基于线性代数的一般图匹配算法

—— *Maximum Matchings via Gaussian Elimination* 阅读报告

AntiLeaf

January 2, 2022

Contents

1 摘要 **2**

2 引言 **2**

3 初步讨论 **2**
3.1 Tutte矩阵 2
3.2 最大匹配与完美匹配 3

4 构造完美匹配 **3**
4.1 利用高斯消元的算法 3
4.2 匹配的验证 4
4.3 针对二分图的算法 4
4.4 针对一般图的算法 5
4.4.1 基本想法 5
4.4.2 最简划分 5

5 结语 **6**

6 参考文献 **7**

7 附录: 部分证明 **7**

1 摘要

本文是 FOCS 2004 上发表的一篇论文 *Maximum matchings via Gaussian elimination* [1] 的阅读报告, 原论文的作者是来自华沙大学信息学研究所的 Marcin Mucha 和 Piotr Sankowski.

原论文的主要内容是介绍一种全新的二分图与一般图匹配算法, 这种算法的理论基础更多地来源于线性代数方面的知识(而非通常的组合数学).

根据论文内容, 这个算法无论在二分图还是在一般图上都可以做到 $O(n^\omega)$ 的复杂度, 其中 ω 是最快的已知矩阵乘法算法的复杂度指数. 众所周知 $\omega < 2.38$, 因此这个算法的最坏复杂度超越了带花树算法的 $O(n^{2.5})$, 是目前解决二分图和一般图完美匹配和最大匹配问题的最优算法.

同时, 论文也解决了一个长期存在的开放问题, 即 Lovász 提出的的在 $O(n^\omega)$ 时间内判断一个图是否有完美匹配的算法能否加以扩展, 使得其可以在 $O(n^\omega)$ 时间内构造出一组完美匹配方案.

由于在二分图上的算法与一般图区别不大, 因此本文将主要介绍一般图上的算法, 在二分图上的算法只会简单提及.

2 引言

由于和本文的关系不大, 匹配, 完美匹配, 最大匹配的定义在此不再赘述. 方便起见, 如无特殊说明, 以下的描述中设 $n = |V|$, $m = |E|$.

一般图匹配最先提出也是最广为人知的算法是 Edmonds [2] 的带花树开花算法(以下简称带花树算法). 这个算法朴素实现的复杂度是 $O(nm) = O(n^3)$, 经过优化后可以达到 $O(m\sqrt{n}) = O(n^{2.5})$ [3] .

另一方面 Lovász [4] 提供了一种能在 $O(n^\omega)$ 时间内判断一个图是否有完美匹配的算法, 目前为止它是判断完美匹配的最优算法. 实际上, 不只是图是否有完美匹配, 有关图的更多信息都可以在 $O(n^\omega)$ 时间内完成.

定理 1. $2 \le \omega < 2.38$.

本文中将介绍如何在期望 $O(n^\omega)$ 的时间内构造一个最大匹配的方案. 另外, 对于二分图的情况, 只需少许改动, 就可以利用经典的 LUP 分解算法 [8] 解决.

与论文中的顺序相同, 本文将先复述算法需要用到的线性代数知识, 然后介绍利用高斯消元算法寻找一组完美匹配的方案. 后面还会介绍如何简单的实现算法的 $O(n^3)$ 版本, 之后讨论如何在 $O(n^\omega)$ 时间内寻找一般图的完美匹配, 最后进行总结, 并提出几个开放问题.

3 初步讨论

3.1 Tutte矩阵

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

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

其中 x_{ij} 是一个变量, 因此 $\tilde{A}(G)$ 中共有 m 个变量.
在无歧义的情况下, 以下将 $\tilde{A}(G)$ 简写为 $\tilde{A}$.

定理 2 (Tutte [6])**.** 图 G 有完美匹配当且仅当 $\det \tilde{A} \neq 0$.

定理 3 (Lovász [4])**.** G 的最大匹配的大小等于 $\operatorname{rank} \tilde{A}$ 的一半.

现有的有关矩阵的算法一般都是针对确定数而非变量的. 可以取一个数域然后将所有变量随机赋值为数域中的数, 例如取素数 p 的剩余系, 并且一切有关运算都在模 p 意义下进行. 方便起见, 我们设为每个变量代入随机值后的矩阵为 $A(G)$.

定理 4. $A(G)$ 的秩至多为 G 的最大匹配大小的两倍, 并且等号成立的概率至少是 $1 - \frac{n}{p}$. [4]

实际应用中 p 的取值一般都远大于n, 因此错误的概率可以忽略不计, 也可以通过多次计算来进一步减小错误概率. 显然直接利用这个定理就可以在 $O(n^\omega)$ 时间内得知一个图是否有完美匹配, 也可以得知最大匹配的大小.

接下来讨论如何寻找一组完美匹配方案. 如果 G 有完美匹配, 根据刚才的结论 $A(G)$ 有很大概率可逆. 进一步地有如下定理:

定理 5 (Rabin, Vazirani [5])**.** 以下命题有非常高的概率成立*:* $A^{-1}_{j,i} \neq 0 \iff G - \{v_i, v_j\}$ 有完美匹配.

也就是说, 如果 $(v_i, v_j) \in E$, 并且 $A^{-1}_{j,i} \neq 0$, 那么这条边就有很大概率存在于某个完美匹配方案中. 下称这样的边 (v_i, v_j) 为**可行边**.

简便起见, 在无需特殊说明的情况下, 以下均省略“有很大概率”.

3.2 最大匹配与完美匹配

定理 6 (Rabin, Vazirani [5] ; Ibarra, Moran [7])**.** 寻找一组最大匹配方案的问题可以在 $O(n^\omega)$ 时间内规约到寻找一组完美匹配方案的问题.

这个定理十分重要, 因为它将我们试图解决的最大匹配问题转化为了一个简单许多的完美匹配问题.

基于这个定理, 接下来我们只考虑寻找完美匹配的问题.

4 构造完美匹配

4.1 利用高斯消元的算法

由 **定理 5** 可以得到一个比较显然的算法: 每次尝试寻找一个点对 (v_i, v_j), 如果 $A(G)^{-1}_{j,i} \neq 0$, 则说明 (v_i, v_j) 存在于某个完美匹配方案中. 将 v_i, v_j 从图中删去, 并将(v_i, v_j)加入匹配方案中, 然后在 $O(n^\omega)$ 时间内重新计算 $A(G)^{-1}$.

虽然 ω 是否等于 2 仍是开放问题, 但多数人相信 $\omega > 2$. 因此这个算法的复杂度是 $O(n(n^2 + n^\omega)) = O(n^{\omega+1})$.

算法的瓶颈在于删去 $A(G)$ 的两行两列之后重新计算 $A(G)^{-1}$. 实际上我们可以在 $O(n^2)$ 时间内维护这一过程:

定理 7 (消去定理)**.** 令

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

并且 $\hat{a}_{1,1} \neq 0$, 那么就有

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

定理中描述的是消去第一行第一列的情况. 实际上, 定理可以非常显然地推广到消去任意一行一列的情况, 因此我们只需在算法最开始计算一次 $A(G)^{-1}$, 后面每次删除两个点时执行两次 $O(n^2)$ 的消去过程即可.

定理 8. 上述算法可以在 $O(n^3)$ 时间内找到一组完美匹配.

4.2 匹配的验证

现在考虑如何将上面的算法优化到 $O(n^\omega)$. 算法的瓶颈在高斯消元部分, 因此我们尝试用更优的算法取代朴素的 $O(n^3)$ 消元.

先考虑一个简化版的问题, 即无需交换的高斯消元: 在消去前 $i-1$ 列后, 总能保证第 i 行的主元 $a_{i,i}$ 不为 0.

我们可以采用延迟更新的技巧, 即不必每次都对剩余行执行 $O(n^2)$ 的更新, 而是将几次修改放在一起执行.

我们有如下的算法:

算法 1 无需交换的高斯消元

```
function SIMPLE-ELIMINATION(A : Matrix)
    for i = 1 → n do
        lazily eliminate the i-th row and i-th column of A
        j ← the largest integer such that 2^j | i
        UPDATE({i+1, ..., i+2^j}, {i+1, ..., n})
        UPDATE({i+2^j+1, ..., n}, {i+1, i+2^j})
```

其中 UPDATE(R, C) 即为对 R 和 C 对应的若干行和列应用之前尚未执行的修改操作. 这一过程的核心实际上就是将若干次修改的向量 u_i 与 v_j 组合成矩阵 U 和 V 并相乘, 因此可以用快速矩阵乘法在 $O(n^\omega)$ 时间内解决.

不难证明这个算法的复杂度与 UPDATE 部分同阶, 即 $O(n^\omega)$. 实际上这个算法可以看作经典的快速 LU 分解算法 [8] 的简化版本. 在这里介绍这个算法的目的主要是引出如下定理:

定理 9. 设图 G 有完美匹配, 对 G 的任意一个匹配 M, 我们都可以在 $O(n^\omega)$ 时间内找到 M 的一个极大子匹配 M', 使得 M' 是某个完美匹配的子集.

4.3 针对二分图的算法

首先, 在二分图上 $\tilde{A}$ 的定义与一般图上稍有不同:

定义 (二分图的Tutte矩阵)**.** 对于一个二分图 $G = (\{U, V\}, E)$, 设 $n = |U|$, $m = |V|$, 定义 G 的 Tutte 矩阵为一个 $n \times m$ 矩阵 $\tilde{A}(G)$:

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

不失一般性, 在接下来的讨论中设 $n \geq m$.

定理 10 (Tutte [6])**.** G 的最大匹配的大小等于 $\tilde{A}(G)$ 的秩.

与一般图类似, 定义 $A(G)$ 表示将所有变量随机代入后的矩阵.

刚才我们讨论的算法只适用于不需要交换的情况, 因此还不能直接使用. 但借助经典的 LU 分解算法 [8] , 我们仍然可以在 $O(n^\omega)$ 时间内完成对 $A(G)$ 的消元, 这样就解决了寻找二分图完美匹配的问题.

4.4 针对一般图的算法

4.4.1 基本想法

对一般图来说, 每次找到一组匹配 (v_i, v_j) 后, 都需要消去矩阵的两行两列, 因此先前的延迟更新做法不再适用.

一个简单的想法是把 G 通过某种方法划分成若干个更容易找到完美匹配的子图, 然后对每个子图递归地运行算法.

算法 2 递归寻找一般图的完美匹配

```
1: function General-Matching(G : 无向图)
2:     贪心地找到一个匹配 M, 满足 |M| ≥ n/4
3:     调用 定理 9 描述的过程, 找到一个极大可行子匹配 M'
4:     if |M'| ≥ n/8 then
5:         对未匹配点的导出子图 G' 递归调用 General-Matching(G')
6:     else
7:         调用 Partition(G')
```

接下来考虑如何实现 Partition 部分.

4.4.2 最简划分

定义 G 是**基本的**, 当且仅当 G 有完美匹配, 并且 G 中所有可行边组成 G 的一个连通子图.

再定义一个点之间的二元关系: $u \sim v$ 当且仅当 $u = v$, 或者 $G - \{u, v\}$ 没有完美匹配.

Lovász [4] 有如下定理:

定理 11. 如果 G 是**基本的**, 则在 G 上定义的 $\sim$ 是一个等价关系.

设 $P(G)$ 表示 $\sim$ 对应的每个等价类组成的集合, 称为 G 的一个最简划分.

由 **定理 2** (Tutte 定理), 实际上由 $A(G)^{-1}$ 就可以得出 G 的最简划分.

定理 12. 设 G 是基本的, $S \in P(G)$ 并且 $|S| \geq 2$, 令 C 表示 $G - S$ 的任意一个连通分量. 则有

1. 将 $G - S$ 中每个连通分量缩成一个点之后形成的图 G'_S 是基本的.
2. $\forall v \in V(C), C - \{v\}$ 都有完美匹配.
3. 把 S 中的点缩成一个点 s, 再对应地与 C 中的点连边后形成的图 C' 是基本的.
4. $P(C') = \{\{s\}\} \cup \{T \cap V(C) | T \in P(G)\}$.

由此可知, $G-S$ 的连通分量个数就等于 $|S|$, 并且因为 S 中的任何两个点之间都不存在可行边, 在 G 的每个完美匹配中 S 中的点一定分别与 $G-S$ 中不同连通分量中的点匹配. 我们自然就可以得出, 任意一个 S 与 $G-S$ 的连通分量之间的匹配都可以还原出 G 的一个完美匹配.

这一部分是一个二分图匹配问题, 所以我们可以把一般图匹配问题借助二分图匹配转化成若干个更小的子问题, 从而递归解决.

算法 3 最简划分算法

```
function PARTITION(G)
    if G 不是基本的 then
        将 G 分解成若干个基本的子图, 对每个子图调用 PARTITION
    else
        令 S ∈ P(G), 并且满足 |S| = k ≥ 2. 再令 C_1, ..., C_k 表示 G − S 的每个连通分量
        设 C_1 是最大的连通分量, 并且如 定理 12 中描述的方式定义 C'_1, 根据 定理 12 , 由 P(G) 快速得到 P(C'_1)
        if P(C'_1) 包含一个非平凡的类 then
            调用 PARTITION(C'_1)
        else
            调用 GENERAL-MATCHING(C'_1)
        设上一步中求出的匹配是 M'_1, 令 c 表示其中与 s 匹配的点
        任取 S 中一个和 c 相邻的点 v, 求出一个包含 (v, c) 的, S 与 G − S 的每个连通分量之间的完美匹配, 将其加入全局的匹配中
        将每个 C_i 中与 S 匹配的点删除, 对其剩余部分递归地寻找完美匹配
```

注意到算法中每次选取最大的连通分量进行特殊处理, 其目的在于有效减小递归调用的子问题规模.

引理. 在上述算法中, 每次递归调用的子问题大小至多为原来的 $\frac{7}{9}$.

实际上, 为了保证划分的复杂度, 作者还引入了动态图连通性算法 [9] . 不过由于其太过复杂且与本文主题联系不大, 在此略去.

定理 13. 上述算法可以在 $O(n^\omega)$ 时间内找到 G 的一组完美匹配.

5 结语

论文中叙述了两位作者的成果, 即一个全新的可以在 $O(n^\omega)$ 时间内寻找一组完美匹配的算法, 并且借助 **定理 6** , 可以进一步地在 $O(n^\omega)$ 时间内找到一组最大匹配. 另外作者还提到了他们研究出了在 $O(n^{\omega/2})$ 时间内找到平面图的一组最大匹配的算法, 不过由于和本文主题关系不大, 在此略过.

文章的最后作者还提出了一个问题: 寻找一般图完美匹配的算法中有一个非常复杂的最简划分过程作为其子算法, 然而在用于二分图的算法中这并不是必要的. 是否可以简化前者的实现, 来省略划分算法?

实际上, 关于这个问题我还没有找到相关研究与资料. 在我看来问题的答案很可能是否定的, 因为一般图与二分图匹配最大的区别在于, 二分图中的点已天然被划分为两个集合, 而一般图不是; 因此划分算法很可能是必要的, 即使不是, 也应当是用一个类似的步骤取代它.

6 参考文献

[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 附录: 部分证明

定理 2. (Tutte) 图 G 有完美匹配当且仅当 $\det \tilde{A} \neq 0$.

证明. 考虑行列式的定义:

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

和式中的任意一个非零项对应的置换 π 都一定满足对任意的 k , $(k, \pi_k) \in E$. 考虑它的逆置换 π^{-1} 对应的项显然也是非 0 的, 而如果 π 中存在长度为奇数的置换, 则 π 与 π^{-1} 对应的项必定正负相消, 因此行列式中只有不包含奇置换的 π.

所以如果行列式不为 0, 就说明 G 至少存在一个偶环覆盖, 所以 G 就有完美匹配. 反之亦然.

定理 7. (消去定理) 令

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

并且 $\hat{a}_{1,1} \neq 0$, 那么就有

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

证明. 考虑到 $AA^{-1} = I_n$, 我们有

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

进一步地

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

证毕.

定理 9. 设图 G 有完美匹配, 对 G 的任意一个匹配 M, 我们都可以在 $O(n^\omega)$ 时间内找到 M 的一个极大子匹配 M', 使得 M' 是某个完美匹配的子集.

证明. 不失一般性, 设 $M = \{(v_1, v_2), (v_3, v_4), \ldots, (v_{k-1}, v_k)\}$, 未匹配点为 $v_{k+1}, \ldots, v_n$. 将 $A(G)^{-1}$ 的列按照 $v_2, v_1, v_4, v_3, \ldots, v_{n-1}, v_n$ 顺序重排, 而行顺序不变.

在执行 **算法 *1*** 时, 遇到 $a_{i,i} = 0$ 的情况就直接跳过, 那么最终被消去的行与列就对应了极大子匹配 M'.

证毕.

引理. 在 **算法 3** 中, 每次递归调用的子问题大小至多为原来的 $\frac{7}{9}$.

证明. 由于 C_1 是最大的连通分量, 我们只需要证明 C_1' 的点数至多为 $\frac{7}{9}n$ 即可.

如果 Partition(G) 是被 General-Matching(G) 调用的, 那么 G 就有一个匹配 $\hat{M}$, 它只包含非可行边, 并且覆盖了至少 $\frac{n}{3}$ 个点, 设这些点为 $\hat{V}$.

$\hat{M}$ 中的边无法跨越划分的两边, 因此划分结束时 C_1' 一定不包含 $\hat{V}$ 的任意一个点. 注意到每次划分时最大的连通分量至少减少三个点, 然后增加恰好一个新建的点. 因此当划分结束时 C_1' 中的点数最多是 $n - \frac{2}{3}|\hat{V}| \leq n - \frac{2}{9}n = \frac{7}{9}n$.

证毕.