最小割模型在信息学竞赛中的应用
Applications of Minimum Cut Model in Informatics

胡伯涛 Amber
[ADN.cn]
福州第一中学 Fuzhou No.1 Middle School
hupo001[at]gmail[dot]com

[摘要]

本文对最小割模型的定义和性质，以及其相关扩展知识进行了研究。其中着重对最小割模型在以下四个方面的应用展开研究：1. 基于定义的直接应用；2. 最大权闭合图；3. 最大密度子图；4. 二分图的最小点权覆盖集和最大点权独立集。展现与剖析了最小割模型应用的巧妙构图方法和独特思维方式，并对这一类应用的通用方法与技巧给予总结。

[关键字]

网络流，最大流，最小割，最大权闭合图，最大密度子图，二分图最小点权覆盖集，二分图最大点权独立集

[Abstract]

The purpose of the thesis is to research the definition, properties, and correlated extended knowledge of minimum cut model.

The thesis sets focus on researching that is in four aspects: 1. the application based on the minimum cut model; 2. the maximum weight closure of a graph; 3. the maximum density sub graph; 4. the minimum weight vertex covering set and the maximum weight vertex independent set in the bipartite graph. It shows and analyzes the construction methods and thinking modes of the minimum cut model. Finally, it summarizes the methods and skills in the applications of the minimum cut model.

[Key Words]

Network Flow, Maximum Flow, Minimum Cut, Maximum Weight Closure of a Graph, Finding a Maximum Density Sub Graph, Minimum Weight Vertex Covering Set and Maximum Weight Vertex Independent Set , Bipartite Graph

[目录]

0. 前言 Preface

最小割是最大流的对偶问题。但在实际建模过程中，它不如最大流来的直观，模型也往往隐蔽得很深，不容易找到构图方法。本文着重研究最小割模型的实际应用，展示了最小割模型应用的巧妙构图方法与独特思维方式，并对于这一类模型建立的通用方法与技巧给予总结。

第 1 节，介绍一些基本的网络流与分数规划的知识，为后文的论述提供依据。特别是分数规划部分，对参考文献[LiuH04]上的 0-1 分数规划进行了拓展，拓展到一般的分数规划形式。

第 2 节，以 2 个较简单例题的分析，展示基于定义直接应用最小割的方法与技巧。

第 3 节，介绍最大权闭合图模型，以及用最小割模型来解决的方法。关于构图的证明是本节重点。

第 4 节，介绍最大密度子图模型。先将其转化为最大权闭合图模型解决，又重新提出了一个利用最小割模型解决的新方法，更好地解决了该问题。并向带点权的图和点边均带权的图进行了拓展，很好地解决了**最大获利(profit)**问题。**本节是本文的亮点，也是研究的重点。**

第 5 节，介绍二分图的最小点权覆盖集与最大点权独立集，以及利用最小割模型解决的方法。构造比较简单，重点是两个例题的化归过程。

第 6 节，总结利用最小割模型解决问题的一些基本方法与思维过程。

本文定位是重在思路的分析，总结利用最小割模型解决问题通用的思维方向，而不是纯粹的算法介绍，所以本文过半的篇幅用来描述思维转化的过程。具体的分析基本上是以波利亚的《怎样解题表》[①]中所描述的步骤作为主线。在每个分析后，都会有简洁凝练的证明过程来总结分析的思维过程。对于那些急于得到结论的读者，直接阅读证明过程即可，但是请记住，**分析过程才是本文重点**。

1. 预备知识 Preliminaries

1.1. 网络与流 Network and Flow

流网络（flow network）（简称网络）$G=(V,E)$ 是一个有向图，其中每条边 $\langle u,v\rangle \in E$ 均有一非负容量 $c(u,v)\geq 0$，规定：若 $\langle u,v\rangle \notin E$，则 $c(u,v)=0$。网络中有两个特殊点：源 s 与汇 t。

流网络 G 的**流**（flow）是一个实值函数 f：$V\times V\to R$，且满足下列三个性质：

[①] 出自参考文献[Pol57]，数学教育家 G. Polya 的 *How to solve it*。详见附录 8.3 节，怎样解题表。

(1) **容量限制**：对于 $\forall u,v \in V$ ，要求 $f(u,v) \le c(u,v)$ 。

(2) **反对称性**：对于 $\forall u,v \in V$ ，要求 $f(u,v) = -f(v,u)$ 。

(3) **流守恒性**：对于 $\forall u \in V - \{s,t\}$ ，要求 $\sum_{v \in V} f(u,v) = 0$ 。

称 $f(u,v)$ 为从 u 到 v 的流。流 f 的值定义为：

$$|f| = \sum_{v \in V} f(s,v) \tag{1.1}$$

一个网络中的**最大流**（maximum flow）就是指该网络中流值最大的流。

为了方便起见，本文在函数的求和上，采用**隐含求和记号**：其中任何一个自变量可以是一个点集，对该集合的所有元素作为自变量的情况求和。例如：

$$f(X,Y) = \sum_{u \in X} \sum_{v \in Y} f(u,v)$$

下面不加证明地给出一个简单的引理，它给出了关于流的一些恒等式，这对后文的推导和计算流的公式很重要。

引理1.1 （流的基本运算）设 f 是流网络 $G = (V,E)$ 中的一个流。那么下列等式成立：

(1) 对于 $\forall X \subseteq V$ ，有 $f(X,X) = 0$

(2) 对于 $\forall X,Y \subseteq V$ ，有 $f(X,Y) = -f(Y,X)$

(3) 对于 $\forall X,Y,Z \subseteq V$ ，其中 $X \cap Y = \varnothing$ ，有 $f(X \cup Y,Z) = f(X,Z) + f(Y,Z)$

■

1.2. 残留网络与增广路径 Residual Network and Augmenting Path

残留网络，增广路径与割这三个概念是构成重要的最大流最小割定理（定理 1.6）的基本概念，该定理巧妙运用网络的最小割来描述最大流的值。

对于流网络 $G = (V,E)$ ，设流 f 为 G 中的流。**残留网络**(residual network)直观上讲是由还可以容纳更多的流的边所组成。对于 G 中的每条边 $\langle u,v \rangle \in E$ ，可以定义**残留容量**(residual capacity)表示在不超过容量限制的条件下，可以通过的额外的网络流量：

$$c_f(u,v) = c(u,v) - f(u,v) \tag{1.2}$$

下面是残留网络的形式化定义：给定网络 $G = (V,E)$ 与流 f ，残留网络为 $G_f = (V,E_f)$ ，

其中边集

$$E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$$

事实上，残留网络也是一个流网络，容量由 c_f 给出。下面的引理，给出了残留网络与原网络的关系，也为提出增广路径提供了前提。

引理1.2 （残留网络与原网络的关系）设 f 是流网络 $G=(V,E)$ 中的一个流， f' 是其残留网络 G_f 的一个流，则 $f+f'$ 仍是网络 G 的一个流。

■

该定理的证明需从网络流定义的三个性质分别来证明，这里从略。

增广路径(augmenting path) p 为残留网络 G_f 上源 s 到汇 t 的一条简单路径。该路径的残留容量，为可以沿该路径增加的最多额外流量：

$$c_f(p) = \min\{c_f(u,v) \mid \langle u,v\rangle \in p\}$$

其值严格大于 0。

引理1.3（增广路径可增广性）定义流 f_p 为

$$f_p(u,v) = \begin{cases} c_f(p) & \langle u,v\rangle \in p \\ -c_f(p) & \langle v,u\rangle \in p \\ 0 & else \end{cases}$$

给定网络 $G=(V,E)$ 与流 f ，则 $f+f_p$ 仍是网络 G 的一个流。

■

该定理的证明思路仍是从网络流定义的三个性质出发，证明 f_p 是残留网络 G_f 上的一个流，再根据引理 1.2，便可得证。这里从略。

1.3. 最大流与最小割 Maximum Flow and Minimum Cut

下面引入本文讨论的核心——割。

流网络 $G=(V,E)$ 的**割**（cut）$[S,T]$ 将点集 V 划分为 S 和 T （ $T=V-S$ ）两个部分，使得源 $s \in S$ 且汇 $t \in T$ 。符号 $[S,T]$ 代表一个边集合 $\{\langle u,v\rangle \mid \langle u,v\rangle \in E, u \in S, v \in T\}$ 。穿过割 $[S,T]$ 的**净流**（net flow）定义为 $f(S,T)$ ，割 $[S,T]$ 的**容量**（capacity）定义为 $c(S,T)$ ，一般记为 $c[S,T]$ 。一个网络的**最小割**（minimum cut）也就是该网络中容量最小的割。

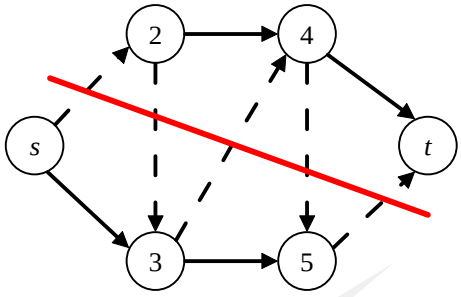

图 1.1 最小割的例子

图 1.1 给出了一个最小割的例子，红线下的点组成S，其他点组成T，则割$[S,T]=\{\langle 1,2\rangle,\langle 3,4\rangle,\langle 5,6\rangle\}$，而割$[T,S]=\{\langle 2,3\rangle,\langle 4,5\rangle\}$。注意，边$\langle 2,3\rangle$，$\langle 4,5\rangle$的流值算在净流$f(S,T)$内（是个负值），但它们的容量并没有算在割$[S,T]$的容量内。

引理1.4 （割与流的关系）在一个流网络$G=(V,E)$中，设其任意一个流为f，且$[S,T]$为G一个割。则通过割的净流为$f(S,T)=|f|$。

证明：

$$\begin{aligned} f(S,T) &= f(S,V)-f(S,S) && \text{根据引理(1.1)(3)} \\ &= f(S,V) && \text{根据引理(1.1)(1)} \\ &= f(s,V)+f(S-\{s\},V) && \text{根据引理(1.1)(3)} \\ &= f(s,V)=|f| && \text{根据流守恒性，} f(S-\{s\},V)=0 \end{aligned}$$

■

推论1.5 （对偶问题的性质）在一个流网络$G=(V,E)$中，设其任意一个流为f，任意一个割为$[S,T]$，必有$|f|\le c[S,T]$。

证明：由引理 1.3 和容量限制，有

$$|f|=f(S,T)=\sum_{u\in S}\sum_{v\in T}f(u,v)\le\sum_{u\in S}\sum_{v\in T}c(u,v)=c[S,T]$$

■

推论 1.5 反映了网络的最大流必定不超过此网络最小割的容量。实际上，最小割问题是最大流的**对偶问题**（duality）[①]，而对偶问题都具有类似推论 1.5 的性质。

定理1.6 （最大流最小割定理）如果f是具有源s和汇t的流网络$G=(V,E)$中的一个流，则下列条件是等价的：

(1) f是G的一个最大流

(2) 残留网络G_f不包含增广路径

[①] 关于流与割的对偶关系，在参考文献[AMO93]中有详细介绍。

(3) 对 G 的某个割 $[S,T]$，有 $|f|=c[S,T]$

证明：

(1)$\Rightarrow$(2)：反证法。设 f 是 G 的一个最大流，但残留网络 G_f 包含一条增广路径 p。由引理 1.3，流的和 $f+f_p$ 仍为 G 的一个流，其值严格大于 $|f|$。这与条件矛盾。

(2)$\Rightarrow$(3)：残留网络 G_f 不包含增广路径，即 G_f 中不存在 s 到 t 的路径。构造性的令：$S=\{v\in V \mid \exists p_{s,v}\in G_f\}$，其中 $\exists p_{s,v}\in G_f$ 表示在 G_f 中存在一条 s 到 v 的通路；且 $T=V-S$。则划分 $[S,T]$ 是一个割：$s\in S$，由于 G_f 中不存在 s 到 t 的路径，所以 $t\notin S$。对于 $\forall u\in S,\forall v\in T$，有 $f(u,v)=c(u,v)$，否则 $\exists\langle u,v\rangle\in E_f$，$v$ 就属于集合 S。由引理 1.4，$|f|=f(S,T)=c[S,T]$。

(3)$\Rightarrow$(1)：由推论 1.5 可知，对于所有割 $[S,T]$，都有 $|f|\le c[S,T]$，因此条件 $|f|=c[S,T]$ 说明 f 是一个最大流。

■

1.4. 最小割的算法 Algorithm for Minimum Cut

由于最大流最小割定理(定理 1.6)中，(1)与(3)等价，即最大流的流值等于最小割容量。并且在最大流最小割定理的(2)$\Rightarrow$(3)部分的证明中，构造性地得到了一个一一对应的割 $[S,T]$。其中点集 S 表示在 G_f 中与 s 连通的点集；$T=V-S$。所以在问题的实现上分两步，先求得最大流（在1.5 节有简要介绍）；再在得到最大流 f 后的残留网络 G_f 中，从 s 开始深度优先遍历（DFS），所有被遍历到的点，即构成点集 S。

注意，虽然最小割 $[S,T]$ 中的边都是满流边，但**满流边不一定都是最小割中的边**。在某些特殊图中，很多初学者容易犯错，认为不用 DFS，就可以直接得出割。下面举一个二分图的例子。

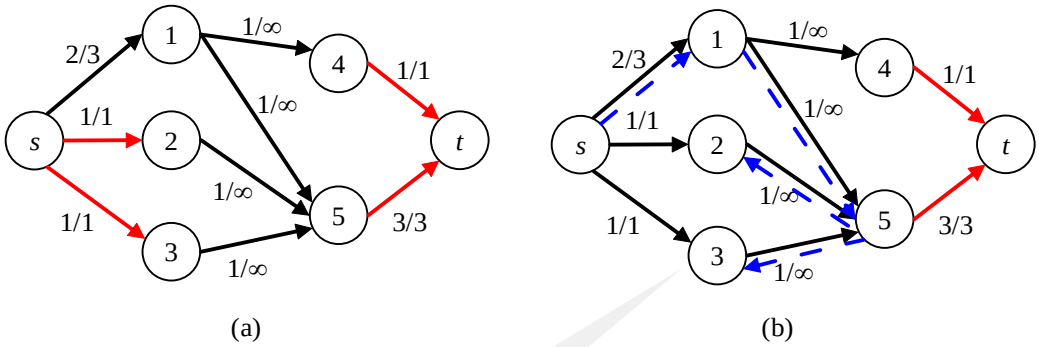

图 1.2 求最小割的二分图例子

图 1.2(a) 给出了一个基于二分图构造的流网络。由于从 X 部到 Y 部都是容量均为正无限的边，都不可能是最小割中的边，有人便会错误地认为与源或汇关联的满流边便组成了最小割（图 1.2(a)的红色边）。然而实际上，在该网络的残留网络中，结点 2 与 3 应该与源 s 是连通的（图 1.2(b)的蓝色路径），所以最小割应该是图 1.2(b)中的红色边。

1.5. 最大流的算法 Algorithm for Maximum Flow

由于本文只研究最小割的应用，对为得到最小割而具体使用的最大流算法本身不进行深入的讨论。下面列出一些常用的适用于**一般图**的最大流算法。有兴趣的读者可以在参考文献[AMO93]，[CLRS01]中找到相关的知识，此外，参考文献[ZCW03]中有对最大流算法历史的简要回顾。

给出流网络 $G=(V,E)$。简记 $n=|V|$，$m=|E|$。设 U 为最大的边容量值。后文不再涉及具体的最大流复杂度，通常以 $O(\text{MaxFlow}(G))$，$O(\text{MaxFlow}(n,m))$ 或 $O(\text{MaxFlow}(n))$ 代替。

算法名称	复杂度	概要
增广路方法 Augmenting path method (Ford Fulkerson method)		
一般增广路算法 Labeling algorithm	$O(nmU)$	在残留网络中，每次任意找一条增广路径增广。
容量缩放增广路算法 Capacity scaling algorithm	$O(nm\log U)$	在残留网络中，每次找一条有**最大可增广容量和**的增广路径增广，即残留网络中源到汇的最长路。
最短增广路算法 Shortest augmenting path algorithm (Edmonds Karp algorithm)	$O(nm^2)$	在残留网络中，每次找一条**含结点数最少**的增广路增广，即残留网络中源到汇的 BFS 路径。

连续最短增广路算法[③] Successive shortest augmenting path algorithm (Dinic algorithm)	$O(n^2m)$	在 Edmonds Karp 的基础上改造。在每次 BFS 找增广路时，记录每个点的距离标号。在距离标号的所构成的最短路图上，不断地 DFS 找增广路。即**一次标号多次增广**，以提高速度。
预流推进方法 Preflow-push method		
一般预流推进算法 Generic preflow-push algorithm	$O(n^2m)$	维护一个**预流**，不断地对活跃结点执行 push 操作或 relabel 操作来调整这个预流，直到不能操作。
先进先出预流推进算法 FIFO preflow-push algorithm	$O(n^3)$	以**先进先出队列**维护活跃结点。
最高标号预流推进算法 Highest-label preflow-push algorithm (Relabel-to-Front algorithm)	$O(n^2\sqrt{m})$	每次检查具有**最高标号**的活跃结点。

增广路方法的复杂度是通过估计增广次数的上界得到的。对于实际应用中的网络，增广次数往往很少，所以使用范围还是很广的，实用性强。预流推进方法看似比增广路方法在复杂度上快很多，然而实际上，预流推进方法的复杂度的上界是比较紧的。对于一些稀疏图，预流推进方法的实际效果往往不如增广路方法。

1.6. 分数规划 Fractional Programming

先给出**分数规划**（fractional programming）的一般形式：

$$\text{Minimize} \quad \lambda = f(\mathbf{x}) = \frac{a(\mathbf{x})}{b(\mathbf{x})} \quad (\mathbf{x} \in S)$$

$$s.t. \quad \forall \mathbf{x} \in S, \quad b(\mathbf{x}) > 0$$

其中，解向量 $\mathbf{x}$ 在解空间 S 内，$a(\mathbf{x})$ 与 $b(\mathbf{x})$ 都是**连续的实值函数**。

解决分数规划问题的一般方法是分析其对偶问题，但更加实用的方法是对其进行**参数搜索**(parametric search)，即对答案进行猜测，再验证该猜测值的最优性，将优化问题转化为判定性问题或其他的优化问题。由于分数规划模型的特殊性，使得能够构造另外一个由猜测值 λ 作为自变量的相关问题，且该问题的解满足一定的单调性，或其他的可以减小参数搜索范围的性质[④]，从而**逼近**答案。

假设 $\lambda^* = f(\mathbf{x}^*)$ 为该规划的最优解，有：

[③] 该连续最短增广路算法与求最小费用流的连续最短增广路算法同名。
[④] 比如 Dinkelbach 算法，每次直接把上次子问题的解向量代入原问题的表达式，算出下一个迭代式的猜测值。详见参考文献[Dink67]，[MSS92]。

$$\lambda^* = f(\mathbf{x}^*) = \frac{a(\mathbf{x}^*)}{b(\mathbf{x}^*)}$$
$$\Rightarrow \quad \lambda^* \cdot b(\mathbf{x}^*) = a(\mathbf{x}^*)$$
$$\Rightarrow \quad 0 = a(\mathbf{x}^*) - \lambda^* \cdot b(\mathbf{x}^*)$$

由上面的形式构造一个新函数 $g(\lambda)$：

$$g(\lambda) = \min_{\mathbf{x}\in S}\{(a(\mathbf{x}) - \lambda \cdot b(\mathbf{x})\} \tag{1.3}$$

这个新函数是一个非分式的规划。先来挖掘函数 $g(\lambda)$ 本身的性质。

性质**1.7**（单调性） $g(\lambda)$ 是一个严格递减函数，即对于 $\lambda_1 < \lambda_2$，一定有 $g(\lambda_1) > g(\lambda_2)$。

证明：设解向量 $\mathbf{x}_1$ 最小化了 $g(\lambda_1)$。则

$$\begin{aligned} g(\lambda_1) &= \min_{\mathbf{x}\in S}\{(a(\mathbf{x}) - \lambda_1 \cdot b(\mathbf{x})\} \\ &= a(\mathbf{x}_1) - \lambda_1 \cdot b(\mathbf{x}_1) \\ &> a(\mathbf{x}_1) - \lambda_2 \cdot b(\mathbf{x}_1) \\ &\geq \min_{\mathbf{x}\in S}\{(a(\mathbf{x}) - \lambda_2 \cdot b(\mathbf{x})\} = g(\lambda_2) \end{aligned}$$

最后一步表示，$g(\lambda_1)$ 上最小解 $\mathbf{x}_1$ 代入 $g(\lambda_2)$ 后，不一定还是最小解，甚至有更小解。

■

有了单调性，就意味着可以采用二分搜索的方法逼近答案。但还不知道目标是什么，下面考察构造出的新函数与原目标函数的最优解关系。

定理1.8 （Dinkelbach定理[5]）设 λ^* 为原规划的最优解，则 $g(\lambda) = 0$ 当且仅当 $\lambda = \lambda^*$。

证明：

(1) 先证必要性 $\lambda = \lambda^* \Rightarrow g(\lambda) = 0$：设 $\lambda^* = f(\mathbf{x}^*)$ 为原规划的最优解，则 $g(\lambda^*) = 0$。

对于 $\forall \mathbf{x} \in S$，都不会比 $\mathbf{x}^*$ 优：

$$\lambda^* = \frac{a(\mathbf{x}^*)}{b(\mathbf{x}^*)} \leq \frac{a(\mathbf{x})}{b(\mathbf{x})} \quad \Rightarrow \quad a(\mathbf{x}) - \lambda^* \cdot b(\mathbf{x}) \geq 0$$

然而 $\mathbf{x}^*$ 可以取到这个下限。

$$\lambda^* = \frac{a(\mathbf{x}^*)}{b(\mathbf{x}^*)} \quad \Rightarrow \quad a(\mathbf{x}^*) - \lambda^* \cdot b(\mathbf{x}^*) = 0$$

[5] 出自参考文献[Dink67]。

故 $g(\lambda^*)$ 的最小值由 $\mathbf{x}^*$ 确定，$g(\lambda^*)=0$。

(2) 再证充分性 $g(\lambda)=0 \Rightarrow \lambda=\lambda^*$：若存在一个解 $\mathbf{x}$，使得 $g(\lambda)=0$，则 $\lambda=f(\mathbf{x})$ 为原规划的最优解。

反证法。反设存在一个解 $\lambda'=f(\mathbf{x}')$，它是比 $\lambda=f(\mathbf{x})$ 更优的解。

$$\lambda'=\frac{a(\mathbf{x}')}{b(\mathbf{x}')}<\lambda$$
$$\Rightarrow \quad a(\mathbf{x}')-\lambda\cdot b(\mathbf{x}')<0$$

这时 $\mathbf{x}'$ 应该使得 $g(\lambda)<0$，这与 $g(\lambda)=0$ 矛盾。

■

由性质 1.7 与定理 1.8，很容易得到下面的推论。

推论1.9 设 λ^* 为该规划的最优解，则

$$\begin{cases} g(\lambda)=0 \Leftrightarrow \lambda=\lambda^* \\ g(\lambda)<0 \Leftrightarrow \lambda>\lambda^* \\ g(\lambda)>0 \Leftrightarrow \lambda<\lambda^* \end{cases}$$

■

由推论 1.9，就可以对最优解 λ^* 进行二分查找逼近。每次查找到一个猜测值 λ，需计算 $g(\lambda)$ 的值，而 $g(\lambda)$ 是一个非分数规划，将原问题化简了，便可以设计出其他有效的算法解决这个规划。算法复杂度是二分迭代次数与每次解决 $g(\lambda)$ 的复杂度的积。

以上分析是针对最小化目标函数的分数规划，实际上对于最大化目标函数也一样适用。

分数规划的一个特例是 **0-1 分数规划**(0-1 fractional programming)[⑥]，就是其解向量 $\mathbf{x}$ 满足 $\forall x_i \in \{0,1\}$（这就是所谓的 0-1）。形式化定义如下：

$$\text{Minimize} \quad \lambda=f(\mathbf{x})=\frac{\sum_i a_i x_i}{\sum_i b_i x_i}=\frac{\mathbf{a}\cdot\mathbf{x}}{\mathbf{b}\cdot\mathbf{x}} \qquad (\mathbf{x}\in\{0,1\}^n)$$

$$s.t. \qquad \mathbf{b}\cdot\mathbf{x}>0$$

并且对解向量 $\mathbf{x}$ 可能还有其他的组合限制。它的应用范围很广，经典例题是**最优比率生成树**（the optimum ratio spanning tree）[⑦]。

[⑥] 0-1 分数规划的相关知识，在参考文献[MSS92]中有详细介绍。
[⑦] 详见参考文献[LiuH04]， 303-304 页。

2. 基于定义的直接应用 Applications based on the Definition

2.1. 引入 Introduction

在这类基于最小割定义的直接应用中，最小割模型的建立往往比较简单，但常常又和一些新颖的知识融合在一起。这样一来便不能直接地套用模型，而是需要使用一些知识与技巧，逐步将问题化归到最小割模型，使其成为某些主算法的子过程。

这类应用比较冗杂，使得本节并没有太多通用的理论性介绍，一些特别的知识与技巧在具体的题目分析中才能得以体现。

2.2. 应用 Application

2.2.1. 网络战争 Network Wars[8]

描述 Description

给出一个带权无向图 $G=(V,E)$，每条边 e 有一个权 w_e。求将点 s 和点 t 分开的一个边割集 C，使得该割集的平均边权最小，即最小化：

$$\frac{\sum_{e\in C} w_e}{|C|} \tag{2.1}$$

解答 Solution

先尝试着用更一般的形式重新叙述本问题。设向量 $\mathbf{w}$ 表示边的权值，令向量 $\mathbf{c}=(1,1,...,1)$ 表示选边的代价，于是原问题等价为：

$$\text{Minimize}\quad \lambda = f(\mathbf{x}) = \frac{\sum_{e\in E} w_e x_e}{\sum_{e\in E} 1\cdot x_e} = \frac{\mathbf{w}\cdot\mathbf{x}}{\mathbf{c}\cdot\mathbf{x}}$$

其中，$\mathbf{x}$ 表示一个解向量，$x_e\in\{0,1\}$，即对于每条边都有选与不选两种决策，并且选出的边集组成一个 $s-t$ 边割集。形式化地，若 $x_e=1$，则 $e\in C$；$x_e=0$，则 $e\notin C$。

联系已有的知识，这是一个 0-1 分数规划。在1.6 节中已经给出了这类规划的普遍转化方法：构造一个新函数

[8] 题目来源：Zhejiang University Online Judge - Andrew Stankevich's Contest #8 - 2676 Network Wars

$$g(\lambda)=\min_{\mathbf{x}}\{(\mathbf{w}-\lambda\mathbf{c})\cdot\mathbf{x}\}$$

即对每条边$\forall e\in E$，进行重赋权：$w_e'=w_e-\lambda\cdot c_e=w_e-\lambda$。$g(\lambda)$便是在这个重新赋权的图上，求一个最小容量的$s-t$边割集。请注意一些细节：若$w_e'<0$，又由于目标函数是加和取最小的操作，则该边必然是在边割集内。对于剩下的所有$w_e'\geq 0$的边，直接利用最小割模型求出$s-t$割即可。

设$\lambda^*=f(\mathbf{x}^*)$为原规划的最优解，由推论 1.9 知：

$$\begin{cases}g(\lambda)=0\Leftrightarrow\lambda=\lambda^*\\ g(\lambda)<0\Leftrightarrow\lambda>\lambda^*\\ g(\lambda)>0\Leftrightarrow\lambda<\lambda^*\end{cases}$$

于是主算法便是对最优解λ^*的二分查找，每次查找用最小割模型求$g(\lambda)$来判定，进而缩小查找范围，直到满足精度要求。设二分查找次数为k。则该算法复杂度为：$O(k\cdot\mathrm{MaxFlow}(N))$。

2.2.2. 最优标号 Optimal Marks[9]

描述 Description

给出一个无向图$G=(V,E)$，每个点v以一个**有界非负**整数作为标号l_v，每条边$e=(u,v)$的权w_e定义为该边的两个端点的标号的异或值，即$w_e=l_u\ \mathrm{xor}\ l_v$。现已知其中**部分点**的标号，求使得该图的总边权和最小的标号赋值。即最小化：

$$\sum_{e\in E}w_e=\sum_{(u,v)\in E}(l_u\ \mathrm{xor}\ l_v)\tag{2.2}$$

解答 Solution

先规范化问题，不妨设所有点一定和已知标号的点连通。否则，令那些不与已知标号点连通的点的标号为 0，这些点间的边权也不必计入目标函数（边权都为 0）。

在问题的优化式子中，发现操作 xor 不好处理，难以转化为一些基本运算。自然地，先考察操作 xor 的定义：$a\ \mathrm{xor}\ b=\sum_{i=0}^{\infty}2^i\cdot(a_i\ \mathrm{xor}\ b_i)$，其中$a_i$，$b_i$分别表示$a$，$b$的二进制表达式第$i$位的值。由于 xor 是按二进制位运算的，所以各个二进制位间是互不影响的。而优化目

[9] 题目来源：Sphere Online Judge - 839 Optimal Marks 作者：Guo Huayang

标的式子是一个和式，可以将每位的公共系数提出来，按位加和：

$$\sum_{(u,v)\in E}\left(l_u \text{ xor } l_v\right)=\sum_{(u,v)\in E}\sum_{i=0}^{\infty}2^i\cdot\left(l_{u,i} \text{ xor } l_{v,i}\right)=\sum_{i=0}^{\infty}2^i\cdot\sum_{(u,v)\in E}\left(l_{u,i} \text{ xor } l_{v,i}\right)$$

于是，主算法就是依次处理各个二进制位，而将子问题化归为：

$$\sum_{e\in E}w_e=\sum_{(u,v)\in E}\left(l_u \text{ xor } l_v\right)$$

该优化式子不变，但是限制条件加强了：对于$\forall u\in V$，$l_u\in\{0,1\}$。即对于每个点，都只有两种取值，可以看成是要**将点集划分成两类**。在这种分类思想的指导下，重新考察操作 xor 的意义：对于边的两个端点，若它们同类则边权无值；若它们异类则边权有值 1。

分析到这步，整理一下关于本题的一些已知信息：点分两类；异类间的关系计入目标函数；优化目标是最小化。此时，似乎产生了一个需要进行合理类比的念头——是否存在某个已知模型具备与本题相似的性质？对，那就是最小割模型，它将点集分为两类——S 和 T；割的容量是 S 和 T 间的边容量和；优化目标也是最小化。下面将以上的灵感，形式化地表达如下：

将原问题转化为网络 $N=(V_N,E_N)$ 的最小割问题。在原图点集的基础上增加源 s 和汇 t；将原图每条无向边 $(u,v)\in E$ 替换为两条容量分别为 $c(u,v)=1$，$c(v,u)=1$ 的有向边 $\langle u,v\rangle$，$\langle v,u\rangle\in E_N$；连接源 s 到原图每个已经具有标号 0 的点 v $(l_v=0)$，即增加有向边 $\langle s,v\rangle\in E_N$，由于该边不应计入目标函数，令容量为 $c(s,v)=\infty$；连接原图每个已经具有标号 1 的点 v $(l_v=1)$ 到汇 t，即增加有向边 $\langle v,t\rangle\in E_N$，同样地，令容量为 $c(v,t)=\infty$。更形式化地，有：

$$V_N=V\cup\{s,t\}$$
$$E_N=\{\langle u,v\rangle\mid(u,v)\in E\}\cup\{\langle s,v\rangle\mid v\in V,l_v=0\}\cup\{\langle v,t\rangle\mid v\in V,l_v=1\}$$
$$\begin{cases}c(u,v)=1 & (u,v)\in E\\ c(s,v)=\infty & l_v=0\\ c(v,t)=\infty & l_v=1\end{cases}$$

定理2.1 （最优性）以上描述的流网络 N 的最小割 $[S,T]$ 容量是原问题的最优解。

■

设最大的标号为 U，则该算法的复杂度为 $O(\log(U)\cdot\text{MaxFlow}(N))$。

3. 最大权闭合图 Maximum Weight Closure of a Graph

3.1. 引入 Introduction

定义一个有向图$G=(V,E)$的**闭合图**（closure）[10]是该有向图的一个点集，且该点集的所有出边都还指向该点集。即闭合图内的任意点的任意后继也一定在闭合图中。更形式化地说，闭合图是这样的一个点集$V'\in V$，满足对于$\forall u\in V'$引出的$\forall\langle u,v\rangle\in E$，必有$v\in V'$成立。还有一种等价定义为：满足对于$\forall\langle u,v\rangle\in E$，若有$u\in V'$成立，必有$v\in V'$成立，在布尔代数上这是一个“**蕴含**(imply)”的运算。按照上面的定义，闭合图是允许超过一个连通块的。

给每个点v分配一个点权w_v（任意实数，可正可负）。**最大权闭合图**（maximum weight closure），是一个点权之和最大的闭合图，即最大化$\sum_{v\in V'}w_v$。

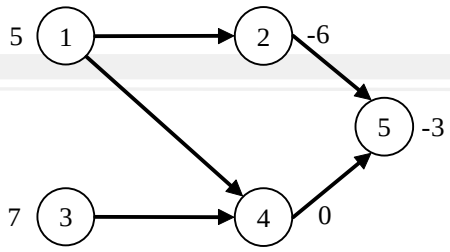

图 3.1 闭合图的例子

在图 3.1 中的网络有 9 个闭合图（含空集）：$\varnothing$，$\{3,4,5\}$，$\{4,5\}$，$\{5\}$，$\{2,4,5\}$，$\{2,5\}$，$\{2,3,4,5\}$，$\{1,2,4,5\}$和$\{1,2,3,4,5\}$，其中有最大权和的闭合图是$\{3,4,5\}$，权和为 4。

在许多实际应用中，给出的有向图常常是一个有向无环图（DAG），闭合图的性质恰好反映了事件间的**必要条件**的关系：一个事件的发生，它所需要的所有前提也都要发生。一个常见的例子就是制定大学的选课计划，其中一些课程需要以另一些课程为基础，这就是给出了有向无环图。若给所有课程打分，最大权闭合图对应了获益最大或效率最高的选课计划，解决了这个问题，就可以容易地选出一个最合理的选课计划。

在另外一些实际应用中，给出的是一个一般有向图，即有可能出现圈。常见的例子就是工程分期，一个工程可能含有很多项目，项目之间也有依赖关系。有些项目是需要同期一起开发，**互相依赖**的（即出现圈）。这样，也可以利用最大权闭合图，将最先应该完成的项目选出来作为工程的第一期。

[10] 闭合图为作者本人对 closure 的翻译，该名词出自参考文献[AMO93]，未见有其他中文文献的翻译。[Jiang01]中虽然提及本问题，但未提出对于该类图的通称。

3.2. 构造 Construction of Algorithm

下面给出将原问题的图 G 转化为网络 $N=(V_N,E_N)$ 的方法，从而可以利用最小割模型。

在原图点集的基础上增加源 s 和汇 t；将原图每条有向边 $\langle u,v\rangle\in E$ 替换为容量为 $c(u,v)=\infty$ 的有向边 $\langle u,v\rangle\in E_N$；增加连接源 s 到原图每个正权点 v $(w_v>0)$ 的有向边 $\langle s,v\rangle\in E_N$，容量为 $c(s,v)=w_v$；增加连接原图每个负权点 v $(w_v<0)$ 到汇 t 的有向边 $\langle v,t\rangle\in E_N$，容量为 $c(v,t)=-w_v$。其中，正无限 ∞ 定义为任意一个大于 $\sum_{v\in V}|w_v|$ 的整数。更形式化地，有：

$$V_N=V\cup\{s,t\}$$

$$E_N=E\cup\{\langle s,v\rangle\mid v\in V,w_v>0\}\cup\{\langle v,t\rangle\mid v\in V,w_v<0\}$$

$$\begin{cases}c(u,v)=\infty & \langle s,v\rangle\in E\\ c(s,v)=w_v & w_v>0\\ c(v,t)=-w_v & w_v<0\end{cases}$$

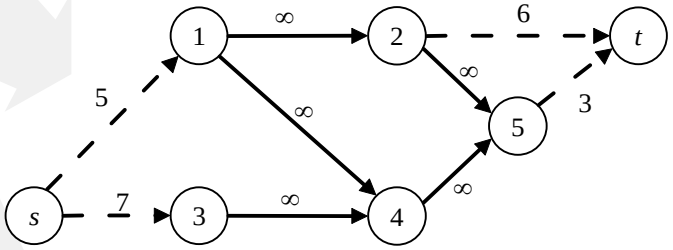

图 3.2 闭合图构图的例子

图 3.2 是将图 3.1 转化后的网络：从源 s 向两个正权点 1, 3 连边，从两个负权点 2, 5 向汇 t 连边。

3.3. 证明 Proof

定义：若一个 $s-t$ 割满足割中的每条边只都与源 s 或汇 t 关联，称该割为**简单割**（simple cut）。根据该定义，网络 N 的简单割是不含原图 G 的边集 E 中的任何边的。

引理3.1 在本问题的网络 N 中，最小割是简单割。

证明：网络 N 中边分两类：与源或汇关联的边，容量为有限值；不与源或汇关联的边，即原来边集 E 中的边，容量均为正无限。

所有与源或汇关联的边组成的割集的容量是有限的，为所有点权的绝对值和。最小割容量也至多为该绝对值和。故最小割不可能取任何容量为正无限的边，即最小割是简单割。

■

先规定一些符号。设简单割$[S,T]$将网络N的点集V_N划分为点集S及其补集T $(T=V_N-S)$，满足$s\in S$，$t\in T$。设闭合图为V_1，它在V中的补集为$\overline{V_1}$或V_2 $(V_2=V-V_1)$。记V^+为V中点权为正的点集，V^-为V中点权为负的点集。同样地，可以定义V_1^+与V_1^-，V_2^+与V_2^-。

下面的工作就是将简单割与闭合图一一对应起来，以便利用最小割模型求解。

引理3.2 （可行性）网络N的简单割$[S,T]$与图G的闭合图V_1方案存在一个一一对应关系：$V_1\bigcup\{s\}=S$。

证明：

(1) 闭合图对应简单割：即$S=V_1\bigcup\{s\}$，$T=\overline{V_1}\bigcup\{t\}$，求证$[S,T]$为简单割。

反证法。反设存在$\langle u,v\rangle\in E$，其中$u\in S-\{s\}=V_1$，$v\in T-\{t\}=\overline{V_1}$，使得$[S,T]$含不与源或汇关联的边。则闭合图$V_1$有一个后继不在闭合图内，矛盾。

(2) 简单割对应闭合图：即证$V_1=S-\{s\}$是闭合图。对于闭合图中$\forall u\in V_1$，考察任意一条由u引出的出边$\langle u,v\rangle\in E$，由于简单割$[S,T]$不含E中的任何边，故$v\notin T-\{t\}=\overline{V_1}$，即$v\in V_1$，符合闭合图的定义。

■

推论3.3 （对应关系的数量化）在引理 3.2 的一一对应关系下，有：

$$c[S,T]=\sum_{v\in V_2^+}w_v+\sum_{v\in V_1^-}(-w_v) \tag{3.1}$$

证明：割$[S,T]$按照与源或汇的关联情况，可以分为 3 部分：

$$[S,T]=[\{s\},V_2]\bigcup[V_1,\{t\}]\bigcup[V_1,V_2]$$

由于割$[S,T]$是简单割，故$[V_1,V_2]=\varnothing$。

又由于源s只与正权点有边，故$[\{s\},V_2]=[\{s\},V_2^+]$。

同样的汇t只与负权点有边，故$[V_1,\{t\}]=[V_1^-,\{t\}]$。

综上，$[S,T]=[\{s\},V_2^+]\bigcup[V_1^-,\{t\}]$，因此式(3.1)成立。

■

定理3.4 （最优性）当网络 N 的取到最小割时，其对应的图 G 的闭合图将达到最大权。

证明：按照定义，闭合图的权和为正权点的权的绝对值和减去负权点的权的绝对值和：

$$w(V_1)=\sum_{v\in V_1^+}w_v-\sum_{v\in V_1^-}(-w_v) \tag{3.2}$$

将式(3.2)与推论 3.3 的式(3.1)相加，得到：

$$\begin{aligned}w(V_1)+c[S,T]&=\sum_{v\in V_1^+}w_v-\sum_{v\in V_1^-}(-w_v)+\sum_{v\in V_2^+}w_v+\sum_{v\in V_1^-}(-w_v)\\&=\sum_{v\in V_1^+}w_v+\sum_{v\in V_2^+}w_v\\&=\sum_{v\in V^+}w_v\end{aligned}$$

整理得：

$$w(V_1)=\sum_{v\in V^+}w_v-c[S,T] \tag{3.3}$$

观察式(3.3)，目标是最大化 $w(V_1)$，而正权点的总权和 $\sum_{v\in V^+}w_v$ 为定值，故最小化简单割的容量 $c[S,T]$，便可得到最大的 $w(V_1)$。根据引理 3.1，最小割为简单割。所以最小割是最小的简单割，故答案为网络 N 的最小割所对应的闭合图。

■

算法复杂度便是求最小割的复杂度：$O(\text{MaxFlow}(N))$。

3.4. 应用 Application

3.4.1. 最大获利 Profit[11]

描述 Description

ADN 公司得到了一共 n 个可以作为通讯信号中转站的地址。由于这些地址的地理位置差异，在不同的地方建造通讯中转站所需要投入的成本也是不一样的，所幸在前期调查之后这些都是已知数据：建立第 i 个通讯中转站需要成本 p_i。另外公司调查得出了所有期望中的用户群，一共 m 个。第 i 个用户群的用户会使用中转站 a_i 和中转站 b_i 进行通讯，公司获益 c_i。ADN 公司可以有选择地建立一些中转站（其成本之和为总成本），为一些用户提供服务并获得收益（收益之和为总收益）。那么如何选择最终建立的中转站才能让公司的净获利最大呢？（净获利 = 总收益 — 总成本）

[11] 题目来源：NOI 2006 Day 2 最大获利(Profit)

解答 Solution

首先分析题目中的决策因素。在满足了第 i 个用户群后，便可以得到收益 c_i，然而满足第 i 个用户群需要有**必要条件**：建立中转站 a_i 和中转站 b_i，同时要花去相应费用。留心这个所谓的必要条件，便可联想到闭合图的性质。分析后发现，本题就是最大权闭合图的一个特例。把它抽象成这样一个有向图模型：每个用户群 i 作为一个结点分别向相应的中转站 a_i 和中转站 b_i 连有向边。事实上，这是一个二分图上的特例。求这个图的最大权闭合图，便可解决本题。复杂度为 $O(\text{MaxFlow}(n+m, n+m))$。

而4.7 节给出的结果，可以使本题在仅 $O(\text{MaxFlow}(n, n+m))$ 的时间内解决。这也是本文比较重要的突破之一。参见应用 4.8.2。

4. 最大密度子图 Finding a Maximum Density Subgraph

4.1. 引入 Introduction

定义一个无向图 $G=(V,E)$ 的**密度**(density) D 为该图的边数 $|E|$ 与该图的点数 $|V|$ 的比值 $D=\frac{|E|}{|V|}$。给出一个无向图 $G=(V,E)$，其具有最大密度的子图 $G'=(V',E')$ 称为**最大密度子图**（maximum density subgraph），即最大化 $D'=\frac{|E'|}{|V'|}$。

简记 $n=|V|$， $m=|E|$。

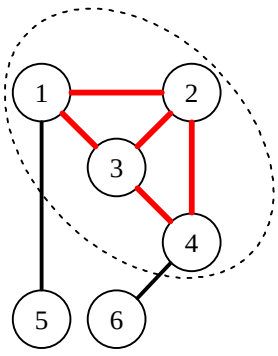

图 4.1 最大密度子图的例子

举例，在图 4.1 中，给出了一幅 6 个点的图，其中在虚线内的点与边组成最大密度子图，

红边组成子图的边集，密度为$\frac{5}{4}$。

4.2. 主算法 Main Algorithm

先尝试着更形式化地重新叙述本模型：

$$\text{Maximize} \quad D = f(\mathbf{x}) = \frac{\sum_{e\in E} x_e}{\sum_{v\in V} x_v}$$

其中$x_v, x_e \in \{0,1\}$，表示对于每个点或每条边是否在子图$G'=(V',E')$中，即$v\in V' \Leftrightarrow x_v=1$，$\forall e\in E' \Leftrightarrow x_e=1$。并且对于$\forall e=(u,v)\in E'$，必有$u,v\in V'$。

联系1.6 节的分数规划，以上规划就是一个 0-1 分数规划的模型。根据分数规划一般步骤，应是进行二分查找答案，对于一个答案的猜测值g，构造一个新函数

$$h(g)=\max_{\mathbf{x}}\left\{\sum_{e\in E}1\cdot x_e-\sum_{v\in V} g\cdot x_v\right\}$$

设$D^*=f(\mathbf{x}^*)$为原规划的最优解，由推论 1.9 知：

$$\begin{cases} h(g)=0 \Leftrightarrow g=D^* \\ h(g)<0 \Leftrightarrow g>D^* \\ h(g)>0 \Leftrightarrow g<D^* \end{cases}$$

于是主算法便是对最优解D^*的二分查找，每次查找得到一个猜测值g，以求解$h(g)$来进行判定，从而缩小查找范围。设二分查找次数为k，解决$h(g)$的复杂度为$\text{Solve}(h(g))$，则该算法复杂度为：$O(k\cdot\text{Solve}(h(g)))$。

先确定二分查找的范围。由于密度不超过$\frac{m}{1}$，且不小于$\frac{1}{n}$，便将它们作为二分查找的上下界。接下来，确定二分查找的精度。

引理4.1 无向图G中，任意两个具有不同密度的子图G_1，G_2，它们的密度差不小于$\frac{1}{n^2}$。

证明：不妨设G_1的密度大于G_2的密度。设G_1，G_2的点数分别为n_1，n_2，边数分别为m_1，m_2。则有：

$$\frac{m_1}{n_1}-\frac{m_2}{n_2}=\frac{m_1n_2-m_2n_1}{n_1n_2}\geq\frac{1}{n_1n_2}\geq\frac{1}{n^2}$$

■

以上引理给出了不同密度的子图的间距，容易有如下推论。

推论4.2 对于无向图G中，有一个密度为D的子图G'，且在无向图G中不存在一个密度超过$D+\frac{1}{n^2}$的子图，则G'为最大密度子图。

■

于是，根据推论 4.2，则二分查找的次数为

$$k=O\left(\log\left(\frac{\frac{m}{1}-\frac{1}{n}}{\frac{1}{n^2}}\right)\right)=O\left(\log\left(n^4\right)\right)=O(\log n)$$

则总算法复杂度为：$O(\log n\cdot \mathrm{Solve}(h(g)))$。

问题便集中在如何求出$h(g)$，下面分别使用了两种方法来求解$h(g)$。

4.3. 初步算法 Simple Algorithm

假设已经给出了一个答案的猜测值g，需求解：

$$h(g)=\max_{\mathbf{x}}\left\{\sum_{e\in E}1\cdot x_e-\sum_{v\in V}g\cdot x_v\right\}$$

更形象化的解释，便是选出一个G的子图$G'=(V',E')$，并且对于$\forall e=(u,v)\in E'$，必有$u,v\in V'$，且最大化

$$\text{Maximize}\qquad \sum_{e\in E'}1-\sum_{v\in V'}g=|E'|-g\cdot|V'|$$

注意限制条件：对于$\forall e=(u,v)\in E'$，必有$u,v\in V'$。就是说边(u,v)的存在的**前提**或**必要条件**是u与v的存在。观察这个必要条件，具有第 3 节论述的最大权闭合图的限制条件形式。可以把本模型中的边e看成点v_e，带上点权1；把本模型中的点保留，带上点权$-g$；对于所有边$\forall e=(u,v)\in E$，建立两条有向边$\langle v_e,u\rangle$，$\langle v_e,v\rangle$。此时，原图便转化为一个最大权闭合图的模型。

更形式化地，由原图$G(V,E)$建立新图$D=(V_D,E_D)$：

$$V_D = V \bigcup \{v_e \mid e \in E\}$$
$$E_D = \left\{ \langle v_e, u \rangle, \langle v_e, v \rangle \mid e = (u,v) \in E \right\}$$
$$\begin{cases} w_{v_e} = 1 & e \in E \\ w_v = -g & v \in V \end{cases}$$

如图 4.2(a)给出一个图 G，通过以上转化过程，转化为图 4.2(b)的新图 D，解决新图的最大权闭合图，就是解决 $h(g)$。

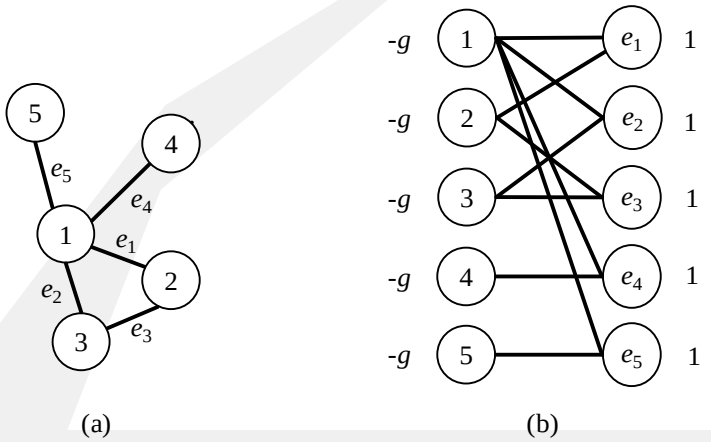

图 4.2 最大密度子图的初步算法的例子

这个转化的一一对应关系比较简单，便不给出具体证明。而根据第 3 节的结论，解决 $h(g)$ 的复杂度为 $O(\text{MaxFlow}(D)) = O(\text{MaxFlow}(n+m, n+m))$。故解决整个模型的复杂度为 $O(\log n \cdot \text{MaxFlow}(n+m, n+m))$。

4.4. 改进算法 Improved Algorithm

4.3 节直接利用了最大权闭合图的模型，将边转成了点，形成了二分图，从而解决了问题。但却更一般化了模型，复杂度自然较高。而原模型应该具有很多特殊性质需要挖掘。下面从另一个角度去分析这个问题。

回顾需求解的问题 $h(g)$，它要求选出图 G 的一个子图 $G' = (V', E')$，并且对于 $\forall e = (u,v) \in E'$，必有 $u, v \in V'$，且最大化

$$\text{Maximize} \qquad |E'| - g \cdot |V'|$$

考察限制条件：$\forall e = (u,v) \in E'$，必有 $u, v \in V'$。它只是一个边集对点集的**最基本的限制**。又由于目标是最大化，在点集 V' 一定的情况下，尽量多地选边，解自然更优。下面给出的引理显然成立。

引理4.3 对于图G的子图$G'=(V',E')$，在点集V'固定的情况下，E'取V'的**导出子图**的边集，比其他的E'的方案，在$h(g)$的目标函数的意义下更优。

■

于是可以直接把E'**重定义**为V'的**导出子图**的边集（端点都在V'的所有边的集合）。

继续分析。如图 4.3，假设其中虚线圈内的点组成最优的点集V'；加粗的边便组成要选出的子图的边集E'。正向思维是主动去选出这些边。若以逆向思维思考，利用补集转换思想：可不可以选出那些与V'关联，同时又不是E'中的边（红色的边），再用与V'关联的总边集（V'的总度数）减去这些红色的边就可以得到E'？

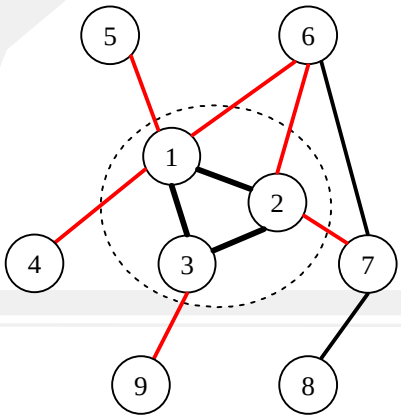

图 4.3

顺着这个思路，继续分析红色的边的本质，它们是什么呢？它们是V'与V'的补集$\overline{V'}$之间的边，即是割集$[V',\overline{V'}]$。能不能利用最小割呢？下面逐步转化。

由于原模型要求目标函数是最大化，应该先转化为最小化（乘上-1）：

$$\text{Maximize} \quad |E'|-g\cdot|V'|=\sum_{e\in E'}1-\sum_{v\in V'}g$$

$$\Updownarrow$$

$$\begin{aligned}\text{Minimize} \quad g\cdot|V'|-|E'| &= \sum_{v\in V'}g-\sum_{e\in E'}1\\ &= \sum_{v\in V'}g-\left(\frac{\sum_{v\in V'}d_v}{2}-c[V',\overline{V'}]\right)\\ &= \sum_{v\in V'}(g-\frac{d_v}{2})+c[V',\overline{V'}]\end{aligned}$$

其中d_u表示u的度数$d_u=\sum_{(u,v)\in E}1$。除以2是由于一条边在度的总和中算了两次。为了让上面的式子更整齐，再同乘以系数2，得到：

$$\sum_{v\in V'}(2g-d_v)+2c[V',\overline{V'}]$$

其中$\sum_{v\in V'}(2g-d_v)$，就是说选了点v，就要花费代价$(2g-d_v)$。可以以常用的处理点权的办法，让V'中的点都与汇t相连，把点权赋给这条边的容量。

由于最小割只能接受非负的边权，但这些点权可能为负数，所以对于所有表示点权的边容量，都需要加上一个足够大的数U，保证非负。

将上面的思路整理一下，即是将原图$G(V,E)$转化为网络$N=(V_N,E_N)$的过程：在原图点集V的基础上增加源s和汇t；将每条原无向边(u,v)替换为两条容量为1的有向边$\langle u,v\rangle$和$\langle v,u\rangle$；增加连接源s到原图每个点v的有向边$\langle s,v\rangle$，容量为U；增加连接原图每个点v到汇t的有向边$\langle v,t\rangle$，容量为$(U+2g-d_v)$。更形式化地，有：

$$V_N=V\cup\{s,t\}$$
$$E_N=\{\langle u,v\rangle\mid(u,v)\in E\}\cup\{\langle s,v\rangle\mid v\in V\}\cup\{\langle v,t\rangle\mid v\in V\}$$
$$\begin{cases}c(u,v)=1 & (u,v)\in E\\ c(s,v)=U & v\in V\\ c(v,t)=U+2g-d_v & v\in V\end{cases}$$

事实上，因为对于$\forall v\in V$，度数d_v不会超过总边数m，且由于猜测值g是非负的，所以令足够大的数U取m，即令$U=m$，便可保证所有的容量都是非负的。

图 4.4(a)给出原图G，通过以上过程转化为图 4.4(b)的网络N。

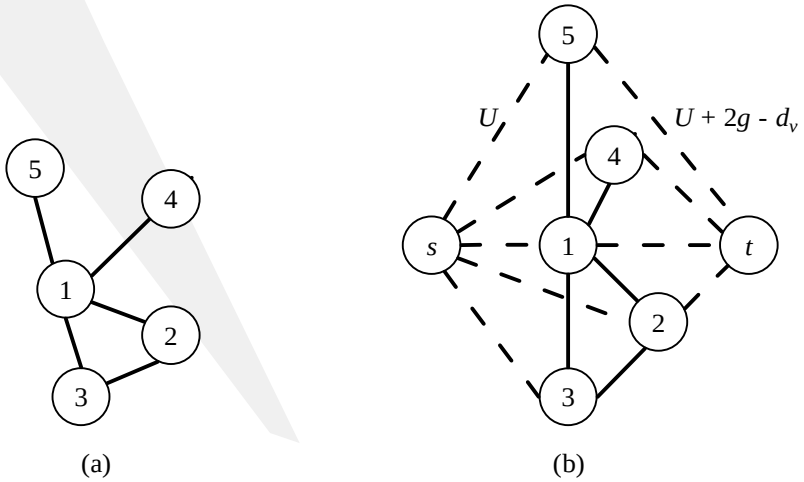

图 4.4 改进算法的构图

4.5. 改进算法的证明 Proof of Improved Algorithm

设割集$[S,T]$为网络N的一个割，$s\in S$，$t\in T$。G'为G的一个子图。由于割$[S,T]$可以是V的任意划分，所有它们之间显然有一个一一对应关系：

引理4.4 （可行性）网络N的一个割$[S,T]$与图G的子图$G'=(V',E')$方案存在一个一一对应关系：$V'\cup\{s\}=S$。

■

定理4.5（最优性）网络N的一个最小割$[S,T]$，它根据引理 4.3 所对应的子图G'是$h(g)$的一个最优解。

证明：问题$h(g)$，要求选出一个G的子图$G'=(V',E')$，且最大化

$$\text{Maximize} \qquad |E'|-g\cdot|V'| \tag{4.1}$$

设$c[S,T]$为任意网络N的割的容量。根据引理 4.4，则$V'=S-\{s\}$，$\overline{V'}=T-\{t\}$。进而有如下式子成立：

$$\begin{aligned}
c[S,T] &= \sum_{u\in S,v\in T} c_{u,v} \\
&= \sum_{v\in\overline{V'}} c_{s,v} + \sum_{u\in V'} c_{u,t} + \sum_{u\in V',v\in\overline{V'},(u,v)\in E} c_{u,v} \\
&= \sum_{v\in\overline{V'}} U + \sum_{u\in V'} (U+2g-d_u) + \sum_{u\in V',v\in\overline{V'},(u,v)\in E} 1 \\
&= U|V| + \sum_{u\in V'}\left(2g-d_u+\sum_{v\in\overline{V'},(u,v)\in E}1\right) \\
&= U|V| + \sum_{u\in V'}\left(2g-\sum_{v\in V',(u,v)\in E}1\right) \qquad (*) \\
&= U\cdot n + 2g|V'| - 2|E'| \\
&= U\cdot n - 2\left(|E'|-g|V'|\right)
\end{aligned}$$

其中，式$(*)$的代换是由于E'已重定义为V'的导出子图的边集（引理 4.3）。

由于$U\cdot n$为定值，则当割取到最小割时，$|E'|-g|V'|$取到最大。即$h(g)=\dfrac{U\cdot n-c[S,T]}{2}$，其中$[S,T]$为最小割。

■

故解决$h(g)$的复杂度为$O(\text{MaxFlow}(N))=O(\text{MaxFlow}(n,n+m))$。整个模型的复杂度为$O(\log n\cdot\text{MaxFlow}(n,n+m))$。

4.6. 向带边权图的推广 Generalization to Edge Weighted Graphs

给出一个带边权无向图 $G=(V,E)$，其中每条边有一个非负边权 w_e（$w_e \geq 0$）。定义带边权无向图的**密度**为该图的边权和 $\sum_{e\in E} w_e$ 与该图的点数 $|V|$ 的比值 $D=\frac{\sum_{e\in E} w_e}{|V|}$。求解在这种定义下的最大密度子图，便构成本小节的内容。

主体思想还是沿用与改造4.4 节的给出算法。而在本模型中，为了化归分数规划而构造的新函数 $h(g)$ 则为：要求选出一个 G 的子图 $G'=(V',E')$，且最大化

$$\text{Maximize} \quad \sum_{e\in E'} w_e - \sum_{v\in V'} g = \sum_{e\in E'} w_e - g\cdot|V'| \tag{4.2}$$

比较式(4.1)与式(4.2)，只有边权不同，可以试着改造边权。重新定义度 d_u，将其值改写为与 u 关联的所有边的权和，即 $d_u=\sum_{(u,v)\in E} w_e$。再将所有边 e 在网络 $N=(V_N,E_N)$ 中对应的边的容量由1换成 w_e。而一个足够大的值 U 从 m 换为总边权和 $\sum_{e\in E} w_e$。形式化地给出新的网络 $N=(V_N,E_N)$ 的构图，有：

$$V_N = V\cup\{s,t\}$$
$$E_N = \{\langle u,v\rangle \mid (u,v)\in E\}\cup\{\langle s,v\rangle \mid v\in V\}\cup\{\langle v,t\rangle \mid v\in V\}$$
$$\begin{cases} c(u,v)=w_e & (u,v)\in E \\ c(s,v)=U & v\in V \\ c(v,t)=U+2g-d_v & v\in V \end{cases}$$
$$U=\sum_{e\in E} w_e$$

由于边权非负，引理 4.3 还是成立的。下面把重定义的量，带入到定理 4.5 中，发现定理 4.5 的推广还是成立，即仍然有 $h(g)=\frac{U\cdot n-c[S,T]}{2}$，其中 $[S,T]$ 为最小割。

故解决 $h(g)$ 的复杂度仍为 $O(\text{MaxFlow}(N))=O(\text{MaxFlow}(n,n+m))$。然而由于密度的定义的推广，引理 4.1 与推论 4.2 便不成立了。需要规定二分查找的精度，二分查找的次数便成为一个伪多项式。但是在实际应用中，二分查找的逼近速度还是很快的。设二分查找次数为 k，所以整个模型的复杂度为 $O(k\cdot\text{MaxFlow}(n,n+m))$。

4.7. 向点边均带权的图的推广 Generalization to Both Node and Edge Weighted Graphs

给出一个点边均带权的无向图$G=(V,E)$，其中每个点v有一个点权p_v（$p_v \in R$，可正可负），其中每条边e有一个非负边权w_e（$w_e \geq 0$）。定义点边均带权的无向图的**密度**为该图的点权和加上边权和的和$\sum_{v\in V} p_v + \sum_{e\in E} w_e$与该图的点数$|V|$的比值，即

$$D = \frac{\sum_{v\in V} p_v + \sum_{e\in E} w_e}{|V|}$$

求解在这种定义下的最大密度子图，便构成本小节的内容。

主体思想还是沿用与改造4.6 节向带边权图的推广中所给出的算法。而在本模型中，为了化归分数规划而构造的新函数$h(g)$则为：要求选出一个G的子图$G'=(V',E')$，且最大化

$$\text{Maximize} \quad \sum_{e\in E'} w_e + \sum_{v\in V'} p_v - \sum_{v\in V'} g = \sum_{e\in E'} w_e - \sum_{v\in V'} (g - p_v) \tag{4.3}$$

同4.6 节一样，重新定义度d_u，改为与u关联的所有边的权和，即$d_u = \sum_{(u,v)\in E} w_e$。再将所有边$e$在网络$N=(V_N,E_N)$中对应的边的容量由1换成$w_e$。比较式(4.2)与式(4.3)，点权部分发生了改变，式(4.2)的g换成了式(4.3)的$(g-p_v)$。所以从任意点v到汇t的容量改为$c(v,t)=U+2(g-p_v)-d_v$。而一个足够大的值U换为点权绝对值和的两倍与总边权和的和$2\sum_{v\in V}|p_v| + \sum_{e\in E} w_e$。

形式化地给出新的网络$N=(V_N,E_N)$的构图，有：

$$V_N = V \cup \{s,t\}$$

$$E_N = \{\langle u,v\rangle \mid (u,v)\in E\} \cup \{\langle s,v\rangle \mid v\in V\} \cup \{\langle v,t\rangle \mid v\in V\}$$

$$\begin{cases} c(u,v) = w_e & (u,v)\in E \\ c(s,v) = U & v\in V \\ c(v,t) = U + 2g - d_v - 2p_v & v\in V \end{cases}$$

$$U = 2\sum_{v\in V}|p_v| + \sum_{e\in E} w_e$$

由于边权非负，引理 4.3 还是成立的。下面把重定义的量，带入到定理 4.5 中，发现定理 4.5 的推广仍成立，即仍然有$h(g)=\frac{U\cdot n - c[S,T]}{2}$，其中$[S,T]$为最小割。

故解决$h(g)$的复杂度仍为$O(\text{MaxFlow}(N)) = O(\text{MaxFlow}(n, n+m))$。然而同4.6 节一样，由于密度的定义的推广，引理 4.1 与推论 4.2 便不成立了，需要规定二分查找的精度。设二分查找次数为k，故整个模型的复杂度为$O(k\cdot \text{MaxFlow}(n, n+m))$。

4.8. 应用 Application

4.8.1. 生活的艰辛 Hard Life[12]

描述 Description

ADN 公司内部共 n 个员工，员工之间可能曾经因为小事有了过节，总是闹矛盾。若员工 u 和员工 v 有矛盾，用边(u, v)表示，共 m 个矛盾。最近，ADN 公司内部越来越不团结，Amber 决定裁员。Amber 想得到一个被裁人员的清单，使得被裁人员间的不团结率最高。不团结率定义为被裁人员间的矛盾总数与被裁人员数的比值（不团结率 = 被裁人员之间的矛盾总数 / 被裁人员数）

解答 Solution

本题中的不团结率就是4.1 节所介绍的图的密度，故直接使用本节所述方法解决即可。

4.8.2. 最大获利 Profit

描述 Description

同3.4.1 节。

解答 Solution

将用户群看成边，中转站看成点，问题就可以转换为：选出一些边 e 使其“满足”，可得到收益 w_e；选出一些点 v 进行“建设”，得到花去费用 p_v。这些选出的点组成 V'，选出的边组成 E'。解题目标便转化为

$$\text{Maximize} \quad \sum_{e\in E'} w_e - \sum_{v\in V'} p_v$$

限制条件是：满足第 i 个用户群需要有**必要条件**：建立中转站 a_i 和中转站 b_i。对应图中的限制条件为：$\forall e=(u,v)\in E'$，必有 $u,v\in V'$。这个限制条件同时也是最大密度子图模型的限制条件，于是可以类比点边均带权的图的最大密度子图。

观察4.7 节的优化目标：

$$\text{Maximize} \quad \sum_{e\in E'} w_e + \sum_{v\in V'} p_v - \sum_{v\in V'} g = \sum_{e\in E'} w_e - \sum_{v\in V'} (g - p_v)$$

若令其中的 $(g-p_v)$ 为本题中的 p_v，得到

[12] 题目来源：ACM/ICPC Northeastern European (NEERC) 2006 – H. Hard Life

$$\text{Maximize} \quad \sum_{e\in E'} w_e - \sum_{v\in V'} p_v$$

这就是本题所要求的优化式子。故本题得到成功化归，用4.7 节的方法可以在仅 $O(\text{MaxFlow}(n, n+m))$ 的时间内解决。

相对3.4.1 节给出的 $O(\text{MaxFlow}(n+m, n+m))$，本算法有了本质上的改进，也达到了用最小割解决该题的一个下界（点边的数量都是输入的数量）。

5. 二分图的最小点权覆盖集与最大点权独立集 Minimum Weight Vertex Covering Set and Maximum Weight Vertex Independent Set in a Bipartite Graph

5.1. 引入 Introduction

下面给出一些与本节有关的定义。

在一个无向图 $G=(V,E)$ 中，有如下定义。

点覆盖集（vertex covering set，VCS）是无向图 G 的一个点集，使得该图中所有边都至少有一个端点在该集合内。形式化的定义是点覆盖集为 $V'\in V$，满足对于 $\forall(u,v)\in E$，都有 $u\in V'$ 或 $v\in V'$ 成立，即 $u\in V'$，$v\in V'$ 至少一个成立。形象地说是若干个点“覆盖”住了与它们邻接的边，这些边恰好组成了原边集。

点独立集（vertex independent set，VIS）是无向图 G 的一个点集，使得任两个在该集合中的点在原图中都不相邻。或者说是导出子图为零图（没有边）的点集。形式化的定义是点独立集为 $V'\in V$，满足对于 $\forall u,v\in V'$，都有 $(u,v)\notin E$ 成立。点独立集还有一种等价的定义：点独立集为 $V'\in V$，满足对于 $\forall(u,v)\in E$，都有 $u\in V'$ 与 $v\in V'$ 不同时成立。

最小点覆盖集（minimum vertex covering set，MinVCS）是在无向图 G 中，点数最少的点覆盖集。

最大点独立集（maximum vertex independent set，MaxVIS）是在无向图 G 中，点数最多的点独立集。

以上两个问题都是经典的NPC问题，但在二分图中都可以用最大匹配模型来快速解决。[13] 本节是要讨论这两个模型的点权一般化扩展。

一个带点权无向图 $G=(V,E)$，对于 $\forall v\in V$，都被分配一个非负点权 w_v。在该图中，

[13] 关于它们的解决方法可参见参考文献[AMO93]。

给出下如定义。

最小点权覆盖集（minimum weight vertex covering set，MinWVCS）是在带点权无向图G中，点权之和最小的点覆盖集。

最大点权独立集（maximum weight vertex independent set，MaxWVIS）是在带点权无向图G中，点权之和最大的点独立集。

一个带点权二分图$G=(V,E)$中，其中$V=X\bigcup Y$，$X\bigcap Y=\varnothing$，且对于$\forall v\in V$，都被分配了一个非负点权w_v（$w_v\geq 0$）。在该二分图中，最小点权覆盖集与最大点权独立集的解法，便是本节所讨论的内容。

5.2. 二分图的最小点权覆盖集算法 Algorithm for MinWVCS in a Bipartite Graph

回顾与此模型相关的模型。简化权的条件后，可以借鉴的是二分图匹配的最大流解法。它加入了额外的源s和汇t，将匹配以一条条$s-u-v-t$形式的流路径“串联”起来。匹配的限制是在点上，恰当地利用了流的容量限制。而点覆盖集的限制在边上，最小割是最大流的对偶问题，对偶往往是将问题的性质从点转边，从边转点。可以尝试着转化到最小割模型。

基于以上尝试动机，建立一个源s，向X部每个点连边；建立一个汇t，从Y部每个点向汇t连边，把二分图中的边看成是有向的。则任意一条从s到t的路径，一定具有$s-u-v-t$的形式。割的性质是**不存在一条从s到t的路径**。故路径上的三条边$\langle s,u\rangle$，$\langle u,v\rangle$，$\langle v,t\rangle$中至少有一条边在割中。若人为地令$\langle u,v\rangle$不可能在最小割中，即令其容量为正无限$c(u,v)=\infty$。则条件简化为$\langle s,u\rangle$，$\langle v,t\rangle$中至少一条边在最小割中，正好与点覆盖集限制条件的形式相符（$u\in V'$，$v\in V'$至少一个成立）。而目标是最小化点权之和，恰好也是最小割的优化目标。

形式化以上推理过程，将原问题的图G转化为网络$N=(V_N,E_N)$的最小割模型。

在原图点集的基础上增加源s和汇t；将每条二分图的边$\langle u,v\rangle\in E$替换为容量为$c(u,v)=\infty$的有向边$\langle u,v\rangle\in E_N$；增加源$s$到$X$部的点$u$的有向边$\langle s,u\rangle\in E_N$，容量即该点权值$c(s,u)=w_u$；增加$Y$部的点$v$到汇$t$的有向边$\langle v,t\rangle\in E_N$，同样，容量为该点权值$c(v,t)=w_v$。更形式化地，有：

$$V_N = V \bigcup \{s,t\}$$
$$E_N = E \bigcup \{\langle s,u\rangle \mid u \in X\} \bigcup \{\langle v,t\rangle \mid v \in Y\}$$
$$\begin{cases} c(u,v) = \infty & \langle u,v\rangle \in E \\ c(s,u) = w_u & u \in X \\ c(v,t) = w_v & v \in Y \end{cases}$$

图 5.1(a)给出了一个带点权的二分图G，图 5.1(b)是转化后的网络N。

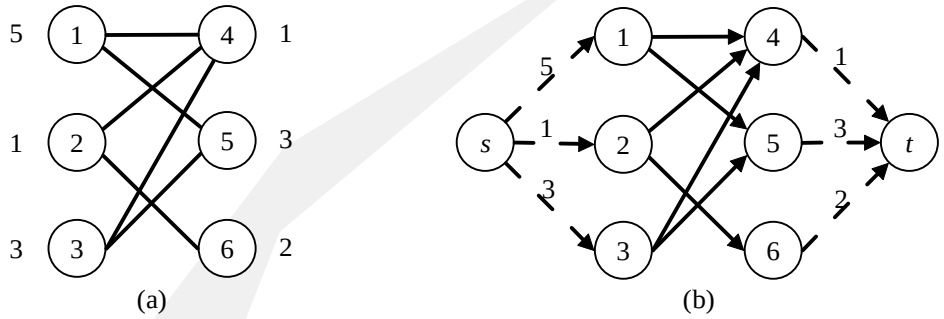

图 5.1 二分图的最小点权覆盖集算法的例子

这里沿用3.3 节的简单割概念（只含与源或汇关联的边）。下面不加证明地给出一个简单的对应关系。

引理5.1 网络N的简单割$[S,T]$与图G的点覆盖集$V' = X' \bigcup Y'$存在一一对应关系：点覆盖集中的点在网络N中相应的带权边组成一个简单割；反之亦然，即：

$$[S,T] = [\{s\}, X'] \bigcup [Y', \{t\}]$$

■

由于引理 5.1 所给出的一一对应，并且因为最小割与最小权点覆盖的优化方向一致，所以本问题可以用最小割解决。算法复杂度便是求最小割的复杂度：$O(\text{MaxFlow}(N))$。

5.3. 二分图的最大点权独立集算法 Algorithm for MaxWVIS in a Bipartite Graph

考察点独立集V'的定义为，所有边$\forall e(u,v) \in E$，都满足$u \in V'$与$v \in V'$不同时成立。重写成布尔代数的形式，让复杂关系代数化：$\neg(u \in V' \text{ and } v \in V')$。接着，由 De Morgan 定律，得到：

$$\neg(u \in V' \text{ and } v \in V')$$
$$\Leftrightarrow \quad \neg u \in V' \text{ or } \neg v \in V'$$
$$\Leftrightarrow \quad u \in \overline{V'} \text{ or } v \in \overline{V'}$$

最右端的式子与点覆盖集的限制条件相符，而且有了5.2 节点独立集的基础，研究点独立集与点覆盖集的关系，化归到点独立集，有充分的动机。

定理5.2 （覆盖集与独立集互补定理）若$\overline{V'}$为不含孤立点的任意图的一个点覆盖集当且仅当V'是该图的一个点独立集。

证明：

(1) 充分性，覆盖集$\Rightarrow$独立集

反证法。反设V'不是独立集，即存在点对$\exists u,v\in V'$，使得$(u,v)\in E$。此时，$u,v\notin\overline{V'}$，然而边(u,v)就没有被$\overline{V'}$覆盖到。矛盾。

(2) 必要性，独立集$\Rightarrow$覆盖集

反证法。反设$\overline{V'}$不是覆盖集，即存在边$\exists(u,v)\in E$，使得其端点都不在覆盖集内$u,v\notin\overline{V'}$。此时，$u,v\in V'$，然而$(u,v)\in E$，与V'是独立集矛盾。

■

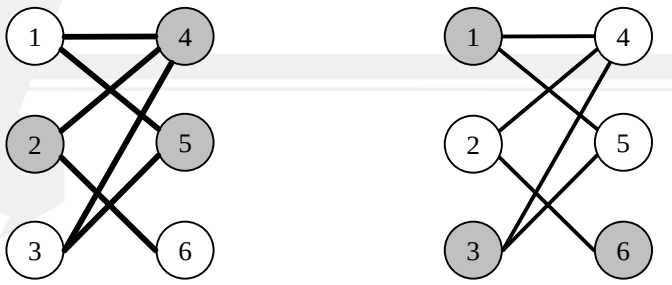

图 5.2(a)(b) 覆盖集与独立集互补定理的例子

图 5.2(a)给出了一个二分图的点覆盖方案，灰色点组成点覆盖集；而图 5.2(b)中的灰色点组成图 5.2(a)的点覆盖集的补集，并且是个点独立集。

推论5.3 （最优性）若V'为不含孤立点的任意图的一个最小点权覆盖集，则$\overline{V'}$就是该图的一个最大点权独立集。

证明：由定理 5.2，且$\overline{V'}+V'=V$，有：

$$\sum_{v\in V}w_v=\sum_{v\in V'}w_v+\sum_{v\in\overline{V'}}w_v$$

又因为总点权和是固定的，所以最小化点权覆盖集V'，便可最大化点权独立集$\overline{V'}$。

■

算法复杂度便是求最小点权覆盖集的复杂度：$O(\mathrm{MaxFlow}(N))$。

5.4. 应用 Application

5.4.1. 有向图破坏 Destroying the Graph[14]

描述 Description

给出一个有向图 $D=(V,E)$。对于每个点 u，定义两种操作 $a(u)$，$b(u)$：

操作 $a(u)$：删除点 u 的所有出边，即 $\forall\langle u,v\rangle\in E$，操作花费代价 $c_{a(u)}$。

操作 $b(u)$：删除点 u 的所有入边，即 $\forall\langle v,u\rangle\in E$，操作花费代价 $c_{b(u)}$。

求将原图的边集的边全部删除的最少代价。

解答 Solution

首先从操作的定义入手，考察本题的每条边被删除的情况。对于指定的一条边 $\langle u,v\rangle\in E$，只有操作 $a(u)$ 或操作 $b(v)$ 可以将其删除掉，即操作 $a(u)$ 或操作 $b(v)$ 至少要有一个被执行。可以联想到点覆盖的定义（$u\in V'$ or $v\in V'$）。若将一个操作本身就看成是一个点，一条有向边看成是连接 2 个操作的边，这不就是一个最小点权覆盖模型么？

形式化以上推理过程，将原问题的有向图 D 转化为图 $G=(V_G,E_G)$ 的最小点权覆盖模型：将原图的点 v 拆分为 v_a 与 v_b，分别将操作 $a(v)$，$b(v)$ 与点 v_a，v_b 对应起来；将原图的每条边 $\langle u,v\rangle\in E$ 替换为有向边 $\langle u_a,v_b\rangle\in E_G$；对每个 u_a 点令 $w_{u_a}=c_{a(u)}$，对每个 v_b 点令 $w_{v_b}=c_{b(v)}$。更形式化地，有：

$$
\begin{aligned}
&X=\{u_a\mid u\in V\}\\
&Y=\{v_b\mid v\in V\}\\
&V_N=X\bigcup Y\\
&E_N=\{\langle u_a,v_b\rangle\mid\langle u,v\rangle\in E\}\\
&w_{u_a}=c_{a(u)}\qquad u\in V\\
&w_{v_b}=c_{b(v)}\qquad v\in V
\end{aligned}
$$

由于图 G 中的边都是由 X 集合指向 Y 集合对应的点。故图 G 是二分图，设 $V_G=X\bigcup Y$。另设原问题的所有 $a(u)$ 操作构成集合 A，所有 $b(u)$ 操作构成集合 B。下面不加证明地给出一个简单的定理（反证法），对上述两方建立起一一对应关系。

[14] 题目来源：ACM/ICPC Northeastern Europe (NEERC) 2003, Northern Subregion – D. Destroying the Graph

定理5.4 在所有操作 $a(v)$，$b(v)$ 与点 v_a，v_b 一一对应的关系下，原问题的一个决策 $A'\subseteq A$，$B'\subseteq B$ 与图 G 的一个点覆盖集 $V_G'=X'\bigcup Y'$ 一一对应。

■

这样就成功地将该问题转化为二分图中的最小点权覆盖模型问题，使用5.2 节所描述的解法解决即可。

5.4.2. 王者之剑 Exca[15]

描述 Description

给出一个 $n\times m$ 网格，每个格子上有一个价值为 $v_{i,j}$ 的宝石。Amber 可以自己决定起点，开始时刻为第 0 秒。以下操作，在每秒内**按顺序**执行。

1．若第 i 秒开始时，Amber 在 (x,y)，则 Amber 可以拿走 (x,y) 上的宝石。

2．在偶数秒时（i 为偶数），则 Amber 周围 4 格的宝石将会消失。

3．若第 i 秒开始时，Amber 在 (x,y)，则在第 $(i+1)$ 秒开始前，Amber 可以马上移动到相邻的格子 $(x+1,y)$，$(x-1,y)$，$(x,y+1)$，$(x,y-1)$ 或原地不动 (x,y)。

求 Amber 最多能得到多大总价值的宝石。

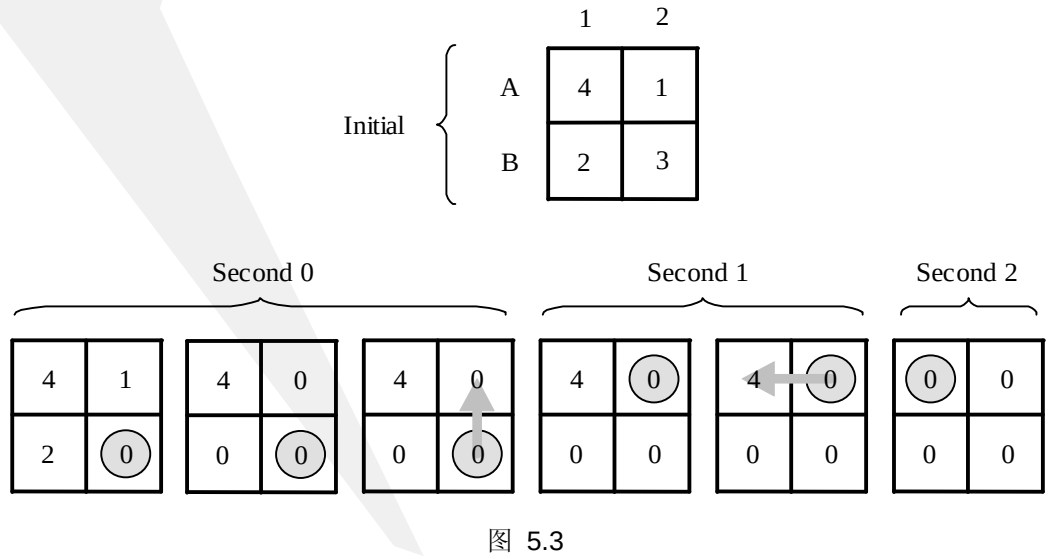

图 5.3

图 5.3 给出了一个 2×2 的网格的例子。在第 0 秒，首先选择 B2 进入，取走宝石 3；由于是偶数秒，周围的格子 A2，B1 的宝石 1，2 消失；向 A2 走去。在第 1 秒，由于 A2 的宝石已消失，无宝石可取；向 A1 走去。在第 2 秒，取走 A1 的宝石 4。全过程共取得 2 块宝石：宝石 3 和宝石 4。

[15] 题目来源：姚金宇原创题

解答 Solution

初看此题，会觉得本题在时间上的奇偶条件很迷惑人，不是可以马上就掌握其性质的，所以应着重从弄清它的性质入手。下面的引理挖掘了题目设置奇偶条件的用意。

引理5.5 只有第偶数秒才能拿到宝石。

证明：由于初始位置可以任意指定，与之后的过程不同，需要分开讨论。

情况 1：任意指定初始起点时，取到宝石。这时为第 0 秒，恰是在第偶数秒，符合命题。

情况 2：由相邻格子 A 进入一个含宝石格子 B 时，取到宝石。

反证法。反设是在第奇数秒进入格子 B 取到宝石。然而在进入格子 B 的前 1 秒，即第偶数秒时，Amber 在格子 A，这时与格子 A 相邻的格子的宝石都要消失，就是说 B 中的宝石在进入前就不存在了。这与假设矛盾，命题成立。

■

掌握了这个性质后，便可以大胆地提出如下猜想。

推论5.6 相邻格子的宝石不能都拿到。

证明：反证法。反设存在两个相邻的格子宝石都被取到。由引理 5.5 知，先取到的宝石必然是在第偶数秒取到的，这时与该宝石相邻的宝石都消失了，则应该后取到的宝石也就不可能取到了。矛盾，命题成立。

■

这个推论说明，相邻格子的宝石是矛盾的，不能共存的。基于这个事实，联想已知的一些图论模型，发现这就是独立集的性质。若把格子看成点，相邻的关系看成边，则可以得到一个点独立集的模型，再根据优化的方向，得到最大点权独立集模型。由推论 5.6，知任意一个合法的取宝石方案必然对应一个点独立集。然而能否利用最大点权独立集模型的关键是任意一个点独立集是否能对应一个合法方案，从而建立一一对关系。

定理5.7 （点独立集对应取宝石方案）给定点独立集，总存在一个行走方案使得在点独立集内的宝石都可以被拿到。

思路：这是一个构造性证明，解可能很多样，但可以自行规定一些原则来进行构造。

根据引理 5.5，首先的原则是必须是在第偶数秒进入一个在点独立集内的格子。这要通过等待 1 秒的操作来进行调整。

由于取宝石的格子可能在网格的任意一个位置，首先考虑将所有的格子都遍历一遍。先尝试一下 Z 字形的方案。

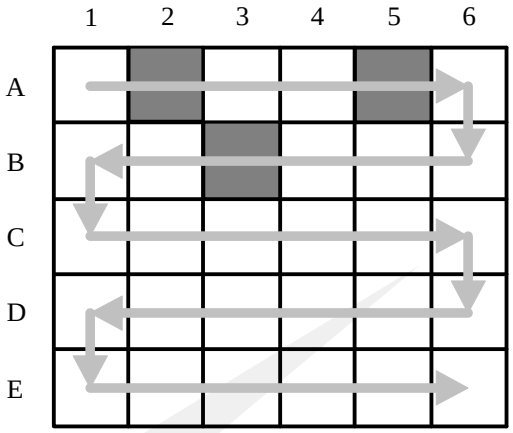

图 5.4

按照图 5.4 中的行走线路，进入 A2 时必是偶数秒，进入 A3 是奇数秒。因为 A5 是要取的，这时不能直接在偶数秒进入 A4，否则 A5 会消失。需要在 A3 停顿一下，但是在停顿的时候，将来要取的 B3 就消失了。故普通 Z 字形的方案不可行。原因是与行走方向不一致的上下行间会互相影响，不能得到调整。

实际上不一定要遍历所有格子，改造一下，以每两行为一个阶段。在每一个阶段中的第一行遍历时，就将这两行的该取的宝石全部取完，第二行不进行遍历，以免影响下一行。总的来说，遵循三个原则：

1. 按遍历方向的顺序，依次取宝石；

2. 第一行的宝石，需要在偶数秒时进入格子获取；

3. 第二行的宝石，需要在奇数秒时进入与它对应的第一行的格子，接着取第二行的宝石后，立即返回原第一行的格子，奇偶性不变。

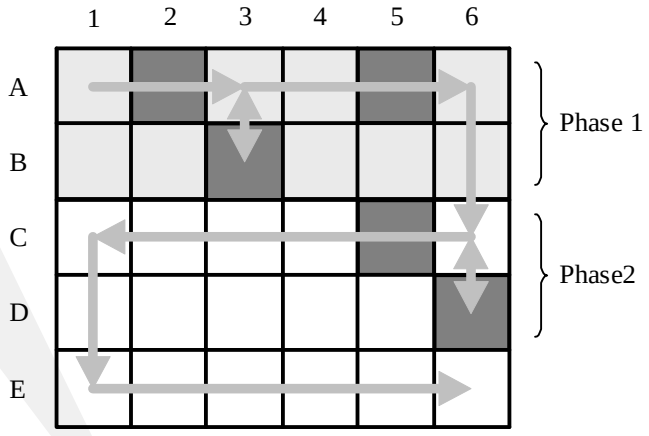

图 5.5

如图 5.5，进入 A2 时是偶数秒，进入 A3 是奇数秒。紧接着进入 B3 取宝石，再返回，这个过程用了 2 秒，奇偶性不变。因为 A5 是要取的，需要在 A3 停顿一下，在停顿的时候，由于 B3 的宝石已经取过了，并没有损失。取过 A5 后，应在奇数秒时进入 C6，以便取到 D6。取完 D6，还是在奇数秒时进入 C6，C5 的宝石没有损失。

证明：下面证明根据以上 3 个原则构造的行走方案，是可以取到独立集中的所有点的。

首先，阶段间不会互相影响。由于上一个阶段已经取完，在当前阶段的行走时，不会影

响上一个阶段。由于当前阶段只有在第二行有独立集点时，才进入第二行，根据推论 5.6，下一阶段的第一行同样位置必然没有需要取的点。故不会影响下一个阶段。

其次，在当前阶段内，必定存在一种操作序列使得独立集中的点都被取到。用序列考察在第一行的行走线路，对于每个格子，以 0 表示必须在偶数秒进入，以 1 表示必须在奇数秒进入，用“?”表示待定。根据原则 2，将第一行的宝石处标上 0；根据原则 3，将第二行的宝石的上一行处标上 1；剩下的格子均为“?”。

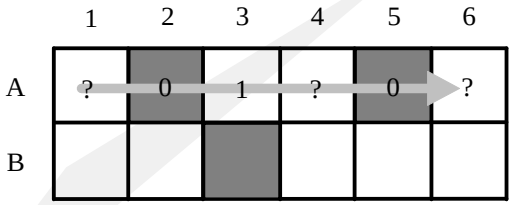

图 5.6

对于一段连续的“?”，容易构造操作序列，总是可以等价于单个“?”。对于一段连续的非“?”，根据推论 5.6，该序列一定不存在连续的两个 0 或 1，必然是 01 相间的形式，依然容易构造操作序列。问题集中在非“?”段与“?”的衔接上，下面分别讨论。

对于“…0？0…”的形式，令“？”为 1 即可，即“…0 1 0…”；

同样地，对于“…1？1…”的形式，令“？”为 0 即可，即“…1 0 1…”；

对于“…0？1…”的形式，则先取完 0 所代表的格子内的宝石，在该格子停留 1 秒，此时并不会改变其他格子的情况，再令“？”为 0，即“…0(1) 0 1…”；

同样地，对于“…1？0…”的形式，也是先停留再前进，令“？”为 1，即“…1(0) 1 0…”。

综上，每个阶段的操作序列均可构造，各个阶段互不影响，依次进行，所以该操作序列可以取到点独立集的所有点。

■

经过这个构造性证明，建立了独立集与操作序列的一一对关系，这样就成功地将该问题转化为最大点权独立集问题。由于这是一个平面网格形成的图，有其特殊性：它是一个**二分图**。将网格黑白相间染色，所有边的端点必然不同色。这样所有黑格作为一个部，所有白格作为一个部，便是一个二分图。接下来使用5.3 节所描述的解法解决即可。

6. 总结 Summary

最小割是最大流这一直观问题的对偶问题，但它相对来说比最大流问题抽象。它的构图，往往不容易经由一般物理经验推导出，本文只是着重分析了四类问题用最小割解决的方法。然而题目是千变万化的，更重要的是掌握分析方法与技巧，才能以不变应万变。这些方法在以上章节的分析过程中都有体现，现将它们总结如下。

6.1. 转化过程的模式 Transforming Pattern

在以上的章节中，所有问题都是化归到最小割模型来解决的。建立原问题与最小割的一一对应关系是化归方法的必要步骤。将原问题的决策方案与具有特定性质的割，**构造性地**一一对应起来，并且保证具有该特定性质的割一定能被最小割取到。

如3.3 节中将简单割与闭合图方案一一对应起来，并且最小割一定是一个简单割。这样就成功地将模型进行了转化。

6.2. 割的性质 Property of Cut

需要熟悉割的性质，才能在解题中进行合理联想与类比。所谓“灵感”便是源自性质的相似性。

性质 1（不连通）在给定的流网络中，去掉割的边集，则不存在任何一条从源到汇的路径。

在5.2 节中，由于性质 1，任意一条路径 $s-u-v-t$，都会被割截断，即至少有一条边在割中。根据点覆盖集的定义，$u\in V'$，$v\in V'$ 至少一个成立，就是说这两个对象至少有一个被取。这就是性质的相似性。利用技巧 1，很容易将点覆盖集化归为割。

性质 2（两类点）在给定的流网络中，任意一个割将点集划分成两部分。割为两部分点之间的“桥梁”。

在应用 2.2.2中，难点是将难以处理 xor 操作转化为了割。割将点划分成了两类，而 xor 操作就是计算这两类点之间的“桥梁”的个数。这样就利用了性质 2，进行了合理类比，解决了问题。

6.3. 技巧 Skills

技巧 1 用正无限容量排除不参与决策的边。

在5.2 节中，为了化归点覆盖集，任意源到汇的路径 $s-u-v-t$ 中，其中 $\langle u,v\rangle$ 不参与决策，这样就可以令该边的容量为正无限 $c(u,v)=\infty$，使其不会被选入最小割中。

技巧 2 使用割的定义式来分析最优性。

由于割在一些特殊的图中很抽象，不易想象，最好的方法就是从定义定量分析。在4.5 节定理 4.5 的证明与3.3 节定理 3.4 的证明中，都运用了割的定义式来展开结果。

技巧 3 利用与源或汇关联的边容量处理点权。

在第 3 节最大权闭合图的网络构造与第 5 节的最小点权覆盖集的构造中，点权都转移到了边权上，以便充分利用最小割这个**最优化工具**来决策点权的取舍。

7. 感谢 Acknowledgments

首先感谢我的指导老师——福州第一中学的陈颖老师。

感谢国家集训队教练——刘汝佳教练。

感谢雅礼中学的朱全民老师。

特别感谢越南胡志明市教育大学的 Thanh Vy。从题源与参考文献的寻找，到解题的讨论，她给予了我非常多的帮助。若没有她在图论方面的指导，我根本无法想象我可以顺利地完成这篇论文。

感谢长郡中学的郭华阳，感谢他提供了他的原创题和源程序。

感谢雅礼中学的姚金宇，感谢他提供了他的原创题和参考解答。

感谢上海市延安中学的王欣上，感谢他提供了 Dinic 算法的参考。

感谢西安一中的杨哲、芜湖一中的周东、安徽师大附中的杨弋和福州三中的杨沐，在平时训练和讨论中，他们给了我很多启发和帮助。

感谢雅礼中学雷涛、郑嬰、陈雪、张弛、郑泽宇和陈丹琦，在与他们的训练中，我学会了许多以前从未涉及的竞赛技巧。其中张弛还帮助我整理题目描述。

感谢那些曾经帮助过我的俄罗斯朋友：

Petr Mitrichev from Moscow State University

WiNGeR from SPbSU ITMO

Dmitry 'Diman_YES' Kovalioff

Vladimir Yakovlev from USU

GaLL from Tyumen SU

每一封求助邮件，他们都认真地给予回复，真的非常感谢。

感谢福州一中 2006 届的梁晓菲和谢凌曦，2007 届的李昱东，2008 届的肖汉骏，余林韵和刘勤。

8. 附录 Appendix

8.1. 涉及题目列表 Problem List

题目名称	测试途径	题目来源	源代码	测试结果
Network Wars	ZJU 2676	Andrew Stankevich's Contest #8	ZJU_2676.dpr	Accepted
Optimal Marks	SPOJ 839 OPTM	Guo Huayang's Original Problem	OPTM.dpr	Accepted
最大获利 Profit	Testdata	NOI 2005 Day 2	Profit.dpr Profit_2.dpr	TLE 2 Tests Accepted(0.71s)

Hard Life	PKU 3155	ACM/ICPC Northeastern Europe 2006	Hard.dpr Hard_2.dpr	Accepted Accepted
Destroying The Graph	PKU 2125	ACM/ICPC Northeastern Europe 2003, Northern Subregion	PKU_2125.dpr	Accepted
Exca	N/A	Yao Jinyu's Original Problem	Exca.dpr	Accepted

8.2. 基本图论定义与术语 Basic Definition and Glossary in Graph Theory

8.2.1. 图与网络 Graph and Network

二元组 (V,E) 称为**图**(graph)。V 为**结点**(node)或**顶点**(vertex)集。E 为 V 中结点之间的边的集合。

点对 (u,v) 称为**边**(edge)或称**弧**(arc)，其中 $u,v\in V$，称 u，v 是**相邻**的(adjacent)，称 u，v 与边 (u,v) 相**关联**(incident)或相邻。

若边的点对 (u,v) 有序则称为**有向**(directed)边，其中 u 称为**头**(head)，v 称为**尾**(tail)。所形成的图称为**有向图**(directed graph)，意即——对于 u 来说，(u,v) 是**出边**(outgoing arc)；对于 v 来说，(u,v) 是**入边**(incoming arc)。反之，若边的点对无序则称为**无向**(undirected)边，所形成的图称为**无向图**(undirected graph)。

若图的边有一个**权值**(weight)，则称为赋权边，所形成的图称为**赋权图**(weighted graph)或**网络**(network)。用三元组 $G(V,E,W)$ 表示网络。其中 W 表示权集，它的元素与边集 E 一一对应。在**流网络**(flow network)中，权集 W 又记作 C，表示**容量**(capacity)。

8.2.2. 图的术语 Glossary of Graph

简单图(simple graph)：没有环、且没有多重弧的图称作简单图。

邻域(neighborhood)：在图中与 u 相邻的点的集合 $\{v\mid v\in V,(u,v)\in E\}$，称为 u 的邻域，记为 $N(u)$。

度：

度(degree)：一个顶点的度是指与该边相关联的边的条数，顶点v的度记作$\deg(v)$或d_v。

握手定理：无向图：$\sum_{v\in V}\deg(v)=2|E|$；有向图：$\sum_{v\in V}\deg^+(v)=\sum_{v\in V}\deg^-(v)$。

入度(indegree)：在有向图中，一个顶点v的入度是指与该边相关联的入边(即边的尾是v)的条数，记作$\deg^+(v)$。

出度(outdegree)：在有向图中，一个顶点v的出度是指与该边相关联的出边(即边的头是v)的条数，记作$\deg^-(v)$。

孤立点(isolated vertex)：度为 0 的点。**叶**(leaf)：度为 1 的点。

源(source)：有向图中，$\deg^+(v)=0$的点。**汇**(sink)：有向图中，$\deg^-(v)=0$的点。

子图：

子图(sub-graph)：G'称作图G的子图如果$V(G')\subseteq V(G)$以及$E(G')\subseteq E(G)$。

点导出子图(induced subgraph)：设$V'\subseteq V(G)$，以V'为顶点集，以两端点均在V'中的全体边为边集所组成的子图，称为G的由顶点集V'导出的子图，简称为G的点导出子图，记为$G[V']$。

点集的补集：记 $\overline{V'}=V-V'$为点集V'的补集。

特殊的图：

零图(null graph)：$E=\varnothing$，即只有孤立点的图。n阶零图记为N_n。

二分图（bipartite graph）：若图G的顶点集可划分为两个非空子集X和Y，即$V=X\bigcup Y$且$X\bigcap Y=\varnothing$，且每一条边都有一个顶点在X中，而另一个顶点在Y中，那么这样的图称作二分图。

8.3. 《怎样解题表》 How to Solve it

《怎样解题表》是 Polya 的数学教育书系列 *How to solve it*（参考文献[Pol57]）中的精华，很好地阐释了数学问题的一般性解决思路。原文摘引如下：

1. UNDERSTANDING THE PROBLEM

First. You have to understand the problem.

- What is the unknown? What are the data? What is the condition?
- Is it possible to satisfy the condition? Is the condition sufficient to determine the unknown? Or is it insufficient? Or redundant? Or contradictory?
- *Draw a figure. Introduce suitable notation.*
- Separate the various parts of the condition. Can you write them down?

2. DEVISING A PLAN

Second. Find the connection between the data and the unknown. You may be obliged to consider auxiliary problems if an immediate connection cannot be found. You should obtain eventually a plan of the solution.

- Have you seen it before? Or have you seen the same problem in a slightly different form?
- Do you know *a related problem*? Do you know a theorem that could be useful?
- Look at the unknown! And try to think of a familiar problem having *the same or a similar unknown*.
- Here is a *problem related to yours and solved before*. Could you use it? Could you use its result? Could you use its method? Should you introduce some auxiliary element in order to make its use possible?
- Could you *restate* the problem? Could you restate it still differently? Go back to definitions.
- If you cannot solve the proposed problem try to solve first some related problem. Could you imagine a more accessible related problem? A *more general* problem? A *more special* problem? An *analogous* problem?
- Could you solve a part of the problem? *Keep only a part of the condition*, drop the other part; how far is the unknown then determined, how can it vary? Could you derive something useful from the data? Could you think of other data appropriate to determine the unknown? Could you change the unknown or data, or both if necessary, so that the new unknown and the new data are nearer to each other?
- Did you use all the data? Did you use the whole condition? Have you taken into account all essential notions involved in the problem?

3. CARRYING OUT THE PLAN

Third. Carry out your plan.

- Carrying out your plan of the solution, check each step. Can you see clearly that the step is correct? Can you prove that it is correct?

4. LOOKING BACK

Fourth. Examine the solution obtained.

- Can you check the result? Can you check the argument?
- Can you derive the solution differently? Can you see it at a glance?
- Can you use the result, or the method, for some other problem?

翻译如下：

1. 弄清问题

首先你必须弄清问题

- 未知数是什么？已知数是什么？条件是什么？
- 满足条件是否可能？要确定未知数，条件是否充分？或者多余？还是矛盾？

- **画一张图，使用恰当的符号**。
- 理清不同的条件，试着把它们都写下来。

2. 拟订计划

找出已知数与未知数之间的联系。如果没有直接的联系，就必须先考虑辅助性的问题。最终你应该得到一个求解计划。

- 你以前见过它吗？你是否见过相同的或形式稍有不同的问题？
- 你是否知道**与此有关的问题**？或者一个可以用得上的定理？
- 看着未知数，试着想出一个**有相同或相似未知数的熟悉问题**。
- 如果有一个与现在的问题有关并且早已解决的问题，你能否利用它？能否利用它的结果或方法？为了利用它是否应该先引入某些辅助元素？
- 你能否**重新叙述**这个问题，尽可能地从不同的角度？很多时候你必须回到定义中去。
- 如果你不能解决所提出的问题，可以尝试先解决一个与此有关的。你能否提出一个更容易着手的相关问题——像是一个**更普遍**的或者更特殊的，或者一个**类比**的问题？
- 你能否解决这个问题的一部分？仅仅**保留条件的一部分而舍弃其余**，这样对于未知数能确定到什么程度？它还能怎样变化？你能否从已知数据推导出某些有用的信息？你是否考虑过用其它数据来确定未知数？如果需要的话，你能否转化未知数或数据（或者二者同时），以使得新未知数和新数据联系更紧密？
- 你是否利用了所有的已知数据？你是否利用了全部的条件？你是否考虑了问题中包含的所有基本概念？

3. 实行计划

实行你的计划

- 实现你的解题计划，**检查每个步骤**。你能否清楚地看出这一步骤的正确性？你能否证明？

4. 回顾

验算所得到的解

- 你能否验算这个解？能否解决争议？
- 你能否用**别的方法得到这个解答**？或者你其实能够一眼就看出它来？
- 你能否把本题的结果或方法应用于其它的问题？

9. 参考文献 References

[1] [**Gol84**] A.V. Goldberg. *Finding a Maximum Density Subgraph*. Technical report UCB CSD 84/171. University of California, Berkeley, CA, 1984

[2] [**AMO93**] R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. New Jersey: Prentice-Hall, 1993

[3] [**CLRS01**] Thomas H. Cormen, Charles E. Leiseison, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithm, Second Edition*. MIT Press and McGraw-Hill, 2001, Section 26

[4] [**MSS92**] T. Matsui and Y. Saruwatari and M. Shigeno. *An Analysis of Dinkelbach's Algorithm for 0-1 Fractional Programming Problems*. Technical Report METR92-14, Department of Mathematical Engineering and Information Physics, Unversity of Tokyo,1992

[5] [**Din70**] E. A. Dinic. *Algorithm for solution of a problem of maximum flow in a network with power estimation*. Soviet Math, 1970, Doklady Vol 11: 1277-1280

[6] [**Dink67**] Werner Dinkelbach. *On Nonlinear Fractional Programming*. Management Science, Vol. 13, No. 7, Series A, Sciences. (Mar., 1967), pp. 492-498.

[7] [**Pol57**] G. Polya. *How to Solve It, Second Edition*. Princeton University Press, 1957

[8] [**ZCW03**] 张宪超, 陈国良, 万颖瑜. 《网络最大流问题研究进展》. 计算机研究与发展, Vol 40, No 9, 中国科学技术大学计算机科学与技术系, 2003

[9] [**WW02**] 吴文虎, 王建德. 《信息学奥林匹克竞赛指导—图论的算法与程序设计(PASCAL 版)》, 清华大学出版社, 2002 年 8 月

[10] [**LiuH04**] 刘汝佳, 黄亮. 《算法艺术与信息学竞赛》. 清华大学出版社, 2004

[11] [**Jiang01**] 江鹏.《从一道题目的解法试谈网络流的构造与算法》. 2001 年国家集训队论文