Reading Report for *Maximum Matchings via Gaussian Elimination*

AntiLeaf

January 2, 2022

Contents

1 Abstract

This is a reading report for a paper *Maximum matchings via Gaussian elimination* [1] published on the 45th FOCS, 2004. The authors of the paper are *Marcin Mucha* and *Piotr Sankowski* from Institute of Informatics of Warsaw University.

In the paper, the authors introduced a fully new algorithm to solve maximum matching problems on both bipartite and general graphs. The algorithm is based mainly on theories of Linear Algebra, instead of combinatorics.

Accroding to the paper, this algorithm has running time $O(n^\omega)$, regardless of whether the graph is bipartite or not, where ω is the exponent of the best known matrix multiplication algorithm. Since $\omega < 2.38$, it breaks through the $O(n^{2.5})$ barrier held by the Edmonds' Blossom algorithm for the matching problem, and comes to the fastest algorithm until now.

Besides, the results in the paper also solved a long-standing open question of, whether Lovász's randomized algorithm on testing whether a graph has a perfect matching in $O(n^\omega)$ time, can be extended to an algorithm finding a perfect matching.

Since there is only a little difference between the algorithms for bipartite graphs and general graphs, the algorithm for general graphs will be focused on, while the one for bipartite graphs will only be mentioned briefly.

2 Introduction

The definitions of matchings, perfect matchings and maximum matchings are omitted as they are not relevant to the main topic. For the purpose of convenience, let $n = |V|$, $m = |E|$.

The best-known and also earliest algorithm to solve matching problems on general graphs is the Blossom algorithm of Edmonds [2] . It has $O(nm) = O(n^3)$ running time for the naive implement, and $O(n^{2.5})$ [3] time when optimized.

On the other hand, Lovász [4] provides an randomized algorithm which tests whether a given graph has a perfect matching in $O(n^\omega)$ time. In addition, more properties of a given graph can be found out in $O(n^\omega)$ time, such as the size of its maximum matchings, or even finding a maximum matching.

Theorem 1. $2 \leq \omega < 2.38$.

I'm going to introduce how to construct a perfect or maximum matching of a given graph in $O(n^\omega)$ time in expectation. Besides, after slightly modifying, the problem on bipartite graphs can be solved easily via the LUP factorization algorithm of Hopcroft and Bunch [8] .

My report will be divided into the following parts in order: first basic knowledges of LA, then a simple approach finding a perfect matching via Gaussian elimination, then how to find a perfect matching in $O(n^\omega)$ time, and finally the conclusion and open problems. In addition, part of the proofs will be shown in the appendix.

3 Preliminaries

3.1 Tutte Matrix

Definition (Tutte matrix). *Let $G = (V, E)$ be a undirected graph. The* Tutte Matrix of G is a n-order square matrix $\tilde{A}(G)$ such that

$$\tilde{A}(G)_{ij} = \begin{cases} x_{ij} & i < j,\ (i,j) \in E \\ -x_{ij} & i > j,\ (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Where x_{ij} is a variable, thus there are exactly m variables in $\tilde{A}(G)$. Without ambiguity, $\tilde{A}(G)$ is abbreviated as $\tilde{A}$ below.

Theorem 2 (Tutte [6]). *G has a perfect matching iff* $\det \tilde{A} \neq 0$.

Theorem 3 (Lovász [4]). *The size of the maximum matching of G equals to half of* $\operatorname{rank} \tilde{A}$.

The existing algorithms on matrices are usually for constant matrices. For the purpose of convenience, it is OK to simply choose a number field, for example, the residual system modulo a prime number p, and replace all variables in $\tilde{A}$ by a random number in the field. The matrix obtained by substituting all variables is denoted as $A(G)$ below.

Theorem 4. *The rank of $A(G)$ is at most twice of the size of the maximum matching G, and the equality holds with probability at least $1 - \frac{n}{p}$.* [4]

Usually p is much larger than n in pratice, thus the probability of error is negligible. Besides it is possible to reduce the probability by simplying running the algorithm repeatedly. Obviously, due to this theorem it is possible to test if a given graph has a perfect matching in $O(n^\omega)$ time, and even finding out the size of the maximum matching of the graph.

Next we will focus on approaches to find a perfect matching. Accroding to the theorem before, with high probability $A(G)$ is invertible if G has a perfect matching, and further we have the following theorem:

Theorem 5 (Rabin, Vazirani). *With high probability, $A^{-1}_{j,i} \neq 0$ iff $G - \{v_i, v_j\}$ has a perfect matching.*

In other words, if there are two vertices v_i, v_j such that $(v_i, v_j) \in E$, and $A^{-1}_{j,i} \neq 0$, then there is a perfect matching including (v_i, v_j) with high probability. The edges that appears in some perfect matchings are denoted as *allowed edges* below.

For the purpose of convenience, "with high probability" is omitted without special instructures.

3.2 Maximum Matchings vs. Perfect Matchings

Theorem 6 (Rabin, Vazirani [5] ; Ibarra, Moran [7]). *The problem of finding a maximum matching can be reduced to the problem of finding a perfect matching in randomized time $O(n^\omega)$.*

This theorem is significant as it transforms the problem of finding a maximum matching to the problem of finding a perfect matching, which is much simpler.

Because of the theorem, we will only focus on the problem of finding a perfect matching.

4 Finding out a Perfect Matching

4.1 An Algorithm via Gaussian Elimination

There is an obvious algorithm according to **Theorem 5**: try to find an edge (v_i, v_j) repeatedly, and if $A(G)^{-1}_{j,i} \neq 0$, then delete v_i and v_j from the graph and add (v_i, v_j) into the matching. After that, just calculate $A(G)^{-1}$ again in $O(n^\omega)$ time.

It remains to be an open problem whether $\omega = 2$, but most people believe that $\omega > 2$. Thus the algorithm has $O(n(n^2 + n^\omega)) = O(n^{\omega+1})$ running time.

The bottleneck of this algorithm is re-calculating $A(G)^{-1}$ after deleting two rows and columns from $A(G)$. In fact, it is possible to update $A(G)$ in just $O(n^2)$ time using the following theorem:

Theorem 7 (Elimination Theorem)**.** *Let*

$$A = \begin{bmatrix} a_{1,1} & v^T \\ u & B \end{bmatrix} \quad A^{-1} = \begin{bmatrix} \hat{a}^{1,1} & \hat{v}^T \\ \hat{u} & \hat{B} \end{bmatrix},$$

where $\hat{a}_{1,1} \neq 0$. *Then*

$$B^{-1} = \hat{B} - \frac{\hat{u}\hat{v}^T}{\hat{a}_{1,1}}.$$

The elimination theorem solves the situation to eliminate the first row and column. In fact, the theorem holds true for any single row and column, so we need to calculate $A(G)^{-1}$ only once at the beginning of the algorithm. When deleting v_i and v_j from the graph, we can simply run the $O(n^2)$ elimination for two times, thus getting the new $A(G)^{-1}$.

Theorem 8. *The algorithm above finds a perfect matching in $O(n^3)$ time.*

4.2 Matching Verification

Now we focus on how to reduce the running time to $O(n^\omega)$. The bottleneck part is Gaussian elimination, so we can try to replace the simple $O(n^3)$ elimination with a better one.

First consider a simplified problem as the elimination without pivoting, i.e. it holds true after eliminating the first $i-1$ columns, that $a_{i,i}$ is always nonzero.

There is a technique called lazy elimination, which means that we need not run the $O(n^2)$ modification each step, but do several times of modification together once of a time.

The algorithm can be described as following:

Where UPDATE(R, C) denotes to apply the modification on rows R and columns C. The procedure is equivalent to combine several vectors u_i and v_i

Algorithm 1 Gaussian Elimination without Pivoting

```
1: function SIMPLE-ELIMINATION($A$ : Matrix)
2:     for $i = 1 \to n$ do
3:         lazily eliminate the $i$-th row and $i$-th column of $A$
4:         $j \leftarrow$ the largest integer such that $2^j | i$
5:         UPDATE($\{i+1, \dots, i+2^j\}$, $\{i+1, \dots, n\}$)
6:         UPDATE($\{i+2^j+1, \dots, n\}$, $\{i+1, i+2^j\}$)
```

into matrices U and V, and then calculate UV, thus it can be done in $O(n^\omega)$ time using fast matrix multiplication.

It can be easily shown that this algorithm has the same complexity as the UPDATE procedure, i.e. it has running time $O(n^\omega)$. Actually the algorithm can be considered as a simplified version of the classic fast LU decomposition algorithm [8] . The main purpose to introduce this algorithm is to prove the following theorem:

Theorem 9. *Let G have a perfect matching. Let M be any matching of G, then it is possible to find a maximal subset $M' \subset M$ in $O(n^\omega)$ time, such that M' is the subset of some perfect matching.*

4.3 The Algorithm for Bipartite Graphs

Firstly, the definition of $\tilde{A}$ should be slightly changed for bipartite graphs:

Definition (Tutte matrix on bipartite graphs)**.** *Let $G = (\{U, V\}, E)$ be a bipartite graph. Let $n = |U|$, $m = |V|$, then the Tutte matrix of G is a $n \times m$ matrix such that:*

$$\tilde{A}(G)_{i,j} = \begin{cases} x_{i,j} & (u_i, v_j) \in E \\ 0 & \textit{otherwise} \end{cases}$$

Theorem 10 (Tutte [6])**.** *The size of the maximum matching of G equals to* rank $\tilde{A}(G)$*.*

Similarly, let $A(G)$ denote the matrix obtained by randomly substituting all variables.

The algorithm before only works with no pivoting, thus it can not be used to find a perfect matching of given pipartite graph directly. However, by the help of the classic LU decomposition algorithm [8] , it is still possible to eliminate $A(G)$ in $O(n^\omega)$ time, which solves the problem.

4.4 The Algorithm for General Graphs

4.4.1 Basic Ideas

For general graphs, we need to eliminate two rows and columns in $A(G)^{-1}$ each time a match (v_i, v_j) is found. Therefore, the lazy elimination technique does not work anymore.

A basic idea is to partition G into several subgraphs which are easier to find a perfect matching, and solve the problem for each subgraph recursively.

Next consider how to implement the PARTITION procedure.

Algorithm 2 Finding a Perfect Matching Recursively

```
1: function GENERAL-MATCHING($G$)
2:     Find a match $M$ greedily, such that $|M| \geq \frac{n}{4}$
3:     Call the procedure described in Theorem 9 to find a maximal sub-
       match $M'$
4:     if $|M'| \geq \frac{n}{8}$ then
5:         Let $G'$ be the induced subgraph of the uncovered vertices, and call
           GENERAL-MATCHING($G'$) recursively
6:     else
7:         Call PARTITION($G'$)
```

4.4.2 Canonical Partition

A graph G is called *elementary*, if G has a perfect matching, and all allowed edges of G forms a connected spanning subgraph of G.

Define a relation $\sim$ on set V, such that $u \sim v$ iff either $u = v$, or $G - \{u, v\}$ has no perfect matching.

The following theorem is due to Lovász [4] :

Theorem 11. *If G is elementary, then $\sim$ defined on G is a equivalence relation.*

Let $P(G)$ denote the set of equivalence classes of $\sim$, the so-called *canonical partition* of G.

According to **Theorem 2** (Tutte's Theorem), the canonical partition of G can be easily obtained from $A(G)^{-1}$.

Theorem 12. *Let G be an elementary graph, let $S \in P(G)$ with $|S| \geq 2$, and let C be any connected component of $G - S$. Then*

1. *The graph G'_S obtained by contracting each component in $G - S$ into a single vertex is elementary.*
2. *$\forall v \in V(C), C - \{v\}$ has a perfect matching.*
3. *The graph C' obtained from $G[S \cup V(C)]$ by contracting S into a artificial vertex s is elementary.*
4. *$P(C') = \{\{s\}\} \cup \{T \cap V(C) | T \in P(G)\}$.*

From the theorem, the number of components of $G - S$ equals to $|S|$. Moreover, it holds true for any perfect matching of G, that different vertices in S matches with vertices from different components of $G - S$, since there are no allowed edges between two vertices in S. Therefore, it is obvious that we can construct a perfect matching of G from any perfect matching between S and components of $G - S$.

We can use the algorithm mentioned above to find a perfect matching between S and components of $G - S$. In other words, it is possible to decomposite the problem of perfect matchings into several smaller problems by the help of bipartite matching.

Notice that in the algorithm, we treat the largest component specially. The purpose of that is to reduce the size of sub-problem effectively.

Algorithm 3 The Part Using Canonical Partition

```
1: function PARTITION($G$)
2:     if $G$ is not elementary then
3:         Decomposite $G$ into several elementary subgraphs, then call PARTITION for each subgraph
4:     else
5:         Let $S \in P(G)$ with $|S| = k \geq 2$. Let $C_1, \ldots, C_k$ denotes each connected components $G - S$
6:         Assume that $C_1$ is the largest component, and define $C_1'$ in the same way Theorem 12 describes, then calculate $P(C_1')$ via Theorem 12
7:         if $P(C_1')$ contains a non-trivial class then
8:             Call PARTITION($C_1'$)
9:         else
10:            Call GENERAL-MATCHING($C_1'$)
11:        Let $M_1'$ denotes the matching found, let $c$ denotes the vertex matching to $s$
12:        Choose a vertex $v$ as a neighbor of $c$ in $S$ arbitarily, and find out a perfect matching containing $(v, c)$ between $S$ and components of $G - S$, then add the chosen edges into the global matching
13:        For each $C_i$, remove the vertex matched with some vertex in $S$, then find a perfect matching for the remaining part recursively
```

Lemma. *In **Algorithm 3** , the size of the sub-problem solved recursively is at most $\frac{7}{9}$ of the original problem.*

Actually, the original paper introduced the dynamic graph algorithm [9] to ensure the complexity of the partition. As it is too complicated, and it has little relavance to our main topic, its details are omitted here.

Theorem 13. *The algorithm above finds a perfect matching of G in $O(n^\omega)$ time.*

5 Conclusion

In the original paper, the authors presented their results, i.e. a new algorithm which finds a perfect matching in $O(n^\omega)$ time, and even finds a maximum matching in $O(n^\omega)$ time accroding to **Theorem 6**. In addition, they mentioned that they had found an algorithm to find a maximum matching of a planar graph in $O(n^{\omega/2})$ time, which is presented in their another paper, but it is omitted because of its little relavance to our main topic.

At the end of the paper, the authors raised an open problem: the algorithm for general graphs have a complicated partition step, which is not necessary for bipartite graphs. Is it possible to simplify the algorithm so that the partition part can be removed?

In fact, I have not found relevant surveys or materials to this problem. In my opinion, the answer is probably false, because the vertices in bipartite graphs are naturally divided into two disjoint sets, while these in general graphs are not. Thus the partition step is very likely to be necessary, even if not, there should still be a similar procedure to replace it.

6 References

[1] M. Mucha and P. Sankowski. *Maximum matchings via Gaussian elimination*, 2004.

[2] J. Edmonds. *Paths, trees and flowers*, 1965.

[3] S. Micali and V. V. Vazirani. *An $O(|V||E|)$ algorithm for finding maximum matching in general graphs*, 1980.

[4] L. Lovász. *On determinants, matchings and random algorithms*, 1979.

[5] M. O. Rabin and V. V. Vazirani. *Maximum matchings in general graphs through randomization*, 1989.

[6] W. T. Tutte. *The factorization of linear graphs*, 1947.

[7] O. H. Ibarra and S. Moran. *Deterministic and probabilistic algorithms for maximum bipartite matching via fast matrix multiplication*, 1981.

[8] J. Bunch and J. Hopcroft. *Triangular factorization and inversion by fast matrix multiplication*, 1974.

[9] J. Holm, K. de Lichtenberg, and M. Thorup. *Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity*, 2001.

7 Appendix : Part of the Proofs

Theorem 2. (Tutte) *G has a perfect matching iff* $\det \tilde{A} \neq 0$.

Proof. Consider the definition of the determinant of a matrix:

$$\det \tilde{A} = \sum_{\pi} (-1)^{\mathrm{sgn}(\pi)} \prod_{k} \tilde{A}_{k,\pi(k)}$$

For any non-zero item on the right of the equation, the corresponding permutation π always satisfies that $\forall k, (k, \pi_k) \in E$. Let us denote the inverse permutation as π^{-1}, then the item corresponding to it is also non-zero.

If π has a rotation of odd order, then the corresponding item of π and π^{-1} are certainly opposite. Thus only these permutations without rotations of odd order will be added to the determinant.

If the determinant is non-zero, it means that G has a even-cycle covering, thus G has a perfect matching, and vice versa.

Q.E.D.

Theorem 7. (Elimination Theorem) *Let*

$$A = \begin{bmatrix} a_{1,1} & v^T \\ u & B \end{bmatrix} \quad A^{-1} = \begin{bmatrix} \hat{a}^{1,1} & \hat{v}^T \\ \hat{u} & \hat{B} \end{bmatrix},$$

where $\hat{a}_{1,1} \neq 0$. Then

$$B^{-1} = \hat{B} - \frac{\hat{u}\hat{v}^T}{\hat{a}_{1,1}}.$$

Proof. Since $AA^{-1} = I_n$, we have

$$\begin{bmatrix} a_{1,1}\hat{a}_{1,1} + v^T u & a_{1,1}\hat{v}^T + v^T\hat{B} \\ u\hat{a}_{1,1} + B\hat{u} & u\hat{v}^T + B\hat{B} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & I_{n-1} \end{bmatrix}$$

Further we get

$$B(\hat{B} - \frac{\hat{u}\hat{v}^T}{\hat{a}_{1,1}}) = I_{n-1} - u\hat{v}^T - \frac{B\hat{u}\hat{v}^T}{\hat{a}_{1,1}}$$

$$= I_{n-1} - u\hat{v}^T + \frac{u\hat{a}_{1,1}\hat{v}^T}{\hat{a}_{1,1}} = I_{n-1} - u\hat{v}^T + u\hat{v}^T = I_{n-1}$$

Q.E.D.

Theorem 9. *Let G have a perfect matching. Let M be any matching of G, then it is possible to find a maximal subset $M' \subset M$ in $O(n^\omega)$ time, such that M' is the subset of some perfect matching.*

Proof. Without loss of generality, let $M = \{(v_1, v_2), (v_3, v_4), \ldots, (v_{k-1}, v_k)\}$, and the uncovered vertices are $v_{k+1}, \ldots, v_n$. Permute the columns of $A(G)^{-1}$ in the order of $v_2, v_1, v_4, v_3, \ldots, v_{n-1}, v_n$, while the order of rows remains unchanged.

When running **Algorithm 1** , we can simply skip to the next column if $a_{i,i} = 0$. Then the rows and columns eliminated during the process just correspond to M'.

Q.E.D.

lemma. *In **Algorithm 3** , the size of the sub-problem solved recursively is at most $\frac{7}{9}$ of the original problem.*

Proof. Since C_1 is the largest component, we only need to prove that $|V(C_1')|$ is at most $\frac{7}{9}n$.

If Partition(G) is called by General-Matching(G), then there is a match $\hat{M}$ of G containing only unallowed edges, and spanning at least $\frac{n}{3}$ vertices. Let $\hat{V}$ denote the set of these vertices.

Since the edges in $\hat{M}$ can never go across the partition, C_1' contains no vertices in $\hat{V}$ when the partition stops. Notice that the largest component loses at lease 3 vertices, and gain exactly one *artificial* vertex. Thus the number of vertices in C_1' when the partition stops is at most $n - \frac{2}{3}|\hat{V}| \leq n - \frac{2}{9}n = \frac{7}{9}n$.

Q.E.D.