基于线性代数的一般图匹配

中山市中山纪念中学 杨家齐

摘 要

图的匹配问题是图论中的经典问题，无论在学术界还是在算法竞赛界都有着重要的地位。本文主要介绍了一种基于线性代数的一般图匹配算法，并给出了它的一些简单应用。这个算法与传统的解决这一问题的组合方法有很大不同，希望这个算法本身以及在得到这个算法的过程中所用到的思想能够给读者带来启发。

引言

图的匹配问题按照图的性质可以分为二分图匹配和一般图匹配两类。其中二分图匹配已经在算法竞赛界中出现很多年了，是算法竞赛的一个常见考点。由于它的算法相对简单，因此它得到了很好的普及：如今大部分选手都已经掌握、并且能够熟练应用二分图匹配的算法。

而一般图匹配则是一个新兴的考点，近几年的集训队作业、冬令营考试以及集训队互测中都有它的身影。但这些比赛都是 NOI 级别以上的，在 NOI 以下的比赛中罕见考察一般图匹配的题目，并且目前能够掌握一般图匹配算法的选手也相对较少。笔者认为，一般图匹配算法普及度不够的一个重要原因是目前解决这类问题常见的带花树算法较为复杂，学习门槛较高。因此本文将要介绍一个便于记忆，易于实现的匹配算法，希望这个算法能够促进一般图匹配算法的普及。

正如本文的题目所述，本文将要介绍的算法是一个代数方法。尽管在算法竞赛中组合方法更为常见，但代数方法同样是解决图论问题的强有力工具。这其中最广为人知的例子恐怕就是生成树计数的 Matrix-Tree 定理了。代数方法是解决问题的一个全新的视角，希望本文能够让读者感受到代数方法在解决问题中的独特魅力。

本文将首先从完美匹配问题入手，解决它对应的判定性问题，然后得到一个构造完美匹配方案的算法，并通过一些线性代数知识来逐步优化它，接着将最大匹配转化为完美匹配问题解决，最后再叙述这一算法的一些应用与拓展。

1 预备知识

1.1 图论

若无特殊说明，文中涉及的图均指无向图。

我们用 $G=(V,E)$ 来表示一个图，其中 V 是点集，E 是边集。我们一般用 n 来表示点集的大小 $|V|$，用 m 来表示边集的大小 $|E|$，并且我们一般认为 $V=\{v_1,v_2,\ldots,v_n\}$。

在不致产生歧义的情况下，我们用 uv 表示边 (u,v)。

图 $G=(V,E)$ 的**匹配** M 是指边集 E 的一个子集，满足 M 中任意两条边都没有公共点。如果结点 v 是 M 中某条边的端点，那么我们称 v 在匹配 M 中。图 G 的所有匹配的大小的最大值记作 $\nu(G)$，并且如果 $|M|=\nu(G)$，那么我们称 M 是图 G 的一个**最大匹配**。特别地，如果图 G 中的每个点都在匹配 M 中，那么我们称 M 是图 G 的一个**完美匹配**。显然，只有当 $|V|$ 为偶数时图 G 才可能有完美匹配。

对于一个图 $G=(V,E)$ 以及一个点集 $U\subseteq V$，我们把从图 G 中删去 U 中所有点得到的图记作 $G-U$，即 $G-U=(V\backslash U,E\cap\{uv|u,v\in V\backslash U\})$；把图 G 关于 U 的导出子图记作 $G[U]$，即 $G[U]=G-(V\backslash U)$。

1.2 线性代数

对于一个 m 行 n 列的矩阵 A，我们设它的行号集合为 $\{1,\ldots,m\}$，列号集合为 $\{1,\ldots,n\}$。矩阵 A 第 i 行第 j 列的元素记作 $A_{i,j}$。

对于一个 $m\times n$ 的矩阵 A 以及任意 $I\subseteq\{1,\ldots,m\},J\subseteq\{1,\ldots,n\}$，我们记从 A 中保留 I 中所有行以及 J 中所有列得到的子矩阵为 $A_{I,J}$。我们将 $A_{I,\{1,\ldots,n\}}$ 简记为 $A_{I,\cdot}$，将 $A_{\{1,\ldots,m\},J}$ 简记为 $A_{\cdot,J}$。

类似地，我们记 $A^{I,J}$ 为从 A 中删去 I 中所有行及 J 中所有列得到的子矩阵。

对于一个 $n\times n$ 的方阵 A，我们将它的**行列式**记作 $\det A$，并且

$$\det A=\sum_{\pi\in\Sigma_n}(-1)^{\operatorname{sgn}\pi}\prod_{k=1}^{n}A_{k,\pi(k)}$$

其中 Σ_n 表示 $\{1,\ldots,n\}$ 的所有置换构成的集合，$\operatorname{sgn}\pi$ 表示置换 π 的符号。如果序列 $\{\pi(1),\ldots,\pi(n)\}$ 的逆序对数为 x，那么 $\operatorname{sgn}\pi=(-1)^x$。

给定 m 个 n 维向量 $\{v_1,\ldots,v_m\}$，如果存在 $\lambda_1,\lambda_2,\ldots,\lambda_m$，使得 $v=\sum_{i=1}^{m}\lambda_i v_i$，则称 v 是 $\{v_1,\ldots,v_m\}$ 的**线性组合**。如果这 m 个向量中的任意一个都不能够表示为其它向量的线性组合，那么我们称这 m 个向量**线性无关**，否则称为**线性相关**。

对于一个矩阵 A，如果我们将它的每一行看作一个向量，那么从中选出线性无关向量的极大数目称为行秩；如果我们将每一列看作一个向量，得到的极大线性无关向量数目称为列秩。矩阵的行秩和列秩总是相等的，因此统称为矩阵 A 的**秩**，记作 $\operatorname{rank}A$。

最后，我们不加证明地给出一个结论

定理 1.1. *以下四个问题的复杂度相等*

1. *计算矩阵的行列式；*
2. *将两个矩阵相乘；*
3. *求矩阵的逆；*
4. *对矩阵做高斯消元（LU 分解）。*

我们设这四者的复杂度都是 $O(n^\omega)$[4]。

2 完美匹配的存在性

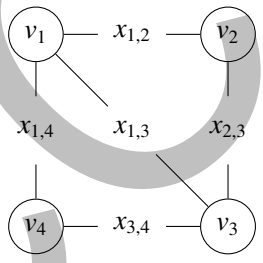
2.1 Tutte 定理

定义 2.1 (Tutte 矩阵)**.** 对于一个无向图 $G = (V, E)$, 定义 G 的 Tutte 矩阵为一个 $n \times n$ 的矩阵 $\tilde{A}(G)$, 其中

$$\tilde{A}(G)_{i,j} = \begin{cases} x_{i,j} & \text{若 } v_i v_j \in E \text{ 并且 } i < j \\ -x_{j,i} & \text{若 } v_i v_j \in E \text{ 并且 } i > j \\ 0 & \text{其它情况} \end{cases}$$

其中，$x_{i,j}$ 是一个与边 $v_i v_j$ 相关联的独一无二的变量。

下面我们来看一个例子。



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

图 1: 图 G 与它的 Tutte 矩阵

图 1 的左半部分给出了一个图 G，右半部分给出了图 G 所对应的 Tutte 矩阵 $\tilde{A}(G)$。可以看到，图 G 的每条边都对应着一个变量，因此 $\tilde{A}(G)$ 中总共会用到 $|E|$ 个变量。

[4]学术界一般认为 $2 \le \omega < 2.373$。

2.2 Tutte 定理

Tutte 矩阵与图的完美匹配的存在性之间有着密切的连系。事实上，我们有如下定理

定理 2.1 (Tutte)**.** 图 G 有完美匹配当且仅当 $\det \tilde{A}(G) \neq 0$。

为了证明这个定理，我们首先需要用“偶环覆盖”的概念来刻画一个具有完美匹配的图。

定义 2.2. 图 $G = (V, E)$ 的一个环覆盖是指用若干个环去覆盖图 G 的所有结点，使得图 G 的任意一个结点恰好在一个环中。

形式化地说，图 G 的每个环覆盖对应于 1 到 n 的一个排列 π，满足 $v_i v_{\pi(i)} \in E$。

如果这个环覆盖中的所有环的长度都是偶数，那么我们称这是一个偶环覆盖，否则我们称它是一个奇环覆盖。

注意，定义 2.2 并没有要求 $v_i v_{\pi(i)}$ 两两不同，也就是说环覆盖中可以出现重复的边。

引理 2.2. 图 $G = (V, E)$ 有完美匹配当且仅当图 G 有一个偶环覆盖。

证明. 若 G 有一个完美匹配，那么我们直接用匹配中每一对点构成的二元环就可以覆盖图 G 了。

若 G 有一个偶环覆盖，那么我们在每一个偶环中隔一条边取一条边，就能够得到 G 的一个完美匹配了。 □

下图 2 将说明如何从图 G 的一个偶环覆盖得到一个完美匹配。

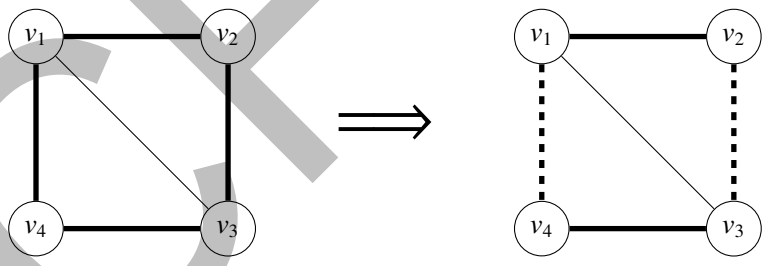


图 2: 偶环覆盖与完美匹配

有了引理 2.2，我们就可以证明 Tutte 定理（定理2.1）了。

证明(Tutte 定理). 我们有

$$\det \tilde{A}(G) = \sum_{\pi \in \Sigma_n} (-1)^{\operatorname{sgn}(\pi)} \prod_{k=1}^{n} \tilde{A}(G)_{k,\pi(k)} \tag{2.1}$$

等式右边的非 0 项形如 $(-1)^{\operatorname{sgn}(\pi)} \prod_{k=1}^{n} \pm x_{k,\pi(k)}$，其中 $v_1 v_{\pi(1)}, \ldots, v_n v_{\pi(n)}$ 是 G 中的边。这些边构成了图 G 的一个环覆盖（与 π 的轮换分解相对应）。

接下来，如果我们将 π 中任意一个长度大于 2 的环反向，那么我们会得到一个相同的环覆盖。现在我们来观察它的符号将如何变化。考虑一个通过将 π 中的某个环反向后得到的置换 π'。将单个环反向并没有改变置换 π 的符号（即 $\operatorname{sgn}(\pi) = \operatorname{sgn}(\pi')$），但改变了环中所有变量 $x_{i,\pi(i)}$ 的符号。

因此，如果我们将 π 中的一个偶环反向，那么得到的项符号与 π 所对应的项是相同的，否则是相反的。

于是所有存在奇环的环覆盖都会被一正一负相抵消掉。由于 $\det \tilde{A}(G) \neq 0$，这就意味着图 G 存在一个偶环覆盖，因此图 G 有完美匹配。 □

为了更加直观地理解这个证明，我们下面以一个三元环为例，来说明为什么奇环覆盖不会出现在 $\det \tilde{A}(G)$ 中。

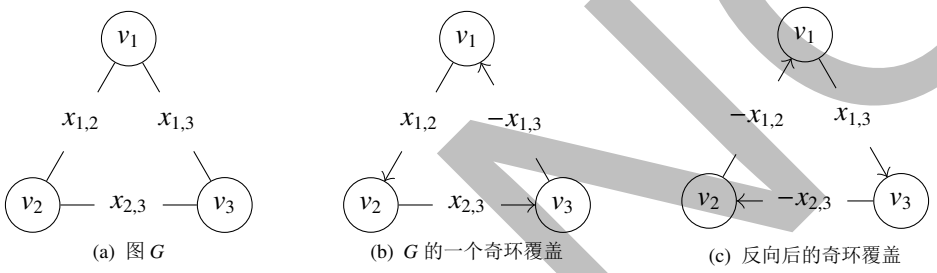


图 3: 三元环 G

我们取图 G 的一个奇环覆盖来进行分析。如图 3(b)，不妨假设我们取的环覆盖所对应的排列为 $\pi = (2\ 3\ 1)$，那么这一项对 $\det \tilde{A}(G)$ 的贡献为 $\operatorname{sgn}(\pi) x_{1,2} x_{2,3} (-x_{1,3})$。现在我们将 π 反向，那么我们将会得到图 3(c)，它对应的排列为 $\pi' = \pi^{-1} = (3\ 1\ 2)$，它对 $\det \tilde{A}(G)$ 的贡献为 $\operatorname{sgn}(\pi')(-x_{1,2})(-x_{2,3})x_{1,3}$。

显然 $\operatorname{sgn}(\pi) = \operatorname{sgn}(\pi')$，并且我们可以看到，$\pi'$ 的贡献是在 π 的贡献的基础上将环上每条边对应的变量都乘以 -1 得到的。因此这两项的符号恰好相反，它们的总贡献是 0。

2.3 随机化

Tutte 定理（定理 2.1）给了我们一个非常好的判定图 G 是否存在完美匹配的算法：我们只需要判断 $\det \tilde{A}(G)$ 是否为零就可以了。但这个做法有一个显著的问题：由于 Tutte 矩阵的每一项都是一个变量，并且变量的总数高达 $|E|$ 个，因此直接计算行列式是非常困难的，所需要的时间甚至是指数级的。

然而，我们并不一定需要计算出 $\det \tilde{A}(G)$，我们所需要做的仅仅是判断 $\det \tilde{A}(G)$ 是否**恒为 0**。

判定一个多项式是否恒为 0 是一个经典问题。考虑我们现在有一个 n 元多项式 $P(x_1, x_2, \ldots, x_n)$，现在我们想判定 $P(x_1, x_2, \ldots, x_n)$ 是否恒为 0，我们该怎么做。

我们首先可以观察到，如果 $P(x_1, x_2, \ldots, x_n) = 0$，那么不论我们选取什么值代入多项式 P 中，得到的结果都一定是 0。那么我们是否能够选取 n 个随机数 $r_1, r_2, \ldots, r_n$，然后通过 $P(r_1, r_2, \ldots, r_n)$ 是否为 0 来推断出 $P(x_1, x_2, \ldots, x_n)$ 是否恒为 0 呢？下面将要叙述的 Schwartz-Zippel 引理将会告诉我们，这个看起来不太靠谱的做法实际上**有很大概率**能得到正确的结果。

引理 2.3 (Schwartz, Zippel)**.** *对于域 $\mathbb{F}$ 上的一个不恒为 0 的 n 元 d 度多项式 $P(x_1, x_2, \ldots, x_n)$，设 $r_1, r_2, \ldots, r_n$ 为 n 个 $\mathbb{F}$ 中的独立选取的随机数，则*

$$\Pr[P(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|\mathbb{F}|}$$

证明. 我们尝试归纳证明这个定理。

考虑单变量的情况：此时 P 至多有 d 个根，因此我们恰好选到其中一个的概率不超过 $\frac{d}{|\mathbb{F}|}$。

现在假设定理对于 $(n-1)$ 的情况成立，我们考虑将 P 中的所有项按照 x_1 的次数分类，设

$$P(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d} x_1^i P_i(x_2, \ldots, x_n)$$

由于 $P \not\equiv 0$，因此存在一个 i 使得 $P_i \not\equiv 0$，我们考虑所有满足条件的 i 的最大者，则 $\deg P_i \leq d - i$。

由归纳假设，$\Pr[P_i(r_2, \ldots, r_n) = 0] \leq \frac{d-i}{|\mathbb{F}|}$。若 $P_i(r_2, \ldots, r_n) \neq 0$，那么 $P(x_1, r_2, \ldots, r_n)$ 为 i 次单项式，因此

$$\Pr[P(r_1, r_2, \ldots, r_n) = 0 | P_i(r_2, \ldots, r_n) \neq 0] \leq \frac{i}{|\mathbb{F}|}$$

设事件 $P(r_1, r_2, \ldots, r_n) = 0$ 为 A，$P_i(r_2, \ldots, r_n) = 0$ 为 B，那么

$$\begin{aligned}
\Pr[A] &= \Pr[A \cap B] + \Pr[A \cap B^c] \\
&= \Pr[B] \Pr[A|B] + \Pr[B^c] \Pr[A|B^c] \\
&\leq \Pr[B] + \Pr[A|B^c] \\
&\leq \frac{d-i}{|\mathbb{F}|} + \frac{i}{|\mathbb{F}|} = \frac{d}{|\mathbb{F}|}
\end{aligned}$$

□

我们知道，对于一个给定的图 G，多项式 $\det \tilde{A}(G)$ 的度是 $O(n)$ 级别的。因此我们不妨选取一个 n^2 级别的质数 p，并在模 p 域 $\mathbb{F}_p$ 下为每条边 $v_i v_j$ 所对应的变量 $x_{i,j}$ 随机赋一个

值，然后通过计算[5] $\det \tilde{A}(G)$ 是否为 0 来判断图 G 是否存在完美匹配。由 Schwartz-Zippel 引理，我们判断错误的概率不超过 $\frac{n}{p} = O(\frac{1}{n})$，这个概率对于我们来说是可以接受的。并且我们可以通过调整 p 的大小，或者多次随机来降低错误率。

从而，我们有如下判定算法

算法 1 Lowasz's testing algorithm

```
if det Ã(G) ≠ 0 then
    print "YES"
else
    print "NO"
end if=0
```

定理 2.4. *算法 1 的时间复杂度为* $O(n^\omega)$。

3 构造完美匹配方案

3.1 一个暴力算法

根据算法 1，我们现在已经能够判定图 G 是否存在完美匹配了。那么我们可以立刻得到一个构造匹配的暴力算法：枚举每条边，删去它的两个端点，并判断剩下的图是否存在完美匹配。

算法 2 A naive matching algorithm

```
M ← ∅
for i ← 1 to n do
    for j ← 1 to n do
        if v_i v_j ∈ E(G) and det Ã(G − {v_i, v_j}) ≠ 0 then
            G ← G − {v_i, v_j}
            M ← M ∪ {v_i v_j}
        end if
    end for
end for
return M =0
```

算法 2 需要进行 $O(n^2)$ 次循环，每次循环需要计算一个 $O(n)$ 阶的方阵的行列式，因此

[5]从此，当我们需要计算 $\tilde{A}(G)$ 的行列式、逆矩阵，或是需要对 $\tilde{A}(G)$ 消元时，我们指的是将变量随机赋值后在 $\mathbb{F}_p$ 下进行相应的操作。

定理 3.1. *算法 2 的时间复杂度为* $O(n^{\omega+2})$。

3.2 一些线性代数知识

算法 2 的瓶颈在于需要判断每条边是否在完美匹配中。那么如果我们能加速这一判断过程，就可以降低这一算法的复杂度了。

为了达到这一目的，我们需要先了解一些线性代数的知识。

定义 3.1. 矩阵 A 关于第 i 行和第 j 列的余子式（记做 $M_{i,j}$）为 $\det A^{i,j}$。

定义 3.2. 矩阵 A 关于第 i 行和第 j 列的代数余子式（记做 $C_{i,j}$）为 $(-1)^{i+j}M_{i,j}$。

矩阵 A 的余子矩阵是一个 $n \times n$ 的矩阵 C，满足其第 i 行第 j 列的元素是 A 关于第 i 行和第 j 列的代数余子式。

定义 3.3. 矩阵 A 的伴随矩阵 $\operatorname{adj} A$ 是 A 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$。

定理 3.2 (拉普拉斯展开)**.** *设矩阵* A *为一个* $n \times n$ *的矩阵，那么对于任意一行* $i \in \{1, \ldots, n\}$*，有*

$$\begin{aligned}\det A &= \sum_{j=1}^{n} A_{i,j}C_{i,j} \\ &= \sum_{j=1}^{n} A_{i,j}(\operatorname{adj} A)_{j,i}\end{aligned}$$

类似地，对于任意一列 $j \in \{1, \ldots, n\}$*，有*

$$\det A = \sum_{i=1}^{n} A_{i,j}(\operatorname{adj} A)_{j,i}$$

定理 3.3. *如果矩阵* A *可逆，那么我们有*

$$A^{-1} = \operatorname{adj} A / \det A$$

证明. 由定义 3.2 及定义 3.3，$A \times \operatorname{adj} A$ 的第 i 行第 i 列的系数是 $\sum_{j=1}^{n} A_{i,j}C_{i,j}$。根据定理 3.2，这个系数等于 $\det A$。

如果 $i \neq j$, 那么 $A \times \operatorname{adj} A$ 的第 i 行第 j 列的系数是 $\sum_{k=1}^{n} A_{i,k}C_{j,k}$。实际上，这就相当于把 A 的第 j 行元素换成第 i 行元素后求行列式。由于有两行相同，因此行列式为 0。

于是我们可以得到

$$A \times \operatorname{adj} A = \operatorname{adj} A \times A = \det A \times I_n$$

即

$$A^{-1} = \operatorname{adj} A / \det A$$

□

3.3 斜对称矩阵的性质

我们先定义斜对称矩阵的概念。

定义 3.4. 对于一个 $n \times n$ 的矩阵 A，如果 $A^T = -A$，即 $A_{i,j} = -A_{j,i}$，那么我们称 A 为斜对称矩阵。

定义 2.1 中的 Tutte 矩阵就是一个斜对称矩阵。可以看到，我们前面的算法都是在围绕计算 Tutte 矩阵的行列式展开，这正是我们研究斜对称矩阵性质的原因。

引理 3.4. *对于一个 $n \times n$ 的斜对称矩阵 A，如果 n 是奇数，那么 $\det A = 0$。*

证明. 由行列式的相关知识可知，$\det A = \det A^T$。又因为 $A^T = -A$，所以 $\det A = \det(-A) = (-1)^n \det A$。

当 n 为奇数时 $(-1)^n = -1$，即 $\det A = -\det A$，于是 $\det A = 0$。 □

定理 3.5. *对于一个 $n \times n$ 的斜对称矩阵 A 和一个行号的子集 $I = \{i_1, i_2, \ldots, i_k\}$，使得 $A_{i_1,\cdot}, A_{i_2,\cdot}, \ldots, A_{i_k,\cdot}$ 是 A 的一组关于行的极大线性无关组，那么 $\det A_{I,I} \neq 0$。*

证明. 不失一般性，设 $I = \{1, 2, \ldots, k\}$。

因为 $A_{1,\cdot}, A_{2,\cdot}, \ldots, A_{k,\cdot}$ 是 A 关于行的一组极大线性无关组，由 A 的斜对称性我们可以得到 $A_{\cdot,1}, A_{\cdot,2}, \ldots, A_{\cdot,k}$ 是 A 关于列的一组极大线性无关组。

因此，对于所有 $k < i \leq n$，$A_{\cdot,i}$ 都可以被 $A_{\cdot,1}, A_{\cdot,2}, \ldots, A_{\cdot,k}$ 线性表出。于是 $\operatorname{rank} A_{I,I} = \operatorname{rank} A_{I,\cdot} = k$，即 $\det A_{I,I} \neq 0$。 □

推论 3.6. *对于一个斜对称矩阵 A，存在一个行号集合 $I = \{i_1, i_2, \ldots, i_k\}$，使得 $\operatorname{rank} A = \operatorname{rank} A_{I,I} = k$。*

定理 3.7. *一个斜对称矩阵的秩为偶数。*

证明. 由推论 3.6 可知，斜对称矩阵存在一个秩与它本身相等的满秩主子式。由于斜对称矩阵的主子式也是斜对称的，结合引理 3.4 可知，这个主子式的秩为偶数。因此一个斜对称矩阵的秩为偶数。 □

引理 3.8. *对于一个 $n \times n$ 的可逆斜对称矩阵 A，以及任意的 $i, j \in \{1, \ldots, n\}, i \neq j$，我们有 $\det A^{i,j} \neq 0$ 当且仅当 $\det A^{\{i,j\},\{i,j\}} \neq 0$。*

证明. 如果 $\det A^{i,j} \neq 0$，那么 $\operatorname{rank} A^{i,j} = n - 1$。由于 $A^{\{i,j\},\{i,j\}}$ 是在 $A^{i,j}$ 的基础上删去一行一列得到，因此 $\operatorname{rank} A^{\{i,j\},\{i,j\}} \geq n - 3$。又因为 $A^{\{i,j\},\{i,j\}}$ 是斜对称矩阵，由定理 3.7 它的秩为偶数，因此 $\operatorname{rank} A^{\{i,j\},\{i,j\}}$ 一定为 $n - 2$。于是 $\det A^{\{i,j\},\{i,j\}} \neq 0$。

反过来, 如果 $\det A^{\{i,j\},\{i,j\}} \neq 0$，那么 $\operatorname{rank} A^{\{i,j\},\{i,j\}} = n - 2$，同时 $\operatorname{rank} A^{i,\{i,j\}} = n - 2$，根据定理 3.7，$\operatorname{rank} A^{i,\{i,j\}} = \operatorname{rank} A^{i,i} = n - 2$。这表明 $A^{i,\emptyset}$ 的第 j 列是其它列的线性组合。于是 $\operatorname{rank} A^{i,j} = \operatorname{rank} A^{i,\emptyset} = n - 1$，$\det A^{i,j} \neq 0$。 □

3.4 Rabin-Vazirani 算法

现在我们尝试得到一个比算法 2 更优的构造匹配方案的算法。

定理 3.9 (Rabin, Vazirani)**.** *设 $G = (V, E)$ 是一个有完美匹配的图，$\tilde{A} = \tilde{A}(G)$ 是 G 所对应的 Tutte 矩阵。那么，$(\tilde{A}^{-1})_{i,j} \neq 0$ 当且仅当 $G - \{v_i, v_j\}$ 有完美匹配。*

证明. 这个定理可以由定理 3.3，引理 3.8 以及 Tutte 定理（定理 2.1）得到。 □

基于定理 3.9，我们可以得到如下算法

算法 3 Matching algorithm of Rabin and Vazirani

```
M ← ∅
for i ← 1 to n do
  if v_i is not yet matched then
    compute Ã(G)^{-1}
    find j, such that v_i v_j ∈ E(G) and (Ã(G)^{-1})_{j,i} ≠ 0
    G ← G − {v_i, v_j}
    M ← M ∪ {v_i v_j}
  end if
end for
return M =0
```

算法 3 共需要进行 $O(n)$ 次循环，每次循环需要对一个 $O(n)$ 阶的方阵求一次逆，因此

定理 3.10. *算法 3 的复杂度为 $O(n^{\omega+1})$。*

3.5 基于高斯消元的构造方法

算法 3 的瓶颈在于计算 Tutte 矩阵 $\tilde{A}(G)$ 的逆。事实上，在每次迭代过程中，我们仅仅删除了 $\tilde{A}(G)$ 的两行两列，但却需要从零开始重新计算矩阵的逆。如果我们每次不再需要重新计算矩阵的逆，而是能够用一些高效的方法来维护矩阵的逆，那么我们就可以降低这个算法的复杂度。

我们的算法基于如下定理

定理 3.11. *如果*

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

其中 $\hat{a}_{1,1} \neq 0$。那么 $B^{-1} = \hat{B} - \hat{u}\hat{v}^T/\hat{a}_{1,1}$。

证明. 由于 $AA^{-1} = I$, 于是我们有

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

从而

$$\begin{aligned} B(\hat{B} - \hat{u}\hat{v}^T/\hat{a}_{1,1}) &= I_{n-1} - u\hat{v}^T - B\hat{u}\hat{v}^T/\hat{a}_{1,1} \\ &= I_{n-1} - u\hat{v}^T + u\hat{a}_{1,1}\hat{v}^T/\hat{a}_{1,1} \\ &= I_{n-1} - u\hat{v}^T + u\hat{v}^T \\ &= I_{n-1} \end{aligned}$$

□

注意到，在上面这个定理中，我们对 $\hat{B}$ 的修改操作恰好是高斯消元中的一次消去操作。在定理中我们是以消去第一行第一列为例来证明的。

作为定理 3.11 的一个直接应用，我们可以得到如下算法

算法 4 Simple matching algorithm

```
M ← ∅
compute Ã(G)^{-1}
for i ← 1 to n do
    if the i-th row is not yet eliminated then
        find j such that v_i v_j ∈ E(G) and (Ã(G)^{-1})_{j,i} ≠ 0
        G ← G − {v_i, v_j}
        M ← M ∪ {v_i v_j}
        update Ã(G)^{-1} by
        eliminating the i-th row and the j-th column
          and then the j-th row and the i-th column
    end if
end for
return M =0
```

算法 4 共需要循环 $O(n)$ 次，每次循环需要进行两次消去操作，单次消去操作的时间复杂度为 $O(n^2)$，因此

定理 3.12. *算法 4 的时间复杂度为 $O(n^3)$。*

4 最大匹配

4.1 最大匹配的大小

到目前为止，我们的所有算法都是用来求解完美匹配的，但在实际应用中我们要求解的往往是最大匹配。

与之前解决完美匹配问题的思路类似，我们先从最简单的问题入手：对于一个给定的图 $G = (V, E)$，如何求出图 G 的最大匹配的大小 $\nu(G)$？对于这个问题，我们有如下定理

定理 4.1. 图 G 的最大匹配的大小 $\nu(G)$ 是 $\frac{1}{2}\operatorname{rank}\tilde{A}(G)$。

证明. 设 $k = \operatorname{rank}\tilde{A}(G)$。由推论 3.6，我们可以选出 $\tilde{A}(G)$ 的一个行号集合 $I = \{i_1, i_2, \ldots, i_k\}$，使得 $\operatorname{rank}\tilde{A}(G) = \operatorname{rank}\tilde{A}(G)_{I,I} = k$。设 $U = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$，那么 $\tilde{A}(G)_{I,I} = \tilde{A}(G[U])$。由于 $\operatorname{rank}\tilde{A}(G[U]) = |U|$，因此图 $G[U]$ 存在完美匹配，这说明 $\nu(G) \geq \frac{k}{2}$。

如果 $\nu(G) > \frac{k}{2}$，我们考虑取出所有在最大匹配中的点，那么一定存在一个行号集合 I，使得 $|I| > k$ 并且 $\det\tilde{A}(G)_{I,I} \neq 0$，这意味着 $\tilde{A}(G)$ 的子矩阵的秩大于它本身的秩，显然这是不可能的。

综上，$\nu(G) = \frac{k}{2}$。 □

4.2 转化一

接下来我们尝试把最大匹配转化为我们所熟悉的完美匹配问题。如果对于一个图 G，我们已经求出它的最大匹配的大小 $\nu(G)$ 了，那么我们可以在图 G 的基础上新增 $n - 2\nu(G)$ 个点，并从新加进来的点往之前 n 个点的每一个连边，这样得到的新图就是一个有完美匹配的图了，并且这个新图的完美匹配与原图的最大匹配相对应。

这样我们就解决了最大匹配问题。这个转化的通用性非常好，但它有一个缺点，在最坏情况下我们可能要新增 n 个点，这将使我们的数据规模（点数）翻倍，因此将会显著增大算法的常数。

4.3 转化二

我们考虑另一个做法，同样是将最大匹配问题转化为完美匹配问题，不过这次我们将通过删点来实现。对于一个图 $G = (V, E)$ 和它的一个最大匹配 M，我们考虑所有在匹配 M 中的点构成的点集 U，显然 $G[U]$ 有完美匹配，并且 $G[U]$ 的完美匹配就是 G 的最大匹配。现在我们的思路就是设法找出这样的点集 U，然后对 $G[U]$ 应用我们之前求解完美匹配的算法。

实际上定理 4.1 的证明过程已经为我们指明了方向。由引理 3.5 和推论 3.6，我们只需要找到 $\tilde{A}(G)$ 的一组关于行的极大线性无关组就可以了。这可以通过一遍高斯消元求出来。

于是我们就可以在 $O(n^\omega)$ 的复杂度内把最大匹配问题转化为完美匹配问题，并且它不会增大我们算法的常数。

5 应用

本文所讲述的是解决一般图最大匹配问题的通用算法，因此它自然可以解决所有用到了一般图匹配的题目。

除此之外，它还有一些较为有特色的应用。下面我们来看两个例子。

5.1 结点接龙[6]

5.1.1 问题描述

Alice 和 Bob 在一个无向图 $G = (V, E)$ 上面玩游戏。游戏由双方交替操作，其中 Alice 先手。

一开始有一个棋子放在某个结点上，接下来每一回合玩家可以将这个棋子移动到相邻并且之前没有到达过（含起点）的结点上，无法操作的一方输。问哪些结点先手必胜。

5.1.2 算法介绍

首先我们有一个结论：一个点 v_i 是必胜态当且仅当它一定在图 G 的最大匹配上。[7]

于是问题转化为判定某个点是否一定在图 G 的最大匹配上。我们按照第 4.2 节的做法新建 $k = n - 2\nu(G)$ 个点并连上相应的边。设这样得到的新图为 G'，新建的点的编号分别为 $n+1, n+2, \ldots, n+k$。然后我们任选一个新点 $v_j (n+1 \le j \le n+k)$，那么对于原图中的每个点 $v_i (1 \le i \le n)$，如果 $v_i v_j$ 能够出现在图 G' 的完美匹配中，那么 v_i 就不一定在图 G 的最大匹配上。

由定理 3.9，我们只需要检查 $(\tilde{A}(G')^{-1})_{i,j}$ 是否为零就可以了。

可以看到，本文介绍的匹配算法在判断 “一个点是否一定在最大匹配上” 以及 “一条边是否一定在最大匹配上” 这类问题上具有优势。因为这个算法可以避免对增广路的分析，而只需要检查逆矩阵的某个位置是否为零。

[6]经典问题，英文为 Undirected Vertex Geography (UVG)。

[7]这个结论的证明不是本文的重点，这里略去，感兴趣的读者可以参考相关资料。

5.2 最大权匹配

5.2.1 问题描述

所谓最大权匹配是指让图 G 的每条边 v_iv_j 带上一个权值 $w_{i,j}$，然后现在我们的目标是求一个边权和最大的匹配 M。最大匹配相当于边权均为 1 的最大权匹配。

5.2.2 算法介绍

我们不妨在每对点之间都连上权值为 0 的边，当 n 是奇数时我们再新增一个点，并让它向所有点连边。因此原图的每一个带权匹配都会对应新图的至少一个带权完美匹配，并且新图的一个带权完美匹配对应了原图的一个带权匹配。

现在我们转化为求新图 G' 的带权完美匹配。我们新增一个变量 y。设 $\tilde{A}'(G')_{i,j} = \tilde{A}(G')_{i,j} \times y^{w_{i,j}}$。这样我们相当于要求 $\det \tilde{A}'(G')$ 中，y 的最高次数（这个值除以 2 就是最大权匹配）。

我们先把所有的 $x_{i,j}$ 赋上一个随机的值。然后设所有边的权值和不超过 W，那么我们可以将 0 到 W 中的每个值代入 y，算一遍行列式，并用这 $(W+1)$ 个结果插值得到一个关于 y 的多项式。那么这个多项式的最高项次数就是我们要求的东西。

这个算法的时间复杂度为 $O(Wn^\omega + W^2)$，在 W 和 n 均较小的时候有一定的价值。

6 总结

至此，我们已经能够在 $O(n^\omega)$ 的复杂度内求出一般图最大匹配的大小，并且在 $O(n^3)$ 的复杂度内构造出一组匹配方案。

本文所介绍的算法仍旧有着巨大的潜力等待我们去探究，并且本文所介绍的应用还较为基础。因此我希望本文能够起到抛砖引玉的作用，希望有读者能够拓展本文中的算法，将这一算法发扬光大，得到更加有趣的应用。

致谢

- 感谢中国计算机学会提供学习和交流的平台；
- 感谢宋新波老师，卓明聪老师多年以来的关心和指导；
- 感谢国家集训队教练张瑞喆和余林韵的指导；
- 感谢周子鑫同学与我分享这个算法；
- 感谢高闻远同学、武弘勋同学为本文审稿；

- 感谢父母对我的关心和照顾。

参考文献

[1] Mucha M, Sankowski P. Maximum matchings via Gaussian elimination[C]//Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on. IEEE, 2004: 248-255.

[2] Ivan I, Virza M, Yuen H. Algebraic Algorithms for Matching[J]. 2011.

[3] Cheung H Y. Algebraic algorithms in combinatorial optimization[D]. The Chinese University of Hong Kong, 2011.

[4] Huang C C, Kavitha T. Efficient algorithms for maximum weight matchings in general graphs with small edge weights[C]//Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2012: 1400-1412.

[5] 陈胤伯. 浅谈图的匹配算法及其应用[C]//2015 年信息学奥林匹克中国国家队候选队员论文集. 2015.