深 圳 大 学 实 验 报 告

课程名称：算法设计与分析

实验项目名称：最大流应用——论文评审问题

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：杨烜

报告人：郑雨婷 学号：2021150122 班级：高性能

实验时间：2023/6/8——2023/6/15

实验报告提交时间：2023/6/15

教务部制

实验目的 （1）掌握最大流算法思想。 （2）学会用最大流算法求解应用问题。
实验内容：论文评审问题 （1）有m篇论文和n个评审，每篇论文需要安排a个评审，每个评审最多评b篇论文。请设计一个论文分配方案。 （2）要求应用最大流解决上述问题，画出m=10，n=3的流网络图并解释说明流网络图与论文评审问题的关系。 （3）编程实现所设计算法，计算a和b取不同值情况下的分配方案，如果没有可行方案则输出无解。

实验过程及内容：

一、问题分析

给定论文数 m,评委数 n,每篇论文需要的评委数 a,每个评委评审最多评 b 篇论文。我们知道有且仅有在满足下面两个条件时可能有解：

①$n>=a$

②$a*m>=b*n$

因为只有评委数大于等于每篇论文需要的评委数时才有可能有解，同时还需要保证所有评委能评阅的总量大于所有论文被评审的次数，这样才可能有解。在满足了上面的两个条件后，我们可以运用最大流的思想来解决问题。

二、构建流网络

要运用最大流的思想解决问题，首先需要有一个流网络，下面以 m=10,n=3 的情况来绘制流网络图。要构建一个流网络，首先需要有源点和汇点，再将 10 篇论文和 3 个评委都抽象为点，得到了流网络中的所有节点。

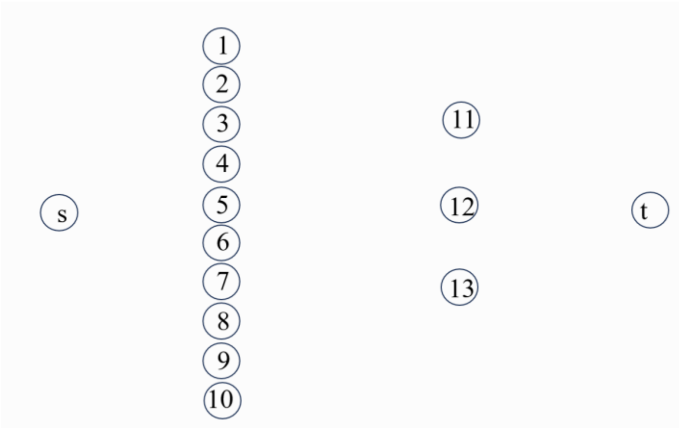

图 1 构建流网络

每篇论文需要安排 a 个评审，因此源点与论文点之间的边容量为 a。每个评委最多评 b 篇论文，因此评委与汇点之间的边容量为 b。评委和论文之间只有两种状态，一种是该评委评审了该论文，另一种是该评委没有评审该论文，因此论文和评委之间边的容量为 1。将论文点与评委点两两相连，最终，构建出如下图所示的流网络图。

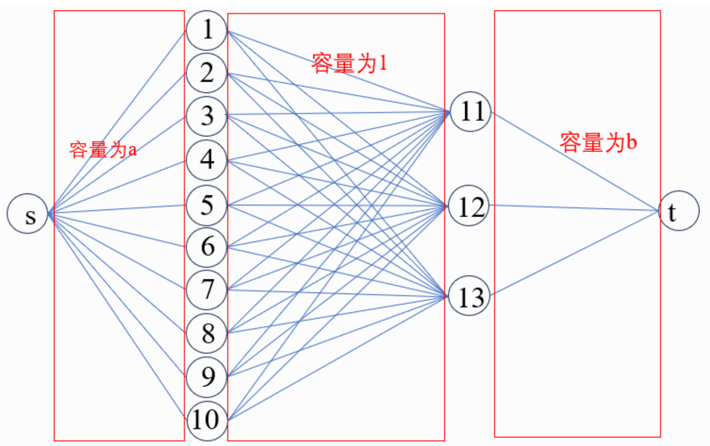

图 2 构建流网络

流网络图与论文评审问题的关系如下：

①在上面的流网络中求最大流，若最大流等于$a\times m$，说明每一篇论文都被 a 个评委评价过了，即为有解。若不相等(小于$a\times m$)，即为无解。

②论文与评委之间的边，流量为 1 的代表该评委评了该论文，流量为 0 代表该评委没有评该论文。

因此，在上面构造的流网络中求最大流，若最大流不等于$a\times m$，则输出无解。反之，遍历论文与评委之间所有边，根据这些边的流量值是 0 或是 1，得到评委是否评了这篇论文，输出论文分配的方案。

三、求最大流的不同方法

相关概念：

①增广路径：从源点 s 到汇点 t 之间的一条路径，该路径上不存在边容量小于等于 0 的边。下面粗边即为一条增广路径。

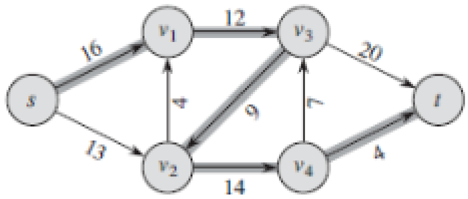

图 3 增广路径示意图

②残留网络：残留网络在最大流问题中是一个非常重要的概念。它是指在原始网络上，考虑已经通过某些路径分配了一定容量后，仍然可以增加流量的那些边所组成的网络。具体来说，在一个残留网络中，对于一条有向边(u, v)，如果其原本的流量为 f(u, v)，该边的容量为 c(u, v)，则其剩余容量即为 c(u, v) - f(u, v)；反向边(v, u)的剩余容量即为 f(u, v)。例如，在图 3 中沿找到的增广路径压入流量为 4 的流时，得到残留网络图 4。

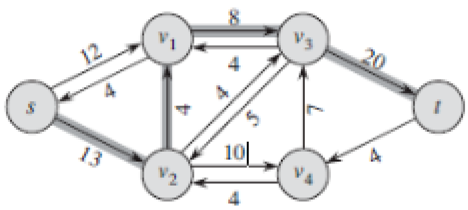

图 4 残留网络示意图

3.1 Ford-Fulkerson 方法

1.方法思想：

若在残留网络中存在一条增广路径，就沿该路径压入流，流量由路径上的最小容量限制。然后再找到另一条增广路径压入流，一直到网络中不存在增广路径为止。

方法步骤：

（1）构建流网络

（2）在残留网络上寻找增广路径

（3）在流网络中的增广路径上压入流

重复（2）—（3）直到没有增广路径

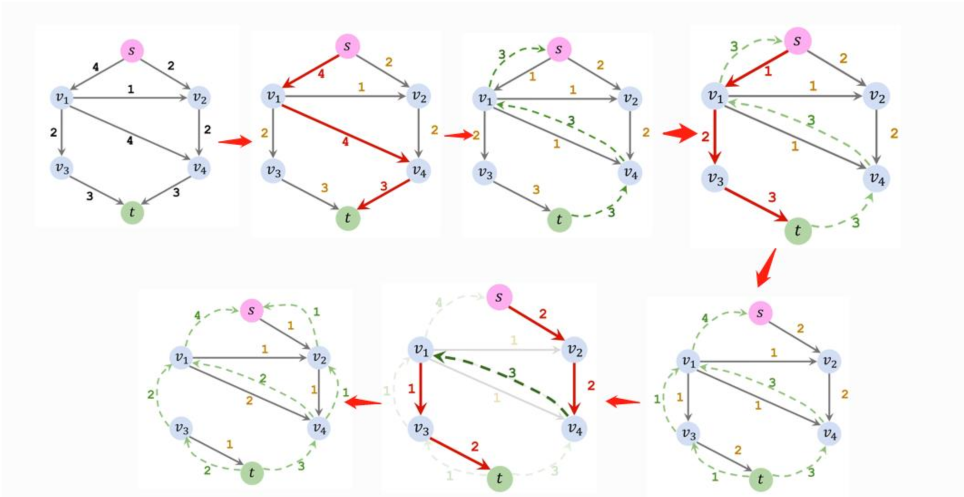

图 5 Ford-Fulkerson 方法思想示意图

2.算法实现

基础的 FF 方法寻找增广路径使用的是深度优先遍历（DFS），每次 DFS 时将较小的容量递归传入入下一次 DFS，这样直到当找到汇点时，当前的流就是这条路径上最小的容量，记为 d。之后再将 d 压入流网络，也就是将这条路径上的正向边加上 d,反向边减去 d。

```
FF(){
    init();
    flow = 0;
    while (1){
        init(vis)
        tmp = dfs(S, INT_MAX)
        if tmp==0: break
        flow += tmp
    }
    return flow;
}
```

```
int dfs(x,flow) {
    if (x == T) return flow;
    vis[x]=1;
    for i = 0 to adj[x].size():
        if adj[x][i].c> 0&&vis[adj[x][i].v2]==0:
            d = dfs(adj[x][i].v2, min(flow, adj[x][i].c))
            if d > 0:
                adj[x][i].c-=d;
                adj[adj[x][i].v2][adj[x][i].rev].c+=d;
                return d;
}
```

图 6 FF 方法伪代码

3.效率分析

设 f*是最大流,E 是边数。在 DFS 过程中，每一次迭代最大流至少增大 1，因此最大流的求解迭代次数至多为 f*。又因为每一轮 DFS 的复杂度为 O(E)。基本的 Ford-Fulkerson 算法的时间最坏时间复杂度为：$O(E|f*|)$

3.2 Edmond-Karp 算法

1.算法思想：

EK 算法是 FF 方法的一个具体实现，整体思想和 FF 方法一致。知识在寻找增广路径时，它采用广度优先遍历（BFS）的方式，这样可以确保每次找到的增广路径都是长度最短的路径。

算法步骤：

（1）构建流网络

（2）在残留网络上用 BFS 寻找增广路径

（3）在流网络中的增广路径上压入流

重复（2）—（3）直到没有增广路径

2.算法实现

寻找增广路径使用的是深度优先遍历（BFS），这样直到当找到汇点时，就找到了一条增广路径。但与 FF 方法使用的 DFS 不同，BFS 需要注意步骤（3）压入流时，无法像 DFS 那样递归天然地返回上一级节点，因此需要记录每个节点的前驱，在压入流 update 时按照前驱后继关系压入。

```
int Edmond_Karp() {
    flow = 0;
    while (bfs()) :
        flow += update();
    return flow;
}
```

```
int update(){
    delta=flow[T],v=T;
    while v!=S:
        int u=pre[v];
        for i=0 to [u].size():
            if adj[u][i].v2==v:
                adj[u][i].z-=delta;
                adj[v][adj[u][i].rev].z+=delta;
        v=u;
    return  delta;
}
```

```
int bfs()
{
    init(vis)
    queue<int> q;
    flow[0] = INT_MAX;vis[0]=1;
    q.push(0);
    while !q.empty():
        u = q.front();q.pop();
        for  i = 0 to adj[u].size():
            v2=adj[u][i].v2;
            if adj[u][i].c>0&&vis[v2]==0:
                flow[v2]=min(flow[u],adj[u][i].z);
                pre[v2]=u;
                q.push(v2);
                vis[v2]=1;
                if v2 == T: return 1;
    return 0;
}
```

图 7 EK 算法伪代码

3.效率分析

存在如下定理：如果对具有源点 s 和汇点 t 的一个流网络 G=(V,E)运行 Edmonds-Karp 算法，对流进行增加的全部次数为Ｏ(VE)。由于在用广度优先搜索寻找增广路径时，Ford-Fulkerson 中的每次迭代都可以在Ｏ(E)时间内完成，所以 Edmonds-Karp 算法的全部运行时间为$O(VE^2)$。

3.3 Dinic 算法

1.算法思想：

Dinic 算法的思想是分层次的在网络中寻找增广路径。先使用 BFS 对图进行分层，然后用 DFS 寻找增广路径。在 Dinic 算法中，我使用了多路增广进行。它与 EK 算法的不同之处在于：EK 算法每个阶段执行完一次 BFS 增广后，需要重新 BFS 从源点开始寻

找另一条增广路径。而多路增广后，只需一次 DFS 过程就可以实现多次增广。

算法步骤：

（1）构建流网络

（2）用 BFS 对图进行分层

（3）在残留网络上 DFS 寻找多条增广路径

（4）在流网络中的增广路径上压入流

重复（2）—（4）直到没有增广路径或者已经联通汇点。

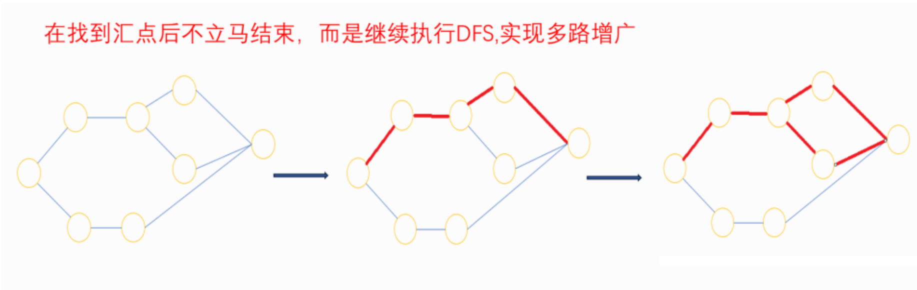

图 8 Dinic 多路增广示意图

2.算法实现

首先初始化每个节点的层数都为 0，用 BFS 对图网络进行分层，之后用 DFS 寻找增广路径。注意在第一次找到汇点后不立马结束，而是继续进行 DFS,如果残余流量没有用完，可以利用残余部分流量，再找出一条增广路。这样就可以在一次 BFS 中找出多条增广路，大大提高了算法的效率。下面图 9 为 Dinic 算法的伪代码，其中 BFS 与 EK 算法中的相似，不再给出。

```
int Dinic()
{
    flow = 0;
    while true:
        bfs(); //对图分层
        if level[T] < 0: break;
        delta= dfs(0, inf);
        if delta==0: break;
        flow += addflow;
    return flow;
}
```

```
int dfs(u,f)
{
    if u == T: return f;
    sum=0;
    for i = 0 to  adj[u].size():
        EDGE &e = adj[u][i];
        if e.c > 0 && level[e.v] ==level[u]+1:
            d = dfs(e.v, min(f, e.c));
            if d>0:
                e.c -= d;
                adj[e.v][e.rev].c += d;
                sum+=d;
                f-=d;
                if f==0: break;
    if sum==0: level[u]=-1;
    return sum;
}
```

图 9 Dinic 算法伪代码

3.效率分析

每轮 BFS 搭建分层图找到的增广路的数量至少为 1，增广路的数量每次都减少至少一条，整个网络中最多有 V - 1 条增广路，（V 为顶点数量），最多 V-1 次，时间复杂度为 O(V).分层图可以在 O(E)的时间复杂度内用 BFS 构建。一条增广路可以在 O(VE)的复杂度内构建。Dinic 算法整体的时间复杂度为：

$$\mathbf{O(V)} \times [\boldsymbol{O(E)} + \boldsymbol{O(VE)}] = \boldsymbol{O(V^2E)}$$

数据处理分析：

运用控制变量法，分别改变 a,b,m,n 测出三种方法的运行时间，得到图 10 的四幅曲线。

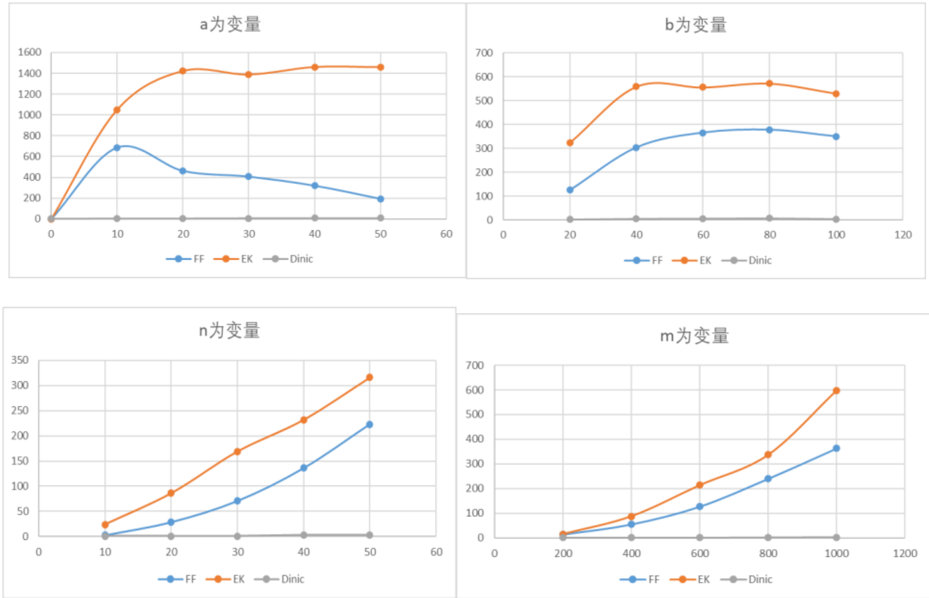

图 10 实际运行时间

可以看到 EK 方法比 FF 慢，这是因为在论文评审问题中，图的层次是固定的，只有源点、论文、评审、汇点这四层，这样 DFS 效率不会很低。但 EK 算法使用的 BFS 会随着 a,b,m,n 的增加而宽度增加，因此效率较低。但 FF 方法和 EK 算法的效率都明显低于 Dinic 算法，这是因为 Dinic 算法中使用了多路增广，相对于单个增广路，多路增广能够更有效地利用残留网络中存在的全部潜在增广路，这种方法会大幅度减少重复搜索的情况，从而节省了计算时间。

实验结论：

在这次实验中，我运用最大流的思想解决了论文评审问题，这个问题是实际生活中真正能够遇到的问题，说明最大流的思想可以运用在日常生活中来帮助我们解决问题。在最开始构建流网络图时，需要理解题意，将实际问题转换为数学模型是一项极为重要的能力。

在求最大流时，首先需要了解流网络中的增广路径、残留网络等概念。之后我学习到了三种方法：**Ford-Fulkerson、Edmond-Karp 和 Dinic。**Dinic 算法的效率最高，因为多路增广可以避免重复计算，这与实验五的路径压缩思想相同，都是尽量避免做无用功。在学习了算法的思想过后，真正通过运行程序体会到三种方法的效率不同。还有这次实验又用到了 DFS、BFS，这些基础知识十分重要。

通过这次实验，对于流网络的最大流有了进一步的认识，掌握了三种求解最大流的方法，收获颇丰。

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。