthermore, the probability of failure (if we choose $|S| = n^4$ as before) is by the union bound at most $m/n^3 \leq 1/n$.

4 An $\tilde{O}(n \cdot n^\omega)$ Algorithm for Bipartite Graphs

Here we show how to compute a perfect matching of a bipartite graph in $\tilde{O}(n \cdot n^\omega)$ time. To do so, we will require the use of the Edmonds matrix.

Definition 11.4. Suppose we have a bipartite graph $G = (L, R, E)$ with $|L| = |R| = n$. Then the Edmonds matrix ε of G is the $n \times n$ matrix given by

$$\varepsilon_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E \text{ and } i \in L, j \in R \\ x_{i,j} & \text{if } (i,j) \in E \text{ and } i \in L, j \in R. \end{cases}$$

Via the Leibniz formula for the determinant of a matrix:

$$\sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} \varepsilon_{i,\sigma(i)}$$

and noting that a matching in G corresponds to a permutation $\sigma \in S_n$, we immediately see the following.

1. G has a perfect matching if and only if its Edmonds matrix has nonzero determinant.
2. Suppose $\tilde{\varepsilon}$ is the Edmonds matrix with its variables instantiated with values from some field $\mathbb{F}_q$. Then if $\det(\tilde{\varepsilon}) \neq 0$, there exists some permutation/perfect matching $\pi \in S_n$ such that $\varepsilon_{i,\pi(i)} \neq 0$ for all $i \in \{1, ..., n\}$.

Thus, one algorithm to compute a perfect matching is the following.

Algorithm 4 PMBipartite($G = (L, R, E)$, S)

let $\tilde{\varepsilon}$ be the Edmonds matrix such that the nonzero entries are randomly sampled from S
if $\det(\tilde{\varepsilon}) = 0$ **then**
  return G does not have a PM
end if
return PMBipartiteHelper($\tilde{\varepsilon}$, L, R)

Algorithm 5 PMBipartiteHelper($\tilde{\varepsilon}$, L, R)

for $i \in \{1, ..., n\}$ **do**
  if $\tilde{\varepsilon}_{1,i} \neq 0$ and $\det(\tilde{\varepsilon}_{-1,-i}) \neq 0$ **then**
    let u be the first vertex of L and v be the i^{th} vertex of R
    return $\{(u, v)\} \cup$ PMBipartiteHelper($\tilde{\varepsilon}_{-1,-i}$, $L \setminus \{1\}$, $R \setminus \{i\}$)
  end if
end for

Now suppose $|S| = n^2$ and the order of $\mathbb{F}$ is similar (say $\leq 4|S| = 4n^2$), then we find that this algorithm runs in $\tilde{O}(n^2 \cdot n^\omega)$ time with probability of error at most $d/|S| = n/n^2 = 1/n$. Clearly, this runtime is not desirable — indeed, we gave a faster algorithm for general graphs in the previous section. The critical factor of the the runtime is the number of determinants computed and so we strive to replace the potentially n^2 determinant computations with n inverse computations. To do this, recall the following identity for any $n \times n$ matrix A.

$$\left(A^{-1}\right)_{i,j} = \frac{(-1)^{i+j} \det(A_{-j,-i})}{\det(A)}.$$

In particular, we have then that for any (randomly) instantiated Edmonds matrix $\tilde{\varepsilon}$

$$\left(\tilde{\varepsilon}^{-1}\right)_{i,1} = \frac{(-1)^{i+1} \det(\tilde{\varepsilon}_{-1,-i})}{\det(\tilde{\varepsilon})}.$$

Thus, to compute the $\det(\tilde{\varepsilon}_{-1,-i})$ for all $i \in \{1, ..., n\}$ we need only compute $\tilde{\varepsilon}^{-1}$ and examine the first column. Specifically if we assume $\det(\tilde{\varepsilon}) \neq 0$, then $\det(\tilde{\varepsilon}_{-1,-i}) \neq 0$ if and only if $\left(\tilde{\varepsilon}^{-1}\right)_{i,1} \neq 0$. This leads to the following refinement of Algorithm 5.

Algorithm 6 PMBipartiteHelper($\tilde{\varepsilon}$, L, R)

for $i \in \{1, ..., n\}$ **do**
  compute $\tilde{\varepsilon}^{-1}$
  if $\tilde{\varepsilon}_{1,i} \neq 0$ and $\left(\tilde{\varepsilon}^{-1}\right)_{i,1} \neq 0$ **then**
    let u be the first vertex of L and v be the i^{th} vertex of R
    return $\{(u, v)\} \cup$ PMBipartiteHelper($\tilde{\varepsilon}_{-1,-i}$, $L \setminus \{1\}$, $R \setminus \{i\}$)
  end if
end for

As desired, we have replaced n^2 determinant computations with n inverse computations. As inverses are computed in $O(n^\omega)$ operations each of which take polylog(q) time, we have that Algorithm 4 runs in $O(n \cdot n^\omega)$ time and with the same probability of error as before (i.e. $1/n$).