二分图匹配

BY JIN

2018-11-13

1 匈牙利算法

1.1 算法示例

若邻接矩阵为：

1111011000
1000010100
0000010010
0100101000
0101001000

求最大匹配。

按行施行匹配：

1. [1] – (1)
2. [2] – (1) 已经匹配，在[1]处进行增广得[2] – (1) - [1] – (2)
3. [3] – (6)，及[2] – (1) - [1] – (2)
4. [4] – (2) - [1] – (1) - [2] – (6) ，但(6)已与[3]匹配，因此继续增广得 (6) - [3] – (9)
5. [5] – (2)，此时(2)的匹配项为[4]，因此对(2) – [4]链进行增广得 [5] – (2) - [4] – (5)，加上 [1] – (1) - [2] – (6) - [3] – (9) 为两条增广路径

故最终匹配方案为 [5] – (2), [4] – (5), [1] – (1), [2] – (6), [3] – (9)

若新增匹配时对应链无法增广，则对新增节点，需尝试下一匹配。若新增节点的所有匹配都无法增广，则加入该节点无论如何都无法增加匹配数，该节点可以抛弃。

1.2 时间复杂度分析

每行依次尝试各个匹配，但只在相应匹配链的末端进行增广；若增广的节点已经匹配，则递归增广。因此最坏情况下需对每条边进行一次计算，邻接矩阵情况下复杂度为O(Vl*Vl*Vr)，邻接表情况下为O(Vl*E)。

上述过程为深度优先的搜索。若采用广度优先的方式，则优先在**新增节点**处搜索尚未匹配的顶点，而非优先处理增广**路径末端**。虽然理想情况下每个节点只需O(Vr)或O(Ei)的尝试次数，但由于存在大量未合并的增广路径，一旦发生冲突，将导致更多次数的递归。因此总复杂度量级不变。

1.3 以最大流算法的角度

给所有行顶点加上一个共同的源s，给所有列顶点加上一个共同的宿t，s、t的每条边均有足够大的容量。此时原二分图的最大匹配，等价于s、t间的最大流。

考虑示例中的各个步骤

1. [1] – (1)，等于增广路径 s – [1] – (1) – t，增广后产生[1] – (1)容量等于0，产生反向流 (1) – [1]
2. 深度优先产生 s – [2] – (1) – [1] – (2) – t，[2] – (1)，(1) – [1]，[1] – (2)再次反向

3. s − [3] − (6) − t

   1–11011000
   –000010100
   00000–0010
   0100101000
   0101001000

4. s − [4] − (2) − [1] − (1) − [2] − (6) − [3] − (9) − t

   –111011000
   10000–0100
   00000100–0
   0–00101000
   0101001000

5. s − [5] − (2) − [4] − (5) − t

   –111011000
   10000–0100
   00000100–0
   0100–01000
   0–01001000

在矩阵中，每次增广先按行寻找正向路径、再按列寻找反向路径，如此循环，直到该列不存在反向路径为止（若某行不存在正向路径，则增广失败）。增广完后，按最大流算法，将该路径经过的所有边反向，事实上等价于匈牙利算法中的交换奇偶边。