深 圳 大 学 实 验 报 告

课程名称：算法设计与分析

实验名称：最短增益路径法求解最大流问题

学院：计算机与软件学院 专业：软件工程

报告人：郑杨 学号：2020151002 班级：腾班

同组人：陈敏涵

指导教师：李炎然

实验时间：2022.6.23~2022.6.27

实验报告提交时间：2022.6.27

教务处制

目录

一、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。

二、实验内容与要求

1 实验内容

一个医院有 n 名医生，现有 k 个公共假期需要安排医生值班。每一个公共假期由若干天（假日）组成，第 j 个假期包含的假日用 D_j 表示，那么需要排班的总假日集合为 $\mathcal{D} = \cup_{j=1}^{k} D_j$。例如，“五一”假期由5月1日至5月7日一共7个假日组成。“元旦”假期由1月1日至1月3日一共3个假日组成。

每名医生 i 可以值班的假日集合是 S_i，$S_i \subseteq \mathcal{D}$。例如，李医生可以值班的假日集合包括“五一”假期中的5月3日、5月4日和“元旦”假期中的1月2日。

设计一个排班的方案使得每个假日都有一个医生值班并且满足下面两个条件：

1. 每个医生最多只能值班c个假日；
2. 每个医生在一个假期中只能值班1个假日。例如，安排李医生在“五一”假期中的5月4日值班。

根据上述场景完成下面任务：

1. 实例化上述场景中的参数，生成数据。
2. 设计一个多项式时间的算法求解上述问题，具体任务如下
    a. 基于生成的数据，设计一个流网络；
    b. 解释说明该流网络中最大流与值班问题的解的关系；
    c. 基于生成的数据，计算出排班的方案。

2 实验要求

1、在blackboard提交电子版实验报告，注意实验报告的书写，整体排版。
2、实验报告的实验步骤部分需详细给出算法思想与实现代码之间的关系解释，不可直接粘贴代码（直接粘贴代码者视为该部分内容缺失）。
3、实验报告中要求证明该算法的关键定理，并说明这些定理所起的作用。
4、源代码作为实验报告附件上传。
5、在实验课需现场运行验证并讲解PPT。

三、实验步骤与结果

1 分析问题

本次实验想要解决的问题是医生排班问题，有 n 名医生，k 个公共假期，每个假期包含若干个假日，第 j 个假期包含的假日用 D_j 表示，那么需要排班的总假日集合为 $\mathcal{D} = \cup_{j=1}^{k} D_j$。每名医生可以值班的日期不同，医生 i 可以值班的假日集合是 S_i，$S_i \subseteq \mathcal{D}$。想要求解的是值班的可行方案，满足以下三个条件：

- 每个假日都有医生值班
- 每名医生在一个假期中最多值一天班
- 每名医生值班的天数不超过c

由于想要使用流网络对该问题，需要对该问题进行抽象，找出实体（医生、假期、假日）之间的关系。对问题进行分析不难发现，实体之间存在着下列几种关系（约束）：

- 每个假期包含一些假日
- 每名医生可以值班的假日不同
- 每个医生最多值c天班
- 每名医生在一个假期内最多值一天班

下面给出一个例子阐述具体关系。假设现在需要值班的假期为**五一假期和元旦假期**，五一假期的假日包含 5 月 1 号、5 月 2 号和 5 月 3 号，元旦假期的假日包含 1 月 1 号和 1 月 2 号，如图 1 所示。假设有三名医生 Doc.Zheng、Doc.Chen 和 Doc.Xu，其中 Doc.Zheng 可以值班的日期为 1.1 和 5.1，Doc.Chen 可以值班的日期为 1.2、5.1 和 5.2，Doc.Xu 可以值班的日期为 1.2、5.2 和 5.3，每名医生最多可以值 2 天班。

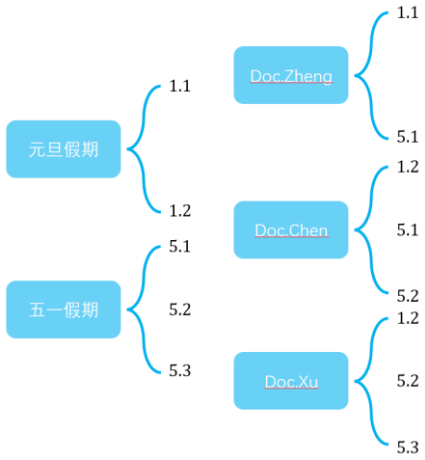


图 1：简单例子

于是可以直观的看出，医生与假日存在较强的关系同时假期与假日也存在着约束关系，约束医生在一个假期中最多值一天班。

2 流网络模型

2.1 建立模型

对于上述例子，不难建立一个简单的流网络模型满足医生与假日之间的约束。把医生与假日抽象为图中的点，每个医生向每一个可以值班的日期连一条容量为 1 的边，满足医生与假日之间的约束。同时，为了满足每个医生最多值2天班，可以通过使用超级源点，向每个医生连一条容量为2的边，起到限流的作用。由于每一个日期都需要被值班，于是使用一个超级汇点，将每个日期向超级汇点连一条容量为 1 的边，如图 2 所示。

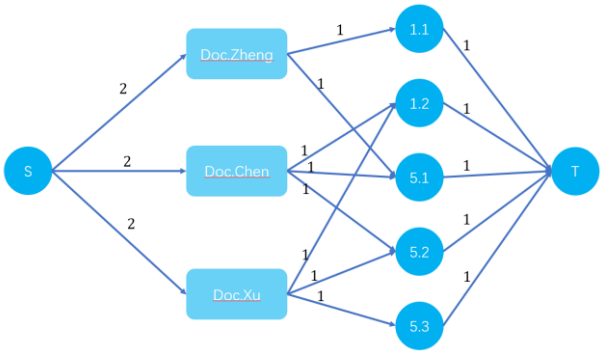


图 2：不完备的简单模型

上述模型解决了医生与假日的关系，但少了假期与假日，假期与医生的关系，于是需要进行完善，满足每名医生在一个假期中最多值一天班。将假期也抽象为节点，由于每个医生假期中可以值班的日期不同，故必须使假期更为具体来表示医生与假期的关系，具体为医生 X 的 Y 假期，如图 3 所示。

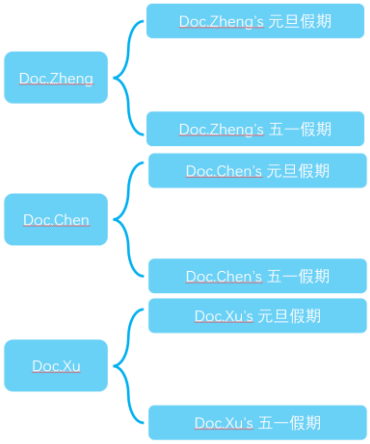


图 3：假期与医生的关系

于是可以把抽象出来的虚拟假期节点加入之前建立的流网络模型中，医生层与假日层之间，先将每名医生向对应的虚拟假期节点连一条容量为 1 的边，通过限流的方式表示医生在这个假期中只能值一天班。然后将假期向该医生对应的可以值班的假日连容量为 1 的边表示医生与假日的关系，如图 4 所示。

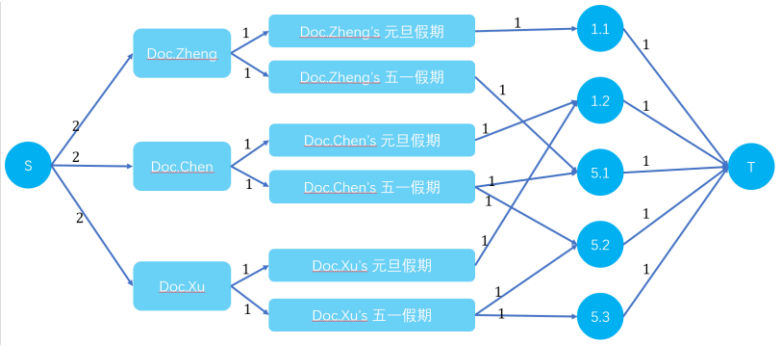


图 4：最终建立的模型

2.2 模型正确性解释

在图 4 所示模型上跑最大流算法计算出来的最大流，就表示了原问题的一种解决方案。从源点到汇点的流表示的就是某医生值某日期的班，如图 5 所示，该流表示了 Doc.Zheng 值 1 月 1 号的班，并且该值班方案满足约束。

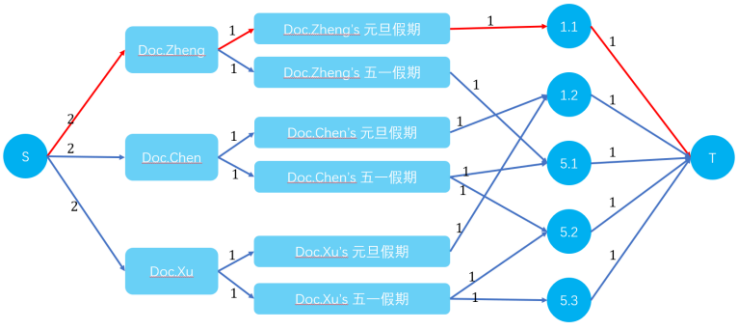


图 5：模型中的某一个流

如果最大流的值与需要值班的日期数目相等，那么该问题至少存在一种可行解，即至少存在一种排班方案；若最大流小于需要值班的日期数目（不可能大于，因为与汇点相连的边的容量只有日期数目那么多），那么该问题无解，即无法找到一种排班方案使得每个日期都有医生值班。

当最大流流量与所需值班日期数目相同时，说明每一个假日所对应的节点连向汇点的那条边都会存在流量，且为 1（因为容量为 1），故表示每一个假日都存在医生值班且满足约束条件。

于是，求解医生的排班方案转化为求解上述模型的一个最大流，接下来的几个小节将重点介绍最大流的求解算法，并对一些重要定理做出证明。

3 Ford-Fulkerson 方法

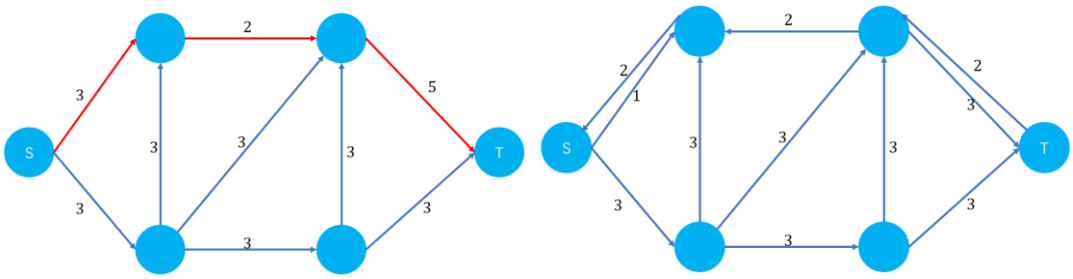
首先介绍最大流算法的基本框架，然后在第 4 节和第 5 节再详细讲述基于该框架的具体算法实现及相关优化。Ford-Fulkerson 方法基于三种重要思想：残存网络、增广路径和流网络的切割，接下来先简单介绍这些概念，再证明 Ford-Fulkerson 方法的核心定理：最大流最小割定理，最后得出 Ford-Fulkerson 方法的思想与流程并对该方法的复杂度上界做出分析。

3.1 残存网络

给定流网络$G(V,E)$和G中的一个流f，残存网络G_f中边的容量定义为：

$$c_f(u,,v)=\begin{cases}c(u,v)-f(u,v), if\,(u,v)\in E\\ f(v,u), if\,(v,u)\in E\\ 0, else\end{cases}$$

如图 6(a)所示，红色边所表示的从源点到汇点的路径为给定的流网络中的一个流，其流量为 2，则该流网络的残存网络如图 6(b)所示。



(a) (b)

图 6：残存网络示意图

可以看到，残存网络中有一些原网络不存在的反向边，这些反向边允许算法将已经发送出去的流量进行回收，流经反向边的流量相当于缩减了原始边的流量，这将允许算法对已经发出的流量进行重新调整。

在残存网络中，可以通过寻找一个新的流对原网络的流进行增广。假设f是G中的一个流，f'是对应的残存网络G_f中的一个流，定义$f\uparrow f'$为流f'对流f的增广，它被定义为一个$V\times V$到$\mathbb{R}$的函数，定义如下：

$$(f\uparrow f')(u,v)=\begin{cases}f(u,v)+f'(u,v)-f'(v,u), if\,(u,v)\in E\\ 0, else\end{cases}$$

该定义的直观解释满足残存网络的定义，因为在残存网络中将流量发送到反向边上等同于在原来的网络中缩减流量，所以将边(u,v)的流量增加$f'(u,v)$，减少$f'(v,u)$。

对于上述定义的$f\uparrow f'$函数，不难证明以下引理成立：

引理 3.1 设$G=(V,E)$为一个流网络，源节点为s，汇点为t，设f为G中的一个流。设G_f为由流f所诱导的G的残存网络，设f'为G_f中的一个流。那么$f\uparrow f'$为G中的一个流，且$|f\uparrow f'|=|f|+|f'|$。

3.2 增广路径

给定流网络$G=(V,E)$和流f，增广路径p是残存网络G_f中一条从源点s到汇点t的简单路径，如图 7 所示路径。称一条增广路径p上能够为每条边增加的流量最大值为p的残存容量，则该容量为：$c_f(p)=\min\{c_f(u,v)|\,(u,v)\in p\}$，也就是路径上所有边残存容量的最小值，并不难理解，因为该路径无法流过比路径上边的容量最小值更大的流量了。如图 7 所示，该路径的残存容量为 3。

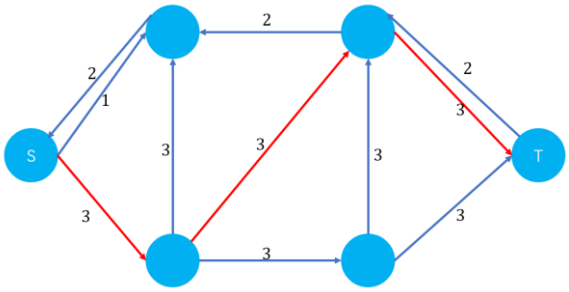


图 7：增广路径

定义完增广路径以及增广路径的残存容量后，不难证明以下引理成立。

引理 3.2 设$G=(V,E)$为一个流网络，设f为G中的一个流，设p为残存网络G_f中的一条增广路径。定义一个函数f_p：$V\times V\to\mathbb{R}$，

$$f_p(u,v)=\begin{cases}c_f(p), if\ (u,v)\in p\\ 0, else\end{cases}$$

则f_p为残存网络G_f中的一个流，其值为$|f_p|=c_f(p)>0$。

根据引理 3.1 与引理 3.2，不难证明以下推论成立。

推论 3.3 设$G=(V,E)$ 一个流网络，设f为G中的一个流，设p为残存网络G_f中的一条增广路径。设f_p为引理 3.2 中所定义的函数，并将f增加f_p的量，则函数$f\uparrow f_p$是图G中的一个流，其值为$|f\uparrow f_p|=|f|+|f_p|>|f|$。

该推论说明了，可以不断在残存网络中寻找增广路径进行增广，以增加原网络中的流的流量。如图 8 所示，将图 7 所示的增广路径增广之后，原网络流量增加（从 2 增加到 5）。

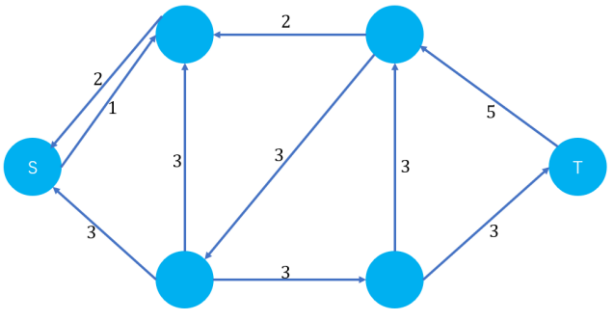


图 8：增广路径进行增广

3.3 切割

流网络$G=(V,E)$中的一个切割(S,T)将节点集合V划分为S和$T=V-S$两个集合，使得$s\in S,t\in T$。若f是一个流，定义切割(S,T)的净流量为：

$$f(S,T)=\sum_{u\in S}\sum_{v\in T}f(u,v)-\sum_{u\in S}\sum_{v\in T}f(v,u)$$

定义切割(S,T)的容量为：

$$c(S,T)=\sum_{u\in S}\sum_{v\in T}c(u,v)$$

图 9 展示了该流网络的一个切割$(S,T),S=\{s,v_1,v_2\},T=\{t,v_3,v_4\}$，该切割的容量为$c(v_1,v_3)+c(v_2,v_3)+c(v_2,v_4)=2+3+3=8$，流量为 0。

流网络中容量最小的切割称为该流网络的最小切割。

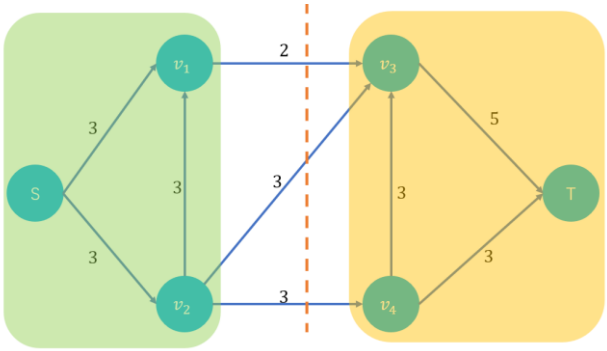


图 9：切割定义

引理 3.4 设f为流网络G中的一个流，该流网络的源节点为s，汇点为t，设(S,T)为流网络G的任意切割，则横跨切割(S,T)的净流量为$f(S,T)=|f|$。

该引理的证明不再赘述，引理 3.4 说明了，流网络中任意切割的净流量等于当前网络中的流。

推论 3.5 流网络G中的任意流f的值不能超过G的任意切割的容量。

推论 3.5 间接说明了，流网络中的最大流的值不能超过该网络最小切割的容量。接下来，使用上述三个概念及相关引理推论证明 Ford-Fulkerson 方法的核心定理：最大流最小切割定理。

3.4 最大流最小切割定理

该定理表述如下：

定理 3.6 设f为流网络$G=(V,E)$中的一个流，该流网络的源节点为s，汇点为t，则下面的条件等价：

1. f是G的一个最大流
2. 残存网络G_f中不包括任何增广路径
3. $|f|=c(S,T)$，其中(S,T)是流网络的某个切割

证明

首先证明$1 => 2$。使用反证法，假设f是G中的一个最大流，但是残存网络G_f中包含一条增广路径p。根据推论 3.3，对f增加流量f_p所形成的新流是G中一个流量严格大于$|f|$的流，这与f是G中最大流矛盾。

然后证明$2 => 3$。假设G_f不包含任何增广路径，那么在残存网络G_f中不存在任何从源节点s到汇点t的路径。定义$S=\{v\in V|$在G_f中存在一条s到v的路径$\}$，$T=V-S$。显然，(S,T)是G的一个切割。现在考虑一对节点$u\in S$，$v\in T$，考虑边(u,v)，若$(u,v)\in E$，则必然有$f(u,v)=c(u,v)$，否则$(u,v)\in E$，将把v归并到S中，与$v\in T$矛盾；若$(v,u)\in E$，则必然有$f(v,u)=0$，否则$(u,v)\in E$，将把v归并到S中，与$v\in T$矛盾；若$(u,v)\notin E$且$(v,u)\notin E$，则$f(u,v)=f(v,u)=0$。故：

$$f(S,T)=\sum_{u\in S}\sum_{v\in T}f(u,v)-\sum_{u\in S}\sum_{v\in T}f(v,u)=\sum_{u\in S}\sum_{v\in T}c(u,v)-0=c(S,T)$$

由引理 3.4 有：$|f|=f(S,T)=c(S,T)$。

最后证明$3 => 1$。由推论 3.5 有，对于所有的切割(S,T)，$|f|\le c(S,T)$，因此，$|f|=c(S,T)$表示f是一个最大流。

该定理说明了，流f是G中的最大流当且仅当G_f中不存在增广路径，根据这一性质可以得出 Ford-Fulkerson 方法的基本思想。

3.5 方法思想

在残存网络G_f中（一开始$G_f = G$），不断寻找增广路径，根据该增广路径的残存容量$c_f(p)$更新路径上的边的流量，从而更新G的流f，直到找不到增广路径为止。通过若干次迭代，算法最终收敛，由最大流最小切割定理可知，在算法结束时，该算法将得到G中的一个最大流。

问题转化为，如何快速找到G_f中的增广路径并进行更新，接下来的一节介绍该方法的两个具体实现，即实例化找增广路径的算法。

3.6 方法流程

以图 10 所示流网络为例子阐述 FF 方法的运行流程。

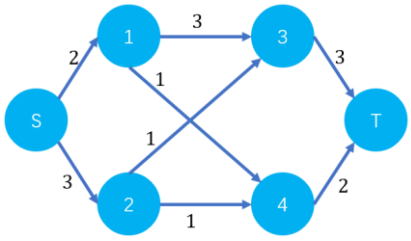


图 10：例子

一开始流量为 0，找到增广路$S \to 1 \to 3 \to T$并进行增广，如图 11 所示，流量增加为 2。

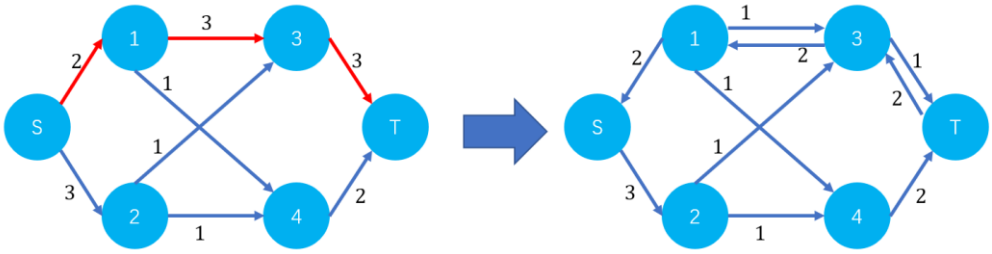


图 11：FF 方法流程（1）

在残存网络中找到增广路$S \to 2 \to 4 \to T$并进行增广，如图 12 所示，流量增加为 3。

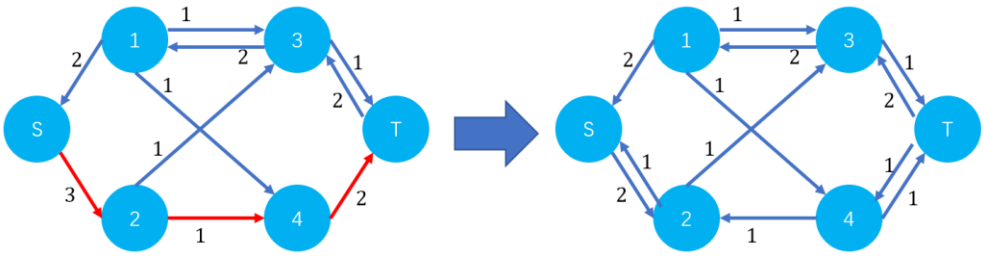


图 12：FF 方法流程（2）

在残存网络中找到增广路$S \to 2 \to 3 \to 1 \to 4 \to T$并进行增广，如图 13 所示，流量增加为 4。增广之后，残存网络中已不存在增广路径，故算法结束，找到最大流流量为 4。

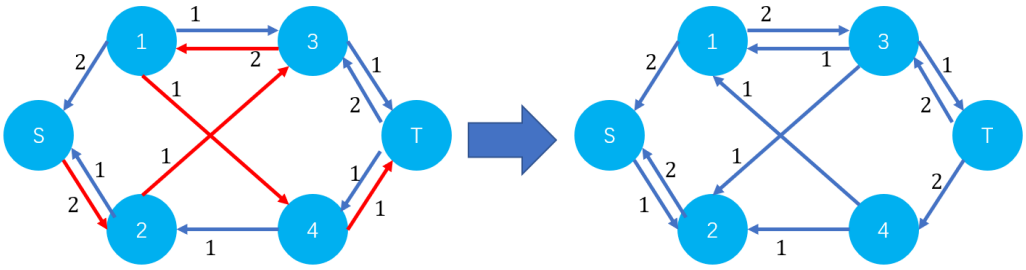


图 13：FF 方法流程（3）

3.7 方法复杂度分析

本节分析一下 FF 方法的时间复杂度上界。该方法的时间复杂度包括找增广路径的代价与寻找增广路径的次数。假设当前流网络为$G=(V,E)$，且$|V|=n,|E|=m$，该流网络的最大流为f^*。

找增广路径时，需要从源节点s开始搜索，直到搜索树中出现汇点t，故时间复杂度为$O(m)$。最坏情况下，寻找增广路的次数为$O(|f^*|)$次，故最坏情况下，FF 方法的时间复杂度为$O(m|f^*|)$。

当$|f^*|$很大时，FF 方法显然不是一个多项式时间复杂度的算法，于是需要对该上界进行优化，在下一节会具体介绍一个使用广度优先搜索寻找增广路径的算法并分析其复杂度上界。

4 Edmonds-Karp 算法

4.1 算法思想

该算法基于 FF 方法的增广路框架，也是不断找增广路进行增广。它确定了找增广路的方式——使用广度优先搜索寻找增广路，若把流网络中每一条边的权值设置为单位距离，那么每一次找出来的增广路都会是当前最短的一条增广路（经过的节点最少），如图 14 所示，一开始流网络中存在两条增广路径，黄色路径与红色路径，此时会选择黄色路径进行增广。

为什么要每次都使用最短的增广路进行增广呢？个人直观的理解就是减少增广路更新的时间，因为更新增广路上每一条边的流量所耗费的时间与增广路长度成正比。在 4.3 节会有更严谨的复杂度证明

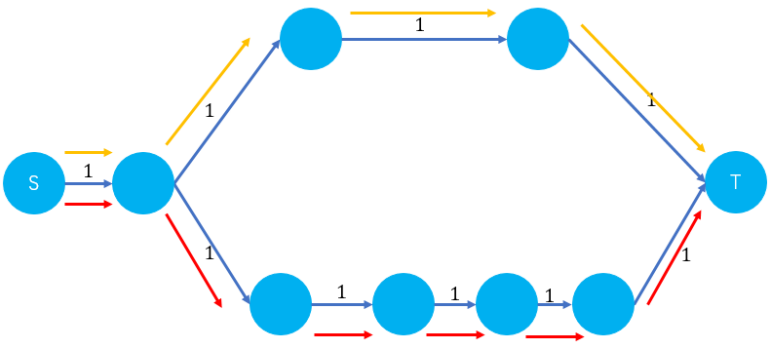


图 14：EK 算法思想

4.2 算法流程

以图 15 所示流网络为例子阐述 EK 算法的运行流程。

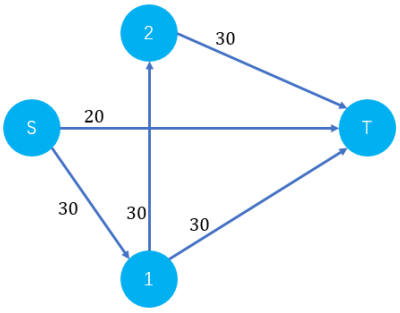


图 15：例子

首先找到当前最短的增广路径：S → T，进行增广，如图 16 所示。

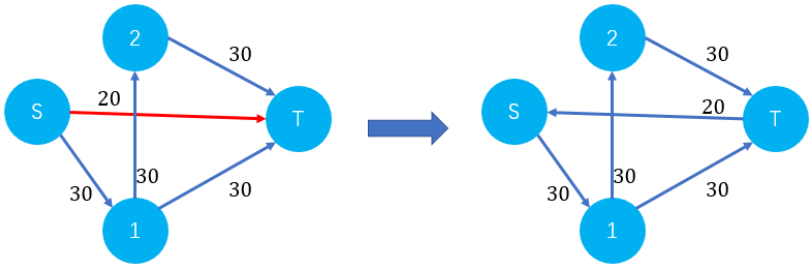


图 16：EK 算法流程（1）

然后找到此时的最短增广路S → 1 → T进行增广，此时流网络中已不存在增广路，算法结束，最大流流量为 50，如图 17 所示。

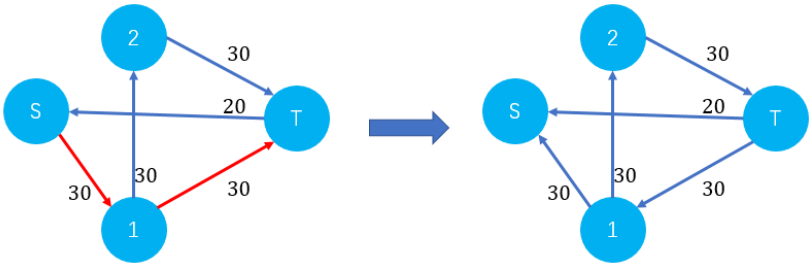


图 17：EK 算法流程（2）

4.3 算法复杂度分析

本节分析一下 EK 算法的时间复杂度上界。该方法的时间复杂度包括找增广路径进行更新的代价与寻找增广路径的次数。假设当前流网络为$G = (V, E)$，且$|V| = n, |E| = m$。

由于在用广度优先搜索寻找增广路径时，每一次寻找增广路径进行更新可以在$O(m)$时间内完成。寻找增广路径的次数也就是算法的迭代次数，该次数由定理 4.1 给出。

定理 4.1 如果 EK 算法运行在源节点为s汇点为t的流网络$G = (V, E)$上，且$|V| = n, |E| = m$，则该算法所执行的流量递增操作的总次数为$O(nm)$。

证明：

在残存网络G_f中，如果一条路径p的残存容量是该条路径上边(u, v)的残存容量，如果$c_f(p) = c_f(u, v)$，则称边(u, v)为增广路径p上的关键边。在沿一条增广路径增加流后，处于

该条路径上的所有关键边都将从残存网络中消失。任何一条增广路上都至少存在一条关键边，不难证明，对于E中的每条边来说，成为关键边的次数最多为$n/2$次。由于一共有$O(m)$对结点可以在一个残存网络中有边彼此连接，故在算法执行过程中，关键边的总数为$O(nm)$。又因为每条增广路径至少有一条关键边，因此定理成立。

于是，整个算法的复杂度上界为$O(nm^2)$。

5 Dinic 算法

Dinic 算法是对 EK 算法进一步的改进，本质上也是使用了 FF 方法的算法框架，寻找增广路进行增广。EK 算法虽然每次都通过最短增广路进行增广，但也存在一定瓶颈，比如说每一次只能对一条增广路进行增广，存在较多的冗余枚举等问题，本节首先介绍 Dinic 算法的基本流程，再针对以上两个 EK 算法存在的问题，提出对应的优化手段。

5.1 Dinic 算法思想

Dinic 算法的基本思想是，先使用广度优先搜索遍历整个流网络，对流网络分层。假设源点s的层数为 0，那么一个点的层数就是它离源点的最近距离，如图 18 所示，该图为图 15 流网络的分层图。

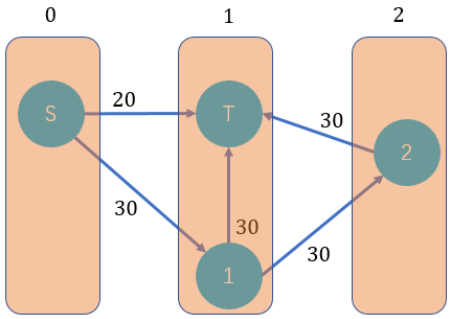


图 18：分层图

将原流网络使用广度优先搜索分层之后，如果不存在到达汇点的增广路，即汇点层数不存在，则可以停止增广。若增广路存在，则按照层数递增的方法寻找增广路进行增广，这样就可以确保找到的增广路是最短的。Dinic 算法使用 dfs 也就是深度优先搜索的方式寻找增广路（如图 19 所示），由于已经用 bfs 提前分好层次了，找增广路使用哪种方式都可以找到最短的增广路，这里使用 dfs 寻找只是为了之后进一步的优化。

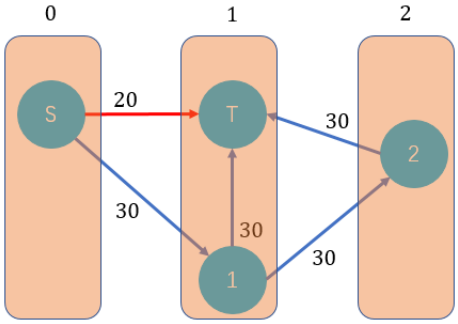


图 19：分层图上找增广路

5.2 多路增广优化

不加优化的 dinic 算法本质上与 EK 算法是一致的，存在如图 20(1)所示例子的问题。由于每一次只增广一条增广路，于是第一次会对增广路S → 1 → 4 → 5 → T进行增广，得到如图 20(2)所示的残存网络。第二次会对增广路S → 1 → 4 → 6 → T进行增广，得到如图 20(3)所示的残存网络。第三次会对增广路S → 1 → 4 → 7 → T进行增广。

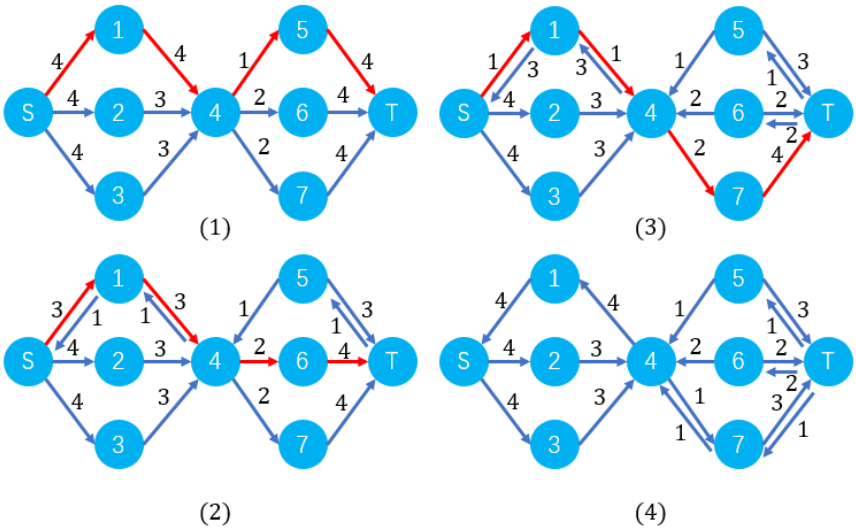


图 20：EK 算法问题

从图 20 中可以看出，三条增广路会重复经过路径S → 1 → 4，这将导致多次更改同一条边的流量信息，造成冗余。多路增广优化，顾名思义就是同时对多条增广路进行增广，减少重复更改相同边的流量信息。可以利用 dinic 算法使用 dfs 寻找增广路径的特点，记录节点当前可能流过的最大流量，向它的后继节点分配剩余流量，直至流量分配完为止，如图 21 所示，第一次寻找增广路时，从源节点S到达节点 4 时可能流过的最大流量为 4，对于节点 4 之后的路径，4 → 5 → T分配了 1 单位流量，流量剩余 3 继续分配给下一条路径；4 → 6 → T分配了 2 单位流量，流量剩余 1 继续分配给下一条路径；4 → 7 → T分配了 1 单位流量，流量分配完毕，此次增广结束。

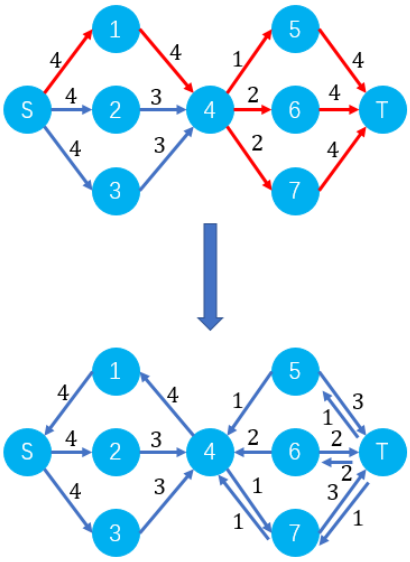


图 21：多路增广改进示例

5.3 当前弧优化

在进行多路增广优化之后，算法还存在着另外一个问题，假设紧接着图 21 所示残存网络继续第二次增广，那么首先增广路经过$S \to 2 \to 4$到达节点 4，此时，算法会按照顺序遍历节点 4 的邻接边判断是否存在容量可以进行增广，于是需要遍历边(4,5)(4,6)两条无效边（因为流量已满），直到(4,7)才找到一条有效边进行增广，如图 22 所示。

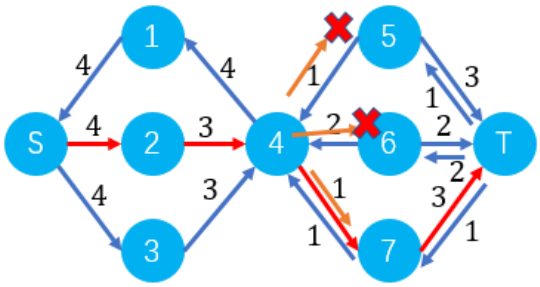


图 22：遍历已满流的边

由于 dinic 算法运行过程中，每一次分层之后的边一旦满流就失效了。于是可以给每个节点的每一条邻接边一个编号，使用额外空间记录每一个节点当前还未满流的第一个弧的编号，节点i当前未满流的第一个弧编号记为$cur[i]$。每一次增广就不需要遍历当前节点的所有邻接边了，直接从当前未满流的第一条边开始分配流量即可，如图 23 所示。

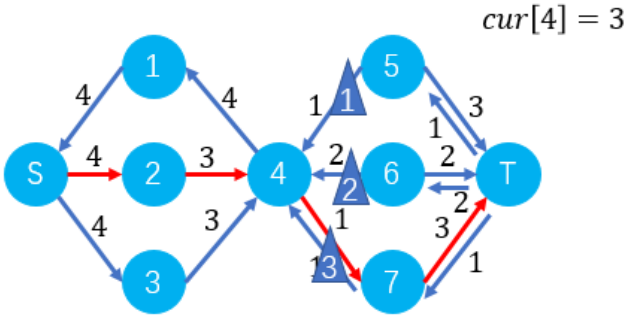


图 23：当前弧优化示例

5.4 算法复杂度分析

对添加了多路增广优化与当前弧优化的 dinic 算法进行时间复杂度上界的分析。

现有流网络$G(V,E)$，$|V| = n, |E| = m$，由于 dinic 算法由 bfs 与 dfs 两个过程组成。首先看 dfs 最多需要执行多少次，由于每个点的当前弧标记最多更新m次，故而每次找增广路最少使得一个点的当前弧标记改变，故 dfs 寻找增广路的时间复杂度上界为$O(nm)$。

再看 bfs 最多需要执行多少次，由于每一次汇点的层数最少增加 1，故最多增加$n-1$次，故 bfs 执行次数的时间复杂度上界为$O(n)$。

于是，优化后 dinic 算法的时间复杂度为$O(n^2m)$。

6 相关实验

6.1 在自拟小数据上测试算法正确性

由于模型本身已经具有正确性，即可行流表示可行方案。故只需要判断最大流的值是否等于假日数目即可。也可以在小数据上输出可行方案进行手动验证。

使用 EK 算法与 Dinic 算法在图 4 所示的小数据上进行方案输出，如图 24 所示，验证算法正确，对应的方案如图 25 所示。

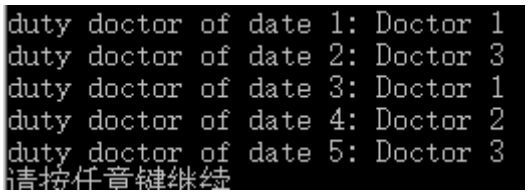


图 24：算法正确性验证

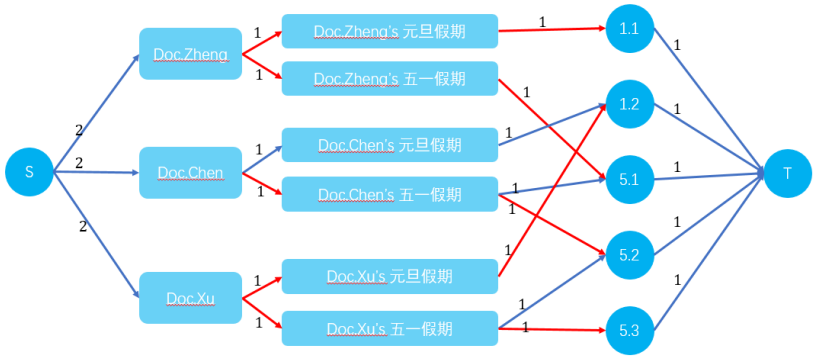


图 25：正确的排班方案

6.2 在随机数据上测试不同算法效率

固定假期数为 20，每个假期的假日数为 5，每个医生的最大值班天数为 5，随机生成数据对不同算法进行测试，取 20 次测试的平均运行时间进行比较，测试结果如表 1 所示。可以看到，在数据范围较小的情况下，多路增广优化的效果较为显著，但随着数据规模增大，多路增广优化的效果持续下降，当医生数量达到 800 时，多路增广优化甚至比 EK 算法效率还低，这是因为模型的特别性，由于许多节点可分配流量都为 1，医生数量多时，多路增广使用的递归方法所产生的影响比优化的力度大，故效率会下降。可以看到，dinic 算法的当前弧优化无论在哪个数据规模下都有较为优秀的表现。

表 1：不同算法在不同医生数量下的运行效率变化

DoctorNumber	200	400	600	800	1000
EK(ms)	89.37	177.27	262.23	366.44	443.79
dinic+Multichannel(ms)	23.50	85.96	199.19	377.65	565.35
dinic+CurrentArc(ms)	3.88	6.40	11.18	14.83	17.61

四、实验心得

本次实验对给定实际问题进行分析并建立流网络模型，之后给出解决方案，对具体算法进行详细叙述并进行相关优化，最后对算法正确性做出实验验证并对算法运行效率进行探究。

实验过程中，遇到的一个问题就是把模型转为代码的问题，只要一步一步慢慢分析，剖析模型的底层规律即可。

五、附件说明

- **code（实验所编写代码）**
  - **dinic.cpp（Dinic 算法）**
  - **EK.cpp（EK 算法）**
  - **FF.cpp（FF 方法的 dfs 实现）**
  - **generatedata.cpp（随机数据生成器）**
- **result（实验结果数据表）**
- **PPT（演讲 PPT）**
- **PDF（实验报告 PDF 版本，更好的阅读体验）**

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。