RECAP — How to find a maximum matching?

First characterize maximum matchings

A maximal matching cannot be enlarged by adding another edge.

A maximum matching of G is one of maximum size.

Example. Maximum $\neq$ Maximal

Let M be a matching. A path that alternates between edges in M and edges not in M is called an M-alternating path.

An M-alternating path whose endpoints are unsaturated by M is called an M-augmenting path.

Theorem(Berge, 1957) A matching M is a maximum matching of graph G iff G has no M-augmenting path.

RECAP — Combinatorial approach

Augmenting Path Algorithm

Input graph G on n vertices
Output matching $M \subseteq E(G)$ of maximum size

$M := \emptyset$
WHILE there exists an M-augmenting path P
    augment M along P
output M

Problem: How to find an augmenting path fast?

Easier in bipartite graphs:

Naive approach: $O(mn)$

Hopcroft-Karp: $O(m\sqrt{n})$

Tougher for general graphs:

Edmonds' Blossom Algorithm* (1965): $O(n^2 m)$

*In his paper "Paths, Trees, and Flowers" Edmonds defined the notion of polynomial time algorithm

History of maximum matching algorithms

Authors	Year	Order of Running Time
Edmonds	1965	$n^2 m$
Even-Kariv	1975	$\min\{\sqrt{n}m \log n, n^{2.5}\}$
Micali-Vazirani	1980	$\sqrt{n}m$
Rabin-Vazirani	1989	$n^{\omega+1}$
Mucha-Sankowski	2004	n^{ω}
Harvey	2006	n^{ω}

$\omega := \inf\{c :$ two $n \times n$ matrices can be multiplied in time $O(n^c)\}$

"time" is actually the number of arithmetic operations

The determinant, the inverse, or a submatrix of maximum rank of an $n \times n$ matrix can also be found in time $O(n^{\omega})$.

Clear: $\omega \geq 2$

Naive algorithm: $\omega \leq 3$

Theorem (Coppersmith-Winograd, 1990) $\omega < 2.38$

RECAP — Algebraic approach

First question: **Is there a perfect matching in G?**

First let G be bipartite
with parts $U = \{u_1, \ldots, u_n\}$, $W = \{w_1, \ldots, w_n\}$.

Let B be the southwest $n \times n$ submatrix of the adjacency matrix of G:

$$b_{ij} := \begin{cases} 1 & \text{if } u_i w_j \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

The permanent of B is

$$\texttt{per}B := \sum_{\pi \in S_n} b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{n,\pi(n)}$$

Claim M has a perfect matching iff $\texttt{per}(B) \neq 0$

Problem: permanent is hard to compute

Determinant is similar and easy to compute

$$\mathtt{det}B := \sum_{\pi \in S_n} (-1)^{\mathrm{sgn}(\pi)} b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{n,\pi(n)}$$

Problem: $\mathtt{det}(B)$ could be 0 even if $\mathtt{per}(B) \neq 0$.

Solution: Introduce one variable x_{ij} for each edge $u_i w_j \in G$, $u_i \in U$, $w_j \in W$ and define a matrix A:

$$a_{ij} := \begin{cases} x_{ij} & \text{if } u_i w_j \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

Claim M has a perfect matching iff $\mathtt{det}(A) \not\equiv 0$

Problem: Exponentially many terms.

Solution: Substitution and then determinant calculation takes only $O(n^\omega)$.

How to ensure that “nonzero-ness” is preserved?
Choose a prime p, $2n \leq p \leq 4n$, work over $\mathbb{F}_p$.
Substitute randomly (Schwartz-Zippel Lemma)

Claim $\mathtt{det}(A) \not\equiv 0 \;\Rightarrow\; \mathtt{Prob}[\mathtt{det}(A) \neq 0] > \frac{1}{2}$

RECAP — Schwartz-Zippel Lemma

Let $q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be nonzero polynomial of degree $d \geq 0$, and let $S \subseteq \mathbb{F}$ be a finite set. Then the number of n-tuples $(r_1, \ldots, r_n) \in S^n$ with $q(r_1, \ldots, r_n) = 0$ is at most $d|S|^{n-1}$. In particular, if $r_1, \ldots, r_n \in S$ is chosen independently and uniformly at random, then

$$\mathtt{Pr}[q(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|}$$

General remark: Correctness proofs proceed in $\mathbb{Z}(x_1, \ldots, x_n)$ arithmetic.

Randomization proofs, i.e., that the probability of an incorrect answer is small, depends on selecting a large enough prime p to substitute randomly over $\mathbb{F}_p$.
If the algorithm performs t zero-tests of polynomials of degree at most d, then selecting $p \geq 2td$ gives that the success probability is at least $\frac{1}{2}$.
In the previous perfect matching test algorithm for bipartite graphs there was $t = 1$ zero-test of a polynomial of degree n (the determinant).

RECAP — Algebraic approach

Let now $G = (V, E)$ be an arbitrary graph.

Define the Tutte matrix $T(G) = T$ of G

$$t_{ij} := \begin{cases} x_{ij} & \text{if } v_i v_j \in E(G) \text{ snd } i < j \\ -x_{ij} & \text{if } v_i v_j \in E(G) \text{ snd } i > j \\ 0 & \text{otherwise} \end{cases}$$

Theorem (Tutte)
G has a perfect matching iff $\mathtt{det}(T) \not\equiv 0$

Then again: random substitution and evaluation of the determinant gives a randomized algorithm to check whether G has a perfect matching.

How to find a perfect matching?

A first try

Input graph G containing a perfect matching
Output perfect matching $M \subseteq E(G)$

$E(G) = \{e_1, \ldots, e_m\}$
$M := G$, $i := 0$
WHILE $i < m$ DO $i := i + 1$
    IF $\det T(M - e_i) \neq 0$ THEN $M := M - e_i$
output M

Running time: $O(mn^\omega)$

Rabin-Vazirani

Edge $e \in G$ is allowed if it is contained in a perfect matching.

Let $N = T^{-1}$ be the inverse Tutte matrix.

Lemma (Rabin-Vazirani)
Assume that G has a perfect matching.
Then edge $e = ij \in E(G)$ is allowed $\Leftrightarrow$ $N_{i,j} \neq 0$

Proof. $e = ij$ is allowed $\Leftrightarrow$ $G - \{i, j\}$ has a perfect matching $\Leftrightarrow$ $\det T_{del(\{i,j\},\{i,j\})} \neq 0$ By Fact 1 and Fact 0, we have

$$\begin{aligned} \det T_{del(\{i,j\},\{i,j\})} &= \pm \det T \cdot \det N_{\{i,j\},\{i,j\}} \\ &= \pm \det T \cdot (N_{i,j})^2 \end{aligned}$$

Definitions and Facts from Linear Algebra

$n \times n$ matrix M; $S \subseteq [n]$

submatrix containing rows and colums of S: $M[S]$
ith column (row) denoted by $M_{*,i}$ ($M_{i,*}$)
when colum set S and row set T is deleted: $M_{del(S,T)}$

M is non-singular if det $M \neq 0$.

The inverse M^{-1} of M is given by

$$(M^{-1})_{i,j} = (-1)^{i+j} \cdot \frac{\det M_{del(j,i)}}{\det M}.$$

M is skew-symmetric if $M = -M^T$.

Remark M is skew-symmetric $\Rightarrow$ M is square, all diagonal entries are 0.

Fact 0. M is skew-symmetric, non-singular
$\Rightarrow M^{-1}$ is skew-symmetric

One more fact from Linear Algebra

Let $M = \begin{pmatrix} W & X \\ Y & Z \end{pmatrix}$, where Z is square

If M is non-singular, let $M^{-1} = \begin{pmatrix} \widehat{W} & \widehat{X} \\ \widehat{Y} & \widehat{Z} \end{pmatrix}$

Fact 1. (Jacobi's Determinant Identity)

$$\det Z = \pm \det M \cdot \det \widehat{W}.$$

Proof of Fact 1.

$$\begin{pmatrix} W & X \\ Y & Z \end{pmatrix} \cdot \begin{pmatrix} \widehat{W} & 0 \\ \widehat{Y} & I \end{pmatrix} = \begin{pmatrix} I & X \\ 0 & Z \end{pmatrix}$$

The Algorithm

Rabin-Vazirani Algorithm

Input graph G containing a perfect matching
Output perfect matching $M \subseteq E(G)$

$H := G$, $M := \emptyset$
WHILE $|M| < n/2$ DO
  compute H^{-1}
  find $ij \in E(H)$ with $\left(H^{-1}\right)_{i,j} \neq 0$
  $M := M \cup \{ij\}$
  $H := H - \{i, j\}$
output M

Running time: $O(n^{\omega+1})$

Question: Do we really have to calculate the inverse always from scratch?

Rank-1 update

M $n \times n$ matrix
$u, v \in \mathbb{F}^n$ (column) vectors
$c \in \mathbb{F}$ scalar

Then $\tilde{M} = M + cuv^T$ is a rank-1 update of M.

Fact 3. W is non-singular $\Leftrightarrow$ $\hat{Z}$ is nonsingular. Also,

$$W^{-1} = \hat{W} - \hat{X}\hat{Z}^{-1}\hat{Y}$$

Proof. First part follows from Fact 1.

$$\begin{aligned}
& \begin{pmatrix} (W - XZ^{-1}Y)^{-1} & 0 \\ Z^{-1}Y(W - XZ^{-1}Y)^{-1} & I \end{pmatrix} \\
= & \begin{pmatrix} \hat{W} & \hat{X} \\ \hat{Y} & \hat{Z} \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ 0 & Z \end{pmatrix} \cdot \begin{pmatrix} I & X \\ 0 & I \end{pmatrix} \\
= & \begin{pmatrix} \hat{W} & \hat{W}X + \hat{X}Z \\ \hat{Y} & \hat{Y}X + \hat{Z}Z \end{pmatrix}
\end{aligned}$$

Speed-up via rank-1 updates

Rabin-Vazirani Algorithm with rank 1-updates
(Mucha-Sankowski)

Input graph G containing a perfect matching
Output perfect matching $M \subseteq E(G)$

$M := \emptyset$
compute $N = T^{-1}$
WHILE $|M| < n/2$ DO
  find $ij \in E(G)$ with $N_{i,j} \neq 0$
  $M := M \cup \{ij\}$
  $N := N - \frac{1}{N_{i,j}} N_{*,j} N_{i,*} + \frac{1}{N_{i,j}} N_{*,i} N_{j,*}$
output M

Correctness: After an update of N:
1. in the $i$the and jth columns all entries are 0.
2. By Fact 3, $N[V \setminus V(M)]$ is the inverse of the Tutte matrix of $G - V(M)$.

Running time: $O(n^3)$

Harvey's divide-and-conquer implementation

FindPerfectMatching(G)
Input graph G containing a perfect matching
Output perfect matching $M \subseteq E(G)$

compute $N = T^{-1}$
output `BuildMatching`$(V(G), N)$

BuildMatching(S, N, α)
Input subset $S \subseteq V(G)$; integer α;
matrix N with $N[S]$ up-to-date;
Output perfect matching $M \subseteq E(G)$

$M := \emptyset$
IF $|S| > 2$ THEN
  partition $S = S_1 \cup \cdots \cup S_\alpha$, $|S_1| = \cdots = |S_\alpha|$
  FOR each $1 \leq a < b \leq \alpha$ DO
    `BuildMatching`$(S_a \cup S_b, N, \alpha)$
    Update N
ELSE ($|S| = 2$)
  IF $T_{i,j} \neq 0$ and $N_{i,j} \neq 0$ THEN
    $M := M \cup \{ij\}$
    Update N
output M

Correctness and Recursion

Correctness: implementation of Rabin-Vazirani; every edge is considered at least once

$h(s)$: running time of `BuildMatching` for $|S| = s$

Assuming that the "Update" lines can be performed in time $O(s^\omega)$ for a subproblem of size $|S| = s$, we have the recursion

$$h(s) \leq \binom{\alpha}{2} h\left(\frac{s}{\alpha/2}\right) + O\left(\binom{\alpha}{2} s^\omega\right)$$

$h(n) = O(n^\omega)$ provided $\log_{\alpha/2} \binom{\alpha}{2} < \omega$

For $\omega = 2.38$, $\alpha = 13$ will do

Efficient updates

A little bit technical...

Idea: At the end of each recursive subproblem do **not** update the full matrix, **only** the part belonging to the parent subproblem

It turns out: for a subproblem of size s, this can be done with a **constant** number of matrix multiplications and inversions of $O(s) \times O(s)$ matrices

Remark: How to generalize all these algorithms finding a perfect matching to find a maximum matching? First, in time $O(n^\omega)$ find a maximum rank submatrix of T. For a skew-symmetric matrix this could be chosen to be a principal submatrix. Then find a perfect matching in the subgraph corresponding to this full rank principal submatrix.