

091M4041H - Assignment Five

NetFlow

张 帅

201828018670119

网络空间安全学院, UCAS

January 11, 2019

1 Load balance

1.1 题目描述

You have some different computers and jobs. For each job, it can only be done on one of two specified computers. The load of a computer is the number of jobs which have been done on the computer. Give the number of jobs and two computer ID for each job. Your task is to minimize the max load.

1.2 解题思路

假设有n个任务和m台计算机：

每个任务 J_i 可以由2台特定的电脑 $P_k \Delta P_l$ 执行，因此我们可以建立一个超级源点 S ，它指向m个jobs，capacity设置为1，流为1时，表示该job被执行；然后将每个任务 J_i 与其2台特定的电脑 $P_k \Delta P_l$ 建立有向边，capacity设置为1，流为1时，表示该边上的任务由该边所指向的电脑执行，该电脑负载加1；然后建立超级汇点 T ，所有电脑指向该汇点，每条边capacit的值都一样，设为 max_load ，表示电脑的最大运行负载，但需要动态设定。

下面就是确定指向汇点的 max_load 大小：题目要求我们要最小化最大负载的值，可以使用二分法来寻找最优的值。由题意知，一台电脑的最大负载发生在所有任务都在该电脑上执行，即负载为m，所以 max_load 的取值范围 $[0, m]$ ，且为整数。这样利用二分法，执行 $\log(m)$ 次最大流算法即可求出最优的值。

建图如下：

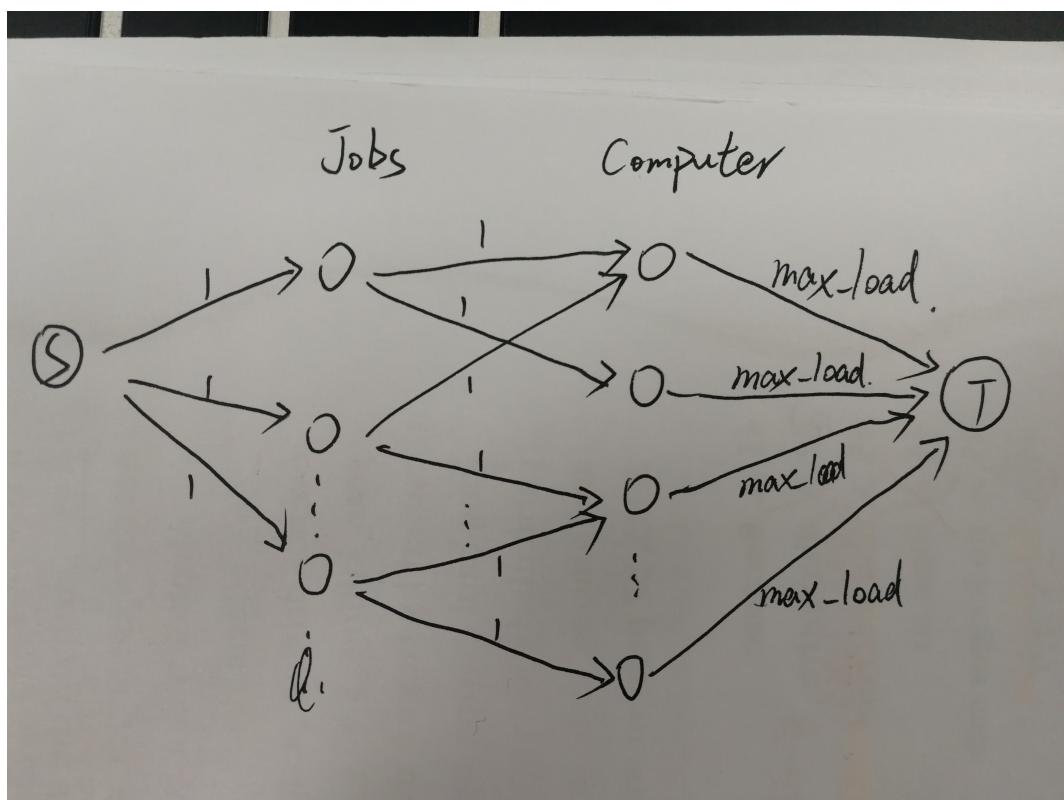


Figure 1: Load balance

PROBLEM 1 Load balance

INPUT: a directed graph G associated with a job and a computer

OUTPUT: Minimum load of the computer

```
1: function MINIMUMLOAD( $G$ )
2:   create super nodes  $S, T$ 
3:    $N = \#job$ 
4:    $r = N$ 
5:    $l = 0$ 
6:    $max\_load = 0$ 
7:   使超级源点 $S$ 指向所有的jobs, capacity设置为1; 图 $G$ 中的有向边的capacity都设置
    为1; 所有主机指向超级汇点 $T$ , capacity暂时设置为 $r$ ; 构造出一个网络流  $G'$ 
8:   while  $l >= num\_jobs$  do
9:      $mid = (l + r)/2$ 
10:    更新所有主机指向超级汇点的capacity, 设置为 $mid$ ;
11:     $f = Dinic(G')$ 执行Dinic算法, 求出最大流 $f$ 
12:    if  $f == M$  then
13:       $l = mid + 1$ 
14:       $max\_load = mid$ 
15:    else
16:       $r = mid - 1$ 
17:    end if
18:   end while
19:   return  $max\_load$ 
20: end function
```

1.3 伪代码

1.4 正确性证明

首先, 从以上的表述中可以看到, 我们动态设置了所有主机指向超级汇点的capacity max_load , max_load 为各个主机到超级汇点 T 的容量上限, 换句话说, 主机的最大负载不能超过 max_load 。另外我们设定超级源点到作业的capacity均为1, 每个作业到其特定的两个主机的capacity也为1, 这就保证每个工作只能运行在特定的两个主机的其中一个。然后我们运行最大流算法得出一个最大流 f , 如果 f 不等于作业数 N , 说明设置的 max_load 不可行, 需要增大; 反之可行, 可继续减小已测试是否有更小的设定。所以算法可行。

1.5 复杂度分析

假设有 n 个任务和 m 台计算机: 那么边的数目 $M = m + n + 2 * n = 3n + m$, 节点数目 $N = 1 + m + n + 1 = m + n + 2$, 运行Dinic算法的时间复杂度为 $O(N^2M)$, 使用二分法共需迭代 $\log_2 n$ 次, 所以总的复杂度为:

$$O(N^2M \log_2 n)$$

其中: $N = 1 + m + n + 1 = m + n + 2$ 、 $M = m + n + 2 * n = 3n + m$

2 Matrix

2.1 题目描述

For a matrix filled with 0 and 1, you know the sum of every row and column. You are asked to give such a matrix which satisfies the conditions.

2.2 解题思路

假设有n行m列。

我们添加超级源点S和超级汇点T，将每一行的和与每一列的和作为结点，行的和的结点与列的和的结点全连接，并设它们的capacity为1；然后让S与行和结点相连，capacity为每一行的和，列和结点与超级汇点T相连，capacity为每一列的和，对此网络流运行最大流算法，即可找到相应的矩阵。

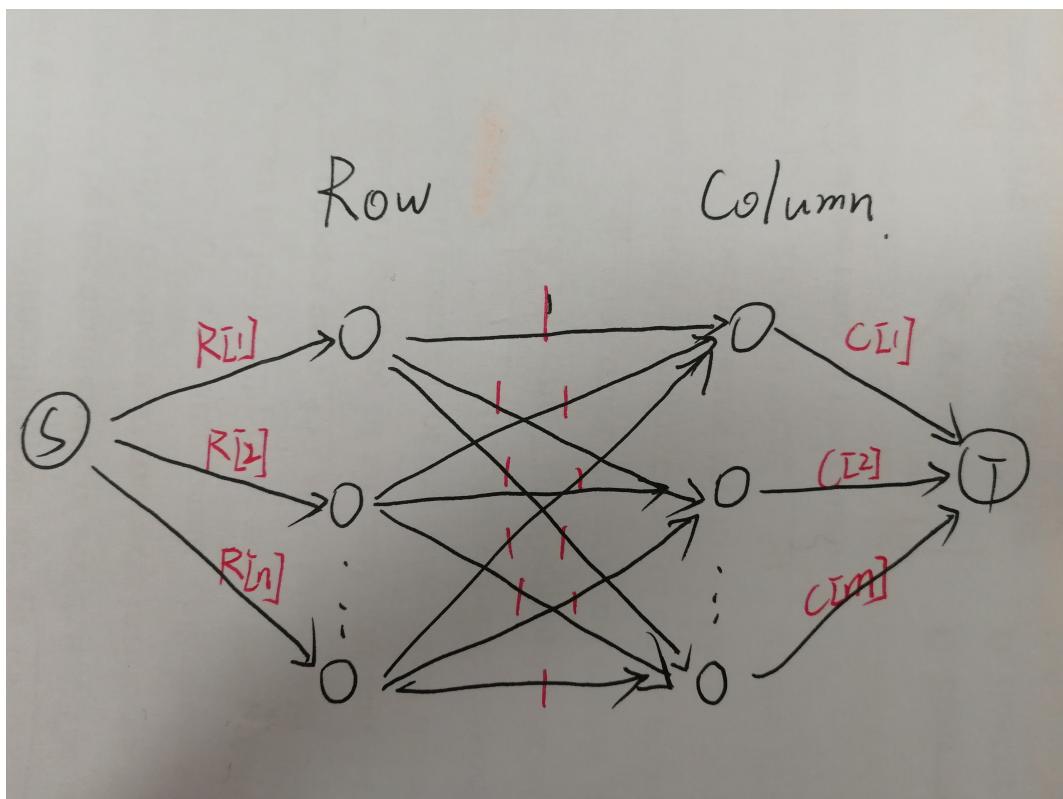


Figure 2: Matrix

2.3 伪代码

2.4 可行性证明

首先我们将每行每列的总和值都转化为流量，即边的capacity，用超级源点S指向各行结点，并设置capacity为行和，这就保证了每一行的最大和不超过其设定的和值；同理，列结点指向超级汇点T也满足条件；而行结点指向各列结点的capacity，我们都设置为1，这就保证在该行该列那个位置处最多会出现一个点。

然后运行Dinic算法求出最大流，如果最大流不等于总和，说明没有这样的矩阵存在；否则存在，然后获取网络流图中各个边的流量，根据行节点和列节点之间边的流量值构造矩阵，这就是最后的矩阵。所以算法正确。

PROBLEM 2 Matrix

INPUT: sum of each row and column; R,C

OUTPUT: Matrix

```
1: function FIND_MATRIX(R[n],C[m])
2:   if sum(R) != sum(C)
3:     return Null
4:   end if
5:   创建超级源点、汇点 S,T, 构造网络流图G
6:   f, f_ma = Dinic(G) 执行Dinic算法, 求出最大流f以及各边的流量值
7:   if f == sum(R) then
8:     根据f_ma构造矩阵Matrix
9:     return Matrix
10:  end if
11:  return Null
12: end function
```

2.5 时间复杂度

假设有n行m列。

那么边的数目 $M = m + n + m * n$, 节点数目 $N = 1 + m + n + 1 = m + n + 2$, 运行Dinic算法的时间复杂度为 $O(N^2M)$, 所以时间复杂度为:

$$O(N^2M)$$

其中: $N = m + n + 2$ 、 $M = m + n + m * n$

3 Problem Reduction

3.1 题目描述

There is a matrix with numbers which means the cost when you walk through this point. You are asked to walk through the matrix from the top left point to the right bottom point and then return to the top left point with the minimal cost. Note that when you walk from the top to the bottom you can just walk to the right or bottom point and when you return, you can just walk to the top or left point. And each point CAN NOT be walked through more than once.

3.2 解题思路

假设矩阵有n行m列。

根据题意，从左上角的点到最右下角的点，然后再回来，使得这条回路中没有经过重复的点，我们可以将原问题转化为找出从左上角的点到最右下角的点的两条没有经过重复节点的路径。

建图如下：

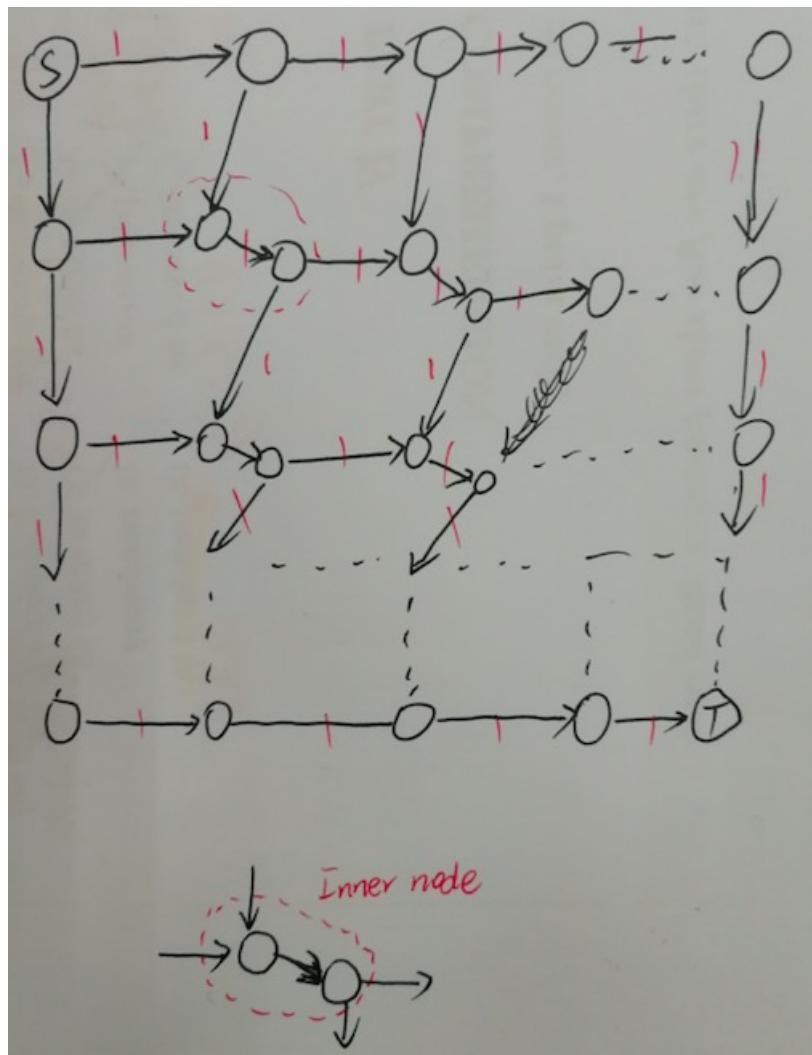


Figure 3: Problem Reduction

因此，可以设置左上角的点为源点S，右下角的点为T，又因为规则说明只能向右或向下，因此，只需要将当前节点指向它右面和下面的节点以建立连边，规则要求不能经过重复

的节点，那么更不可能经过同一条边，因此可以将capacity设置为1；规则要求不能就跟重复的点，我们可以将每个内部节点转化为两个节点，一个节点与左面和上面的点相连，另一个点指向右面和下面的点，设置这两个点之间的边capacity为1。然后运行Dinic算法，若最大流为2，则说明可以找到这样的两条路径。然后根据网络流图中的流量建立这样的两条路径，并将其中一条路径方向反转，这样就形成最终的解。

3.3 伪代码

PROBLEM 3 Problem Reduction

INPUT: Matrix, M

OUTPUT: Path, p

```

1: function FIND_MATRIX(M)
2:   根据上面建图思路构造网络流图G
3:    $f, f_{ma} = \text{Dinic}(G)$ 执行Dinic算法，求出最大流f以及各边的流量值
4:   if  $f == 2$  then
5:     根据 $f_{ma}$ 构造两条路径 $p_1, p_2$ 
6:     反转 $p_2$ 的方向，将其与 $p_1$ 拼接，构造出路径p
7:   return p
8: end ifreturn Null
9: end function

```

3.4 可行性证明

首先每个节点只指向右面和下面的点，这就限制路径只能朝右面或下面的方向。边的capacity设置为1，这样每条边至多经过一次；对于节点，根据上面建图可直观发现内部节点被分离后的两个节点之间的capacity设为1，这样就限制了该节点最多只能经过一次。而对于外围节点，由于其左右或上下的边capacity都设置为1，它的入边总capacity和出边的总capacity必有一个为1，因此，这就限制外围节点只能经过一次，这也是不需要对其进行分离的原因。

另一方面，原题要求找一个回路，我们将其转化为找两条从左上到右下的路径，因为这两条的路径的源点都为S，汇点都为T，将其中一条路径方向翻转，这样源点就变成T，汇点就变成S，拼接起来正是原题要求的路径。

如果运行最大流算法能找到这样的两条路径，即最大流为2，则原问题存在解；否则，无解。

3.5 时间复杂度

假设有n行m列。

那么边的数目 $M = (n - 1) * (m - 1) + (m - 2) * (n - 2) = 2mn - 3m - 3n + 5$ ，节点数目 $N = 2 * m + 2 * n - 4 + (m - 2) * (n - 2) * 2 = 2mn - 2m - 2n + 4$ ，运行Dinic算法的时间复杂度为 $O(N^2M)$ ，所以时间复杂度为：

$$O(N^2M)$$

其中： $N = 2mn - 2m - 2n + 4$ 、 $M = 2mn - 3m - 3n + 5$