Exact Algorithms of Graph Coloring

231240058 陈小川

Nanjing University

2024 年 12 月 5 日

Table of Contents

Table of Contents

动机

对于阶为 n、边数为 m、最大度为 Δ 的二分图，假设 $m \in \Omega(n)$，则该算法调用 MDMatching 算法（采用霍普克罗夫特-卡普算法计算二分图最大匹配）的时间复杂度为 $O(\sqrt{n}m)$，调用 EulerPartitioning 算法的时间复杂度为 $O(m)$，递归的深度为 $O(\log \Delta)$。因此，该算法的时间复杂度为 $O(\sqrt{n}m \log \Delta)$。

课外阅读

若采用更好的方法实现 MDMatching，则计算二分图正常 Δ 边染色的分治算法的时间复杂度可降为 $O(m \log m)$[71] 或 $O(m \log \Delta)$[72]。

动机

对于阶为 n、边数为 m、最大度为 Δ 的二分图，假设 $m \in \Omega(n)$，则该算法调用 MDMatching 算法（采用霍普克罗夫特-卡普算法计算二分图最大匹配）的时间复杂度为 $O(\sqrt{n}m)$，调用 EulerPartitioning 算法的时间复杂度为 $O(m)$，递归的深度为 $O(\log \Delta)$。因此，该算法的时间复杂度为 $O(\sqrt{n}m \log \Delta)$。

课外阅读

若采用更好的方法实现 MDMatching，则计算二分图正常 Δ 边染色的分治算法的时间复杂度可降为 $O(m \log m)$[71] 或 $O(m \log \Delta)$[72]。

- 如何在 $O(m \log m)$ 时间内找出二分图的正常 Δ 边染色？

算法回顾

对二分图的最大度分类讨论

- 若最大度为偶数，则将原图拆分成两个子图使得每个子图最大度等于原来的一半
- 若最大度为奇数，则找出原图一个饱和所有度数等于最大度的顶点的匹配，将其染成同一种颜色并删去，于是就能归结为偶数的情形

改进思路

- 将二分图转化为正则二分图
- 对正则二分图进行分治染色

转正则二分图

对于二分图 $G = \langle L \cup R, E \rangle$

- 若存在 $u, v \in L$，使得 $d(u) + d(v) \leq \Delta$，则将 u 和 v 合并为一个点，重复操作直至不存在这样的 u, v
- 对于 R 同理
- 如果合并后两边点的数目不等，则在点数较少的那边补足，记两侧点集为 L', R'
- 若存在 $u \in L'$，$d(u) < \Delta$，则必存在 $v \in R'$，$d(v) < \Delta$，在图上新加一条边 uv，重复操作直至图为正则二分图

转正则二分图

Lemma

在进行上述操作后，图的边数至多为 $2m + \Delta$

证

不妨假设合并后 $|L| \geq |R|$，至多有一个顶点 v_0 满足 $d(v_0) \leq \frac{\Delta}{2}$，因此添加的边数为，

$$\begin{aligned}\sum_{v \in L} \Delta - d(v) &\leq \sum_{v \in L} [2d(v) - d(v)] + \Delta \\ &= m + \Delta\end{aligned}$$

因此转为二分图后边数仍然为 $O(m)$

欧拉划分

假设图是正则二分图且每个顶点度数均为偶数，欧拉划分将会更简单：

- 只需要对图的每个连通分支找出一条欧拉回路，然后交替地把边放到两个子图当中

完美匹配

定理

任何非空 k 正则二分图均有完美匹配

证

利用霍尔定理即可

完美匹配

$O(m \log m)$ 时间求完美匹配

- 取最小的 t 使得 $2^t \geq m$，令 $\alpha = \lfloor \frac{2^t}{k} \rfloor, \beta = 2^t - k\alpha$，将图中每条边复制为 α 条重边，再将 L 和 R 中的点两两配对（不一定要有边），每对点之间添加 β 条虚拟边，最后得到 2^t 正则图
- 对这个图做欧拉划分，得到 2^{t-1} 正则子图 H_1 和 H_2，选择 H_1 和 H_2 中虚拟边较少的再次做欧拉划分，重复 t 次就能得到完美匹配

完美匹配

如何在 $O(m)$ 时间内完成欧拉划分

- 我们注意到前面的 2^t 正则图边数比原来多了很多，但有很多重边，经过去重的边数是 $O(m)$ 的
- 对于边 e，假设他有 $m(e)$ 条重边，我们将其中 $\lfloor \frac{m(e)}{2} \rfloor$ 条放入一个子图 H_R，另外 $\lfloor \frac{m(e)}{2} \rfloor$ 放入另一个子图 H_B
- 对于每条边都如此操作，最后每条边都只剩下 1 或 0 条重边，边数为 $O(m)$，再在这张图上跑欧拉划分就是 $O(m)$ 的

完美匹配

Theorem

上述过程得到的是原图的完美匹配

证

只需证明两件事

- 得到的是 2^t 正则图的完美匹配
  重复操作 t 轮后得到的是 1 正则图，因此正好是 2^t 正则图的完美匹配
- 结果不包含虚拟边
  虚拟边的数量为 $\beta n < kn = m \leq 2^t$，经过 t 轮以后就全部被筛掉了

分治染色

仍对图的最大度分奇偶讨论

- 如果最大度为偶数，我们可以在 $O(m)$ 时间内完成欧拉划分，将问题归结为对两个子图的染色
- 如果最大度为奇数，我们可以在 $O(m \log m)$ 时间内找出完美匹配并将其删去

分治染色

仍对图的最大度分奇偶讨论

- 如果最大度为偶数，我们可以在 $O(m)$ 时间内完成欧拉划分，将问题归结为对两个子图的染色
- 如果最大度为奇数，我们可以在 $O(m \log m)$ 时间内找出完美匹配并将其删去

时间复杂度分析

若度数 k 为偶数，有

$$T(m) = 2\,T(m/2) + O(m)$$

若 k 为奇数，有

$$T(m) = 2\,T(m/2) + O(m \log m)$$

利用主定理即可得到 $T(m) = O(m \log m)$

Table of Contents

整数规划

Definition

对于图 $G = \langle V, E \rangle$ 定义两个 0-1 矩阵 $\mathbf{X}_{n\times n}$ 和 $\mathbf{Y}_n$，含义如下：

$$X_{ij} = \begin{cases} 1 \text{ if vertex } v_i \text{ is assigned to color } j, \\ 0 \text{ otherwise} \end{cases}$$

$$Y_j = \begin{cases} 1 \text{ if at least one vertex is assigned to color } j, \\ 0 \text{ otherwise} \end{cases}$$

约束条件

$$X_{ij} + X_{lj} \leq Y_j \quad \forall (v_i, v_l) \in E, \forall j \in \{1, \ldots, n\}$$

$$\sum_{j=i}^{n} X_{ij} = 1 \quad \forall v_i \in V$$

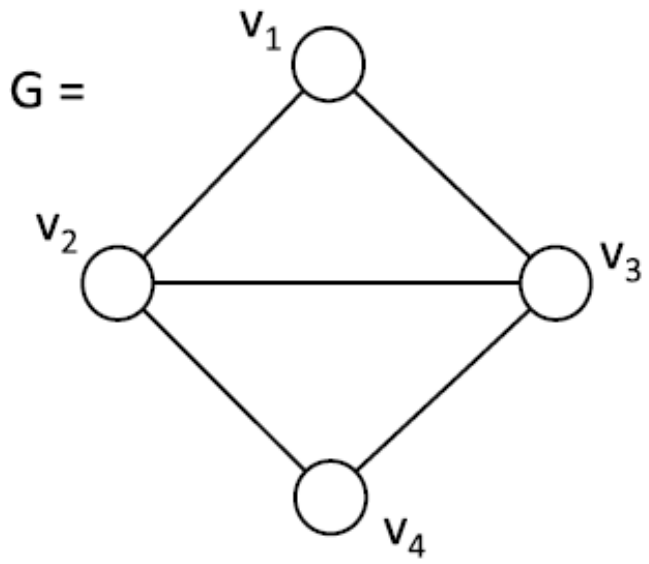
$$X_{ij},\ Y_j \in \{0, 1\}$$

例子

Example

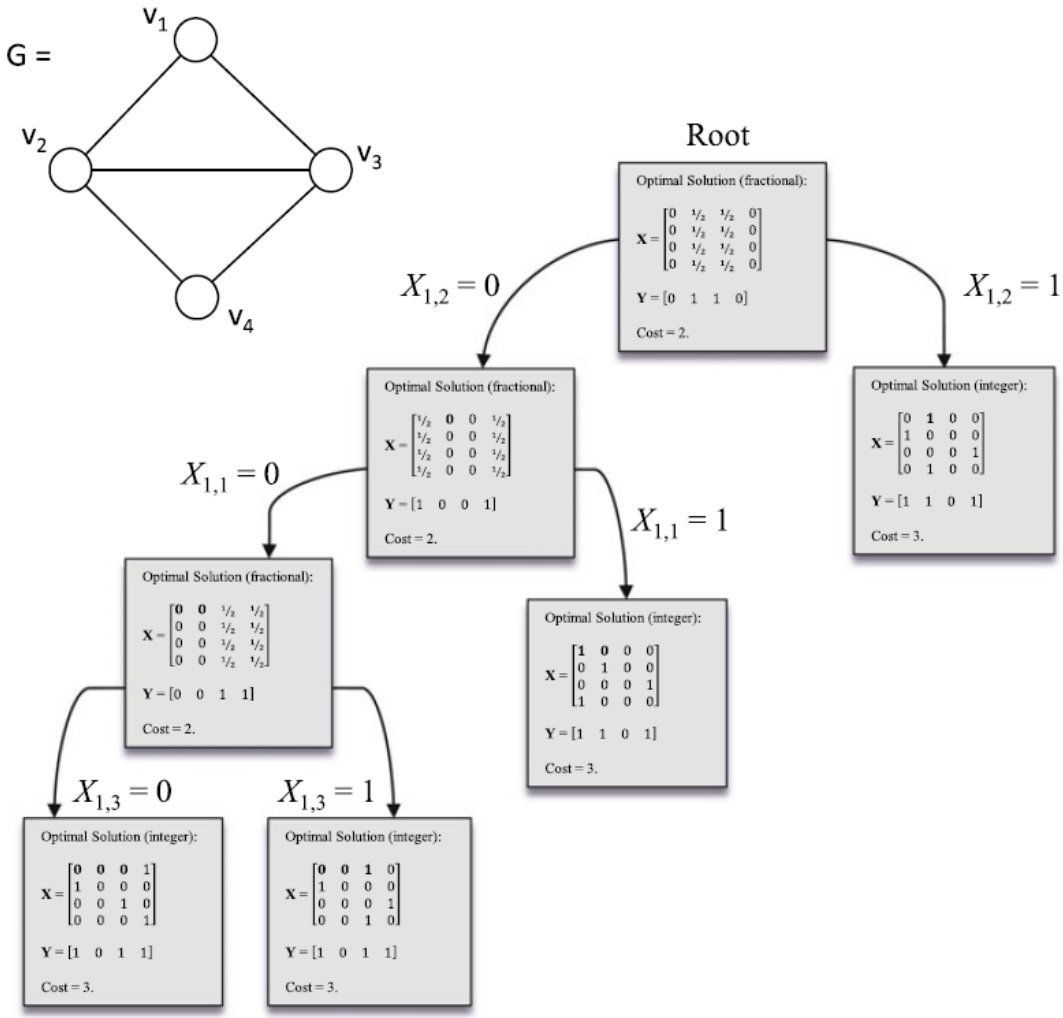
$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$$



分支定界法

- 先去除整数约束，即把每个变量都当成 $[0, 1]$ 区间上的连续变量，就相当于求解一次线性规划
- 如果线性规划的解为整数解，那么就是最优解
- 如果不是整数解，任取解中非整数的一个变量 X_{ij}，分两种情况，将 X_{ij} 的取值加入约束条件递归求解
  - $X_{ij} = 0$
  - $X_{ij} = 1$



分支定界法

在分支定界法运行过程中，我们需要维护两个值

- 迄今为止所取得的最优整数解，如果某一节点线性规划的结果高于这一值，就不需要扩展这一节点了，可以将其标注为 fathomed
- 所有 unfathomed 的叶子节点的线性规划最优解，一旦有整数解到达这个界，就可以直接结束程序了

模型优化

我们的模型有一个很大的缺点，就是会形成很多对称的解，例如，对 $\mathbf{X}$ 和 $\mathbf{Y}$ 的列进行重排就能得到两个表示不同但本质一样的解

$$\mathbf{X} = \begin{pmatrix} 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} 1&0&1&0&0&1&0&0 \end{pmatrix}.$$

$$\mathbf{X} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} 1&1&1&0&0&0&0&0 \end{pmatrix}.$$

模型优化

解决方法是添加更多的约束，例如

$$Y_j \geq Y_{j+1} \quad \forall j \in \{1, \ldots, n-1\}$$

$$\mathbf{X} = \begin{pmatrix} 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} 1&0&1&0&0&1&0&0 \end{pmatrix}.$$

$$\mathbf{X} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} 1&1&1&0&0&0&0&0 \end{pmatrix}.$$

模型优化

但同一组解仍对应整数规划的 $k!$ 组解，可以进一步约束：

$$\sum_{i=1}^{n} X_{ij} \geq \sum_{i=1}^{n} X_{ij+1}, \quad \forall j \in \{1, \ldots, n-1\}$$

模型优化

但同一组解仍对应整数规划的 $k!$ 组解，可以进一步约束：

$$\sum_{i=1}^{n} X_{ij} \geq \sum_{i=1}^{n} X_{ij+1}, \quad \forall j \in \{1, \ldots, n-1\}$$

还是有可能重复，可以把前面的约束替换为下面两个约束：

$$X_{ij} = 0 \quad \forall v_i \in V, j \in \{i+1, \ldots, n\}$$

$$X_{ij} \leq \sum_{l=j-1}^{i-1} X_{lj-1} \quad \forall v_i \in V - \{v_1\}, \forall j \in \{2, \ldots, i-1\}$$

实际上就是将 $\mathbf{X}$ 的所有列向量按字典序进行了排序

模型优化

在这一约束下同一个解就只能有一种表示

$$\mathbf{X} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 1&1&1&0&0&0&0&0 \end{pmatrix}$$

References

1. Alon, Noga (2003), "A simple algorithm for edge-coloring bipartite multigraphs", Information Processing Letters, 85 (6): 301–302, doi:10.1016/S0020-0190(02)00446-5, MR 1956451, S2CID 34965604
2. R. M. R. Lewis (2021), Guide to Graph Colouring, Algorithms and Applications.