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**

引入

NOI 2006 最大获利

- 两个部分
  - 最小割是最大流的对偶问题。
  - 平直观，模型隐蔽。
- 展示最小割模型应用的巧妙构图方法和独特思维方式
  - 标准解答的一个更一般化的扩展模型
  - 达到用最大流解决该问题的理论下界

1.最大权闭合图

2.改进算法

网络流首次进入 NOI

NOI 2006 最大获利 (Profit) 问题描述

简要描述

- 有 n 个结点， m 条无向边可供建设。
- 建立一个结点 u 有一定的花费 p_u。建立一条无向边有一定的非负收益 w_e。
- 建立一条无向边 (u, v) 的必要条件是要先建立点 u ， 点 v 。
- 求最大获利。

NOI 2006 最大获利（Profit）分析

- 目的：选出一个边集 *E'*，点集 *V'* 。且最大化：

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

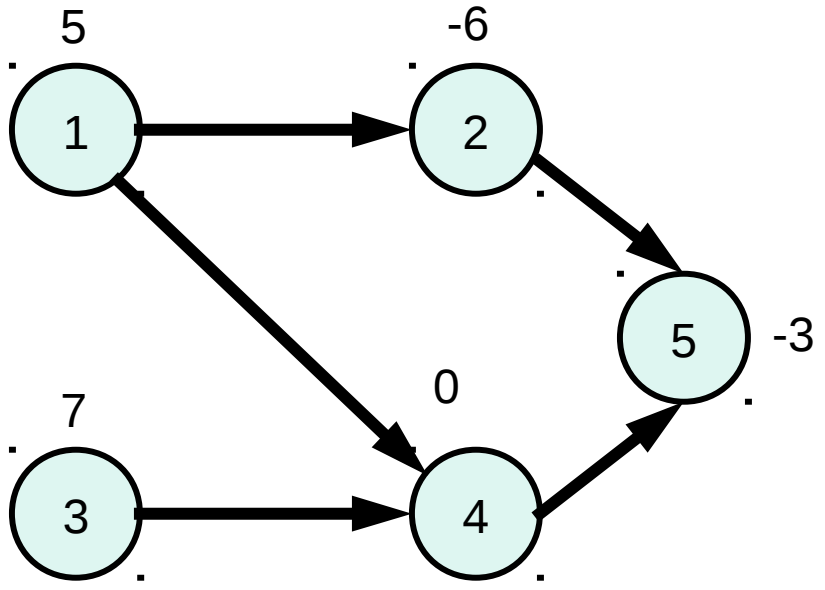
- 限制条件：对于在 *E'* 中每条边 (*u*, *v*)，它的端点 *u*，*v* 一定要在 *V'* 中。

边 —— **依赖** / **必要条件** ——→ **点**

提出解决事件依赖关系的有力图论工具：闭合图。

最大权闭合图
定义

- 有向图的闭合图（closure）：
  闭合图内任意点的任意后继也一定还在闭合图中。
  - 物理意义
    事物间依赖关系：一个事件要发生，它需要的所有前提也都一定要发生。
- 最大权闭合图是点权之和最大的闭合图。



其中{1, 2, 5}是一个闭合图，因为点1的后继是点2和5，点2的后继是5，都在闭合图中。

其中{1, 4, 5}不是一个闭合图，因为点4的后继是5，但不在闭合图中。

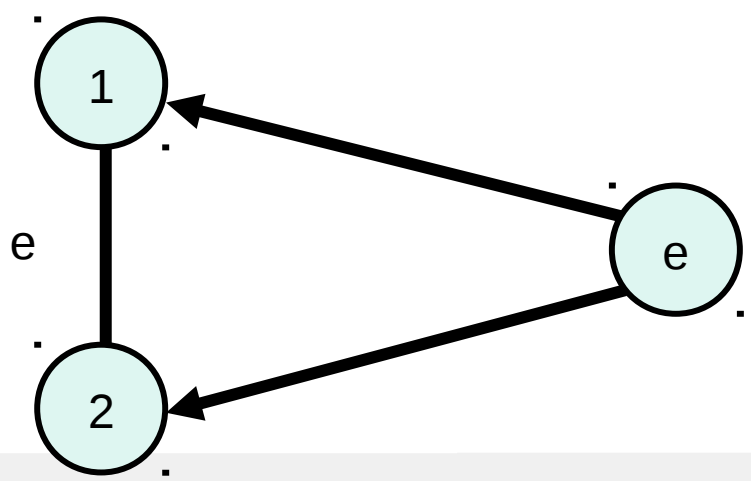
最大权闭合图
解决

解法略去

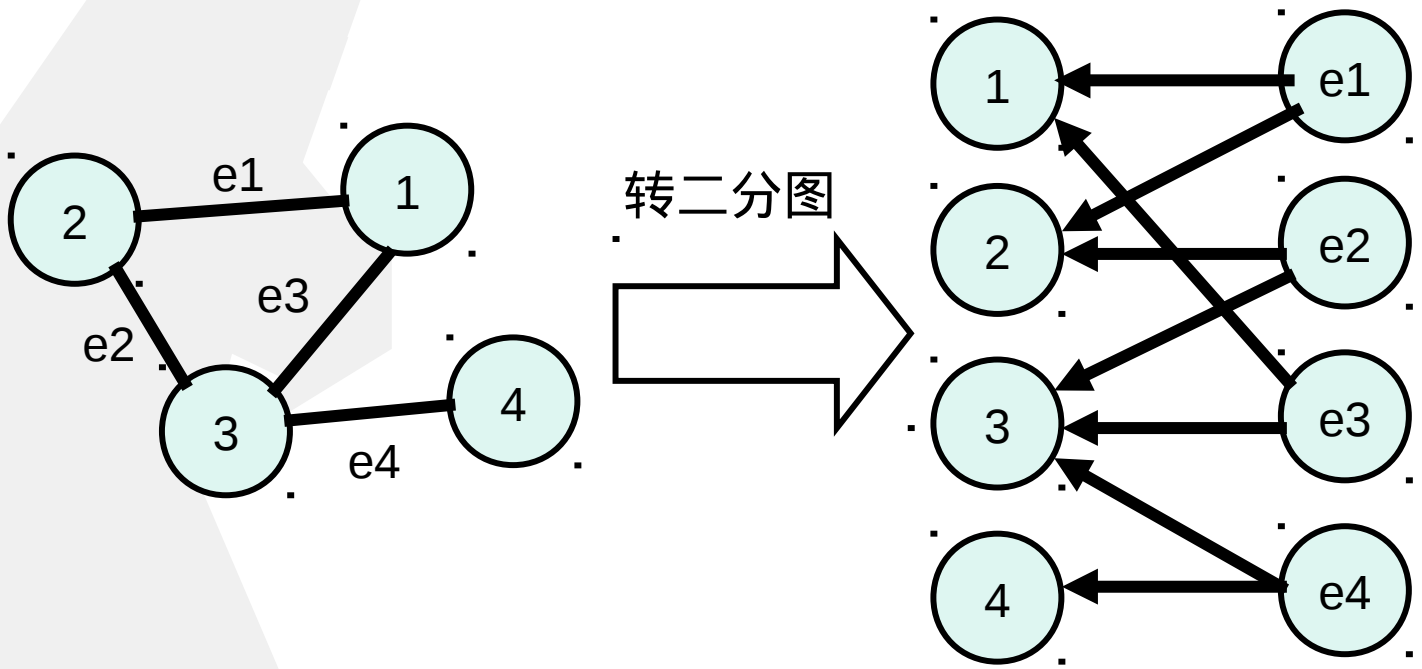
复杂度为 $O(\text{MaxFlow}(n, n+m))$

NOI 2006 最大获利 (Profit) 标准算法

- 将原题中的边和点都看成事件。边事件依赖边的两个端点事件的发生。这与闭合图的性质相似。
- 构造性地，将边转化为点事件。



NOI 2006 最大获利 (Profit) 标准算法



- 将所有边都转化为事件点，原图便转化为一个二分图。这样新构造的二分图的闭合图就对应了原问题的一个解。解决该二分图的最大权闭合图即可

复杂度为 $O(\text{MaxFlow}(n+m, n+m))$

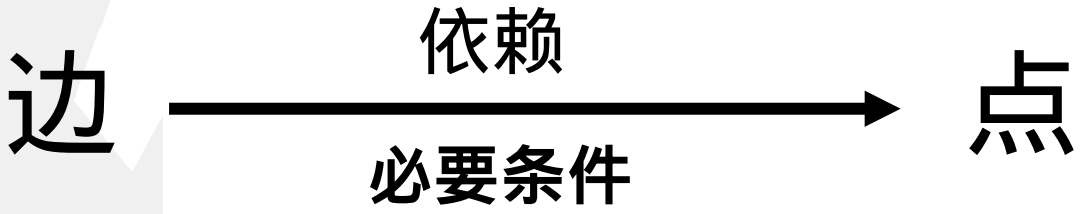
最大权闭合图
小结

- 在任意带权有向图中，只要有依赖关系需要解决，最大权闭合图都普遍适用。（普适性）

- 在最大获利的解决方法 1 中，最大权闭合图来解决二分图模型。（特殊性）

改进算法提出

正向思维（被动）



逆向思维（主动）

重定义

两个端点都在点集 V' 里的所有边组成了边集 E'
即 V' 的导出子图。

改进算法分析

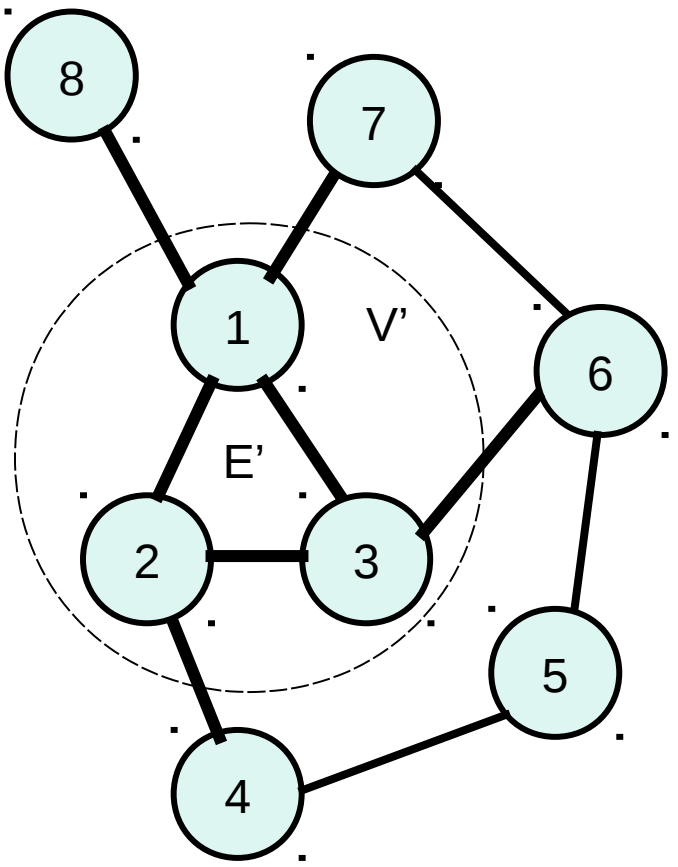
- 先选点集 *V'*
- 再找 *V'* 之间的边集 *E'*

补集转化
再次逆向思维

V' 间的边 *E'* → 最大化

= 与 *V'* 关联的所有边

- *V'* 与 *V'* 补集之间的边 → 最小割

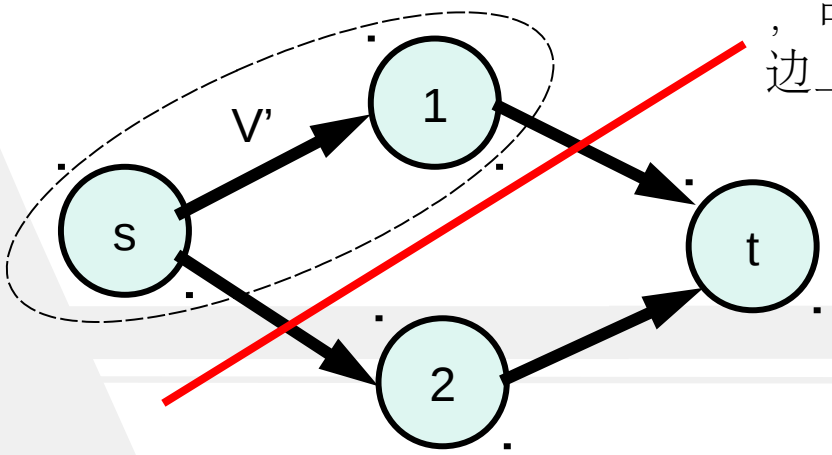


- 圈内的点组成 *V'*
- 蓝边组成 *E'*
- 红边组成 *V'* 与 *V'* 补集之间的边

改进算法
尝试构图

- 选出点集 *V'*
  - 对于每个点：选或不选
- 构图
  - 从源向每个点连边
  - 从每个点向汇连边

- 对于每个点，割必会割断它到源或它到汇的两条边中的一条
- 不妨设，到汇的边被割断的点组成 *V'*
  - 则 *V'* 中每个点连接汇的边都在割内
  - 选入 *V'* 的点的一些代价信息，可以加载到这条被割掉的边上。



改进算法分析

V' 间的边 E' =

凑入最小割

- （与 V' 关联的所有边
- 与 V' 关联的所有边）

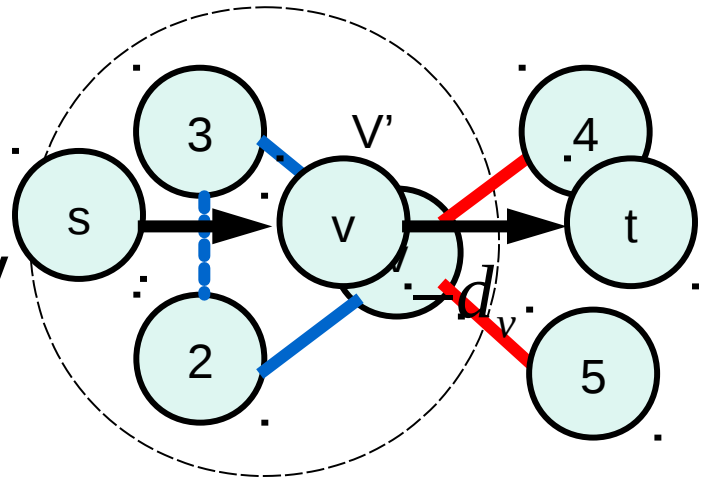
V'与V'补集之间的边

V'与V'补集之间的边

- 微观地，考察单独的在 V' 中点 v

与 v 关联所有 E' 内的边 =

- （与 v 关联所有割边
- 与 v 关联所有边）

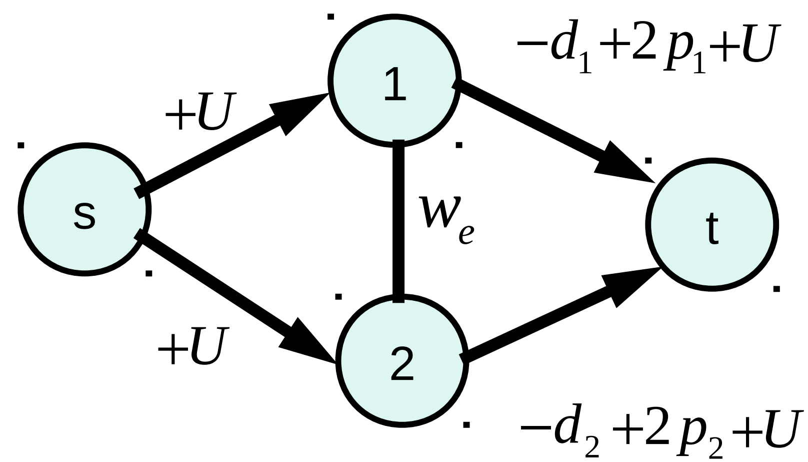


- 每个点到汇的边容量为 $-d_v$

- 令 d_v 表示与点 v 关联的总边权和

改进算法
构图

- 每个点向汇连的边的容量为 $-d_v$
- 每个点到汇的边容量增加该点点权的两倍 $2p_v$



- 由于最小割算法只能处理非负边权，故在每条边的容量加上一个足够大的数 U 即可。
- 考虑点权：

V' 间的边 + [illegible] V' 的点权 =

凑入最小割

- 最后（保留原图的边，容量即为原边权 [illegible] 与 V' 关联的所有边 - 边）V' 的点权 ）

改进算法
解决

- 通过以上公式变形，可知答案为

$$\frac{U \times n - c[S,T]}{2}$$

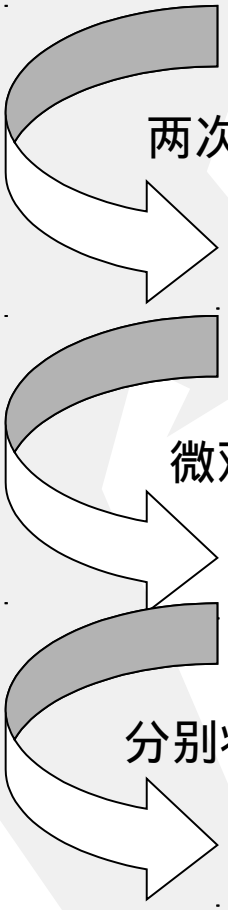
- 其中 c[S,T] 为最小割，证明从略

复杂度为 $O(\mathrm{MaxFlow}(n, n+m))$

改进算法
对比

	点数	边数	实际效果
最大权闭合图	n+m	n+m	40.41s
	↓	=	↓
改进算法	n	n+m	0.71s

改进算法
小结



改进动机

数学美

不断的完善这个想法

得出极为精妙的构造

论文特点

- 研究的重点是最小割模型的**应用**
- 不仅仅给出了结论，还着重阐述得出结论的分析过程。
  - 不仅授之以**鱼**，还授之以**渔**。
- 分析过程，是以 Polya 在数学思想方法论中的精华——《**怎样解题表**》作为贯串思维的主线。如刚才的构造过程就充分的展示了这一特点。

论文研究内容

主要研究四个方面的应用

- 基于最小割定义的直接应用
- 最大权闭合图
- 最大密度子图
- 二分图的最小点权覆盖集与最大点权独立集

•刚才所谈的例题最大获利便涉及了最大权闭合图，最大密度子图这两个方面的内容。
•其中改进算法可以作为求解**最大密度子图**的一个子过程。

论文研究内容

- Sorry for poor time.

感谢

感谢越南的Thanh Vy
感谢郭华阳提供原创题
感谢王欣上的测试实验

感谢CCF提供给我一个展示自我的舞台

谢谢大家

Thanks to you all

Amber

[ADN.cn]

hupo001@gmail.com

关于实现效率

- 本人实现的 Preflow Push
  40.41s $\Rightarrow$ 0.71s
- 王欣上提供的 Dinic 测试:
  1.7s $\Rightarrow$ 0.3s

最大权闭合图 — 证明

- 该通过对以上网络的最小割的求解，可以得到原问题的解。
- 概念：若一个割不包含正无限容量的边，称该割为简单割。最小割必是简单割。
- 闭合图 V_1 与简单割 [S, T] 间有一一对应关系

$$S = V_1 \cup \{s\}$$

因为在简单割中， S 到 T 间的边都不是正无限容量的边，即都不是原图的边。故一一对应关系成立。

最大权闭合图 — 证明

由最小割的定义，有：

$$[S,T]=[\{s\},\overline{V_1}]\cup[V_1,\{t\}]\cup[V_1,\overline{V_1}]$$

$$[\{s\},\overline{V_1}^{+}] \quad [V_1^{-},\{t\}] \quad \varnothing$$

所以得到：

$$c[S,T]=\sum_{v\in\overline{V_1}^{+}} w_v+\sum_{v\in V_1^{-}}(-w_v) \qquad \text{式 (1)}$$

最大权闭合图 — 证明

又由闭合图的定义，得到：

$$w(V_1) = \sum_{v \in V_1^+} w_v - \sum_{v \in V_1^-} (-w_v) \qquad \text{式 (2)}$$

将式 (1) 与式 (2) 加起来，得到：

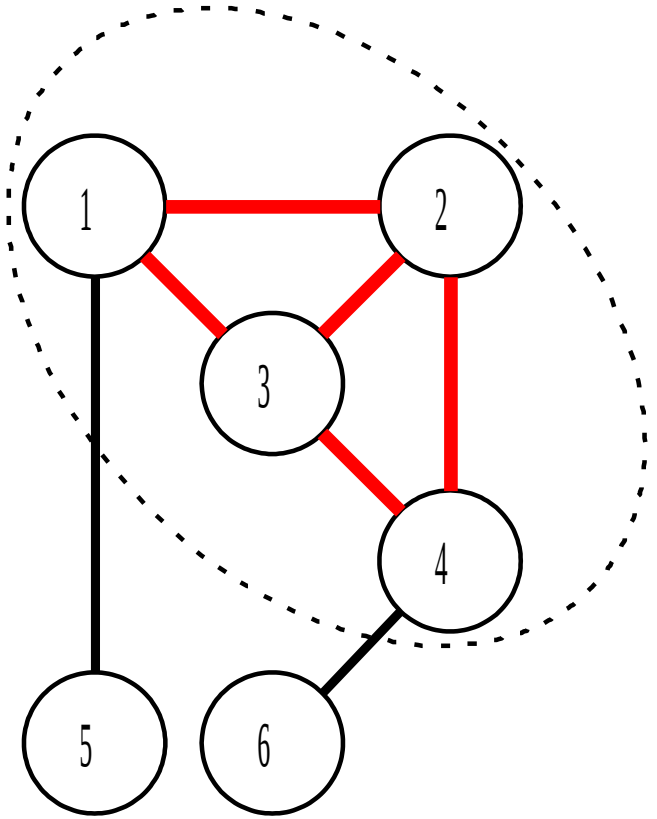
$$w(V_1) = \sum_{v \in V^+} w_v - c[S,T]$$

总复杂度为

$$O(\text{MaxFlow}(n,m))$$

最大密度子图 – 定义

- 定义一个无向图的密度 (density) 为该图的边数与该图的点数的比值
- 最大密度子图是一个具有最大密度的子图
- 由于目标是求最大，可以直接把子图重定义为的子图点集的导出子图



其中在虚线内的点与边组成最大密度子图，密度为 5/4

最大密度子图 — 主算法

- 形式化地重新叙述本模型

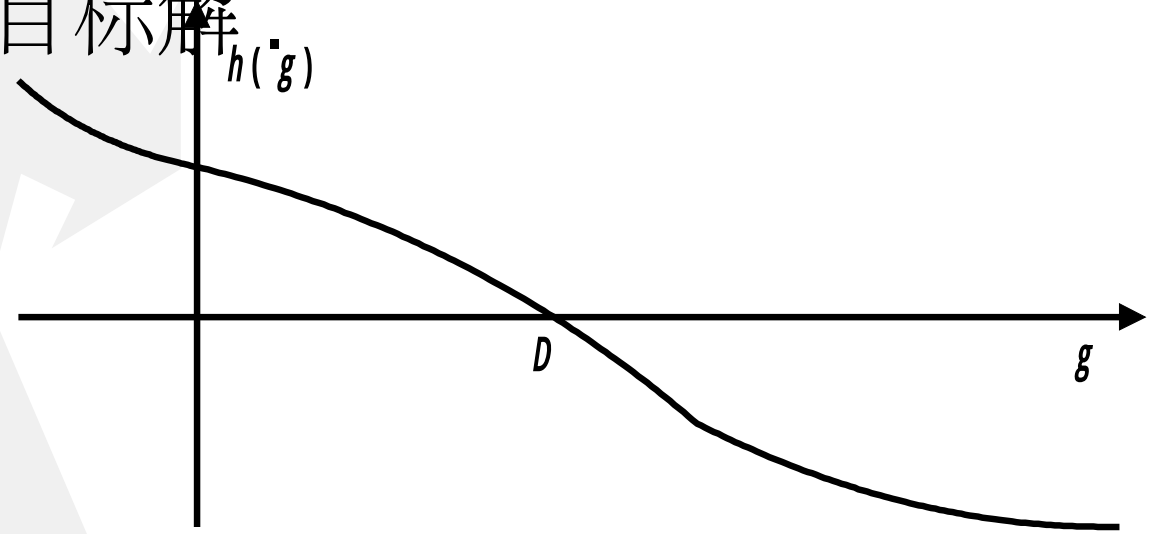
$$\text{Maximize} \quad D = f(\mathbf{x}) = \frac{\sum_{e \in E} x_e}{\sum_{v \in V} x_v} = \frac{|E'|}{|V'|}$$

- 这是 0-1 分数规划的模型
- 对答案值的二分查找，将分数规划转化为一般规划
- 对于一个答案的猜测值 g，新函数

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

最大密度子图 – 主算法

- 性质：1. 具有单调性；2. 又根据 Dinkelbach 定理，函数图像与 x 轴的交点，即为目标解。



- 对答案进行二分查找．设二分查找的次数为 k，则总复杂度为

$$O(k \times \mathrm{Solve}(h(g)))$$

最大密度子图 － 初步算法

Maximize $h(g) = |E'| - g \times |V'|$

- 基本的限制条件：边 (u, v) 存在于子图中的必要条件为点 u, v 也存在于子图中 .
- 根据这必要条件的关系，想到使用最大权闭合图的方法解决 . 依然是将边看成点即可 .

复杂度为 需要改进！！！

$$O(h(g)) = O(\text{MaxFlow}(n+m, n+m))$$

最大密度子图 — 改进算法

×（−1）

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

c

Minimize $g \times |V'| - |E'| = \sum_{v \in V'} g - \sum_{e \in E'} 1$

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

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

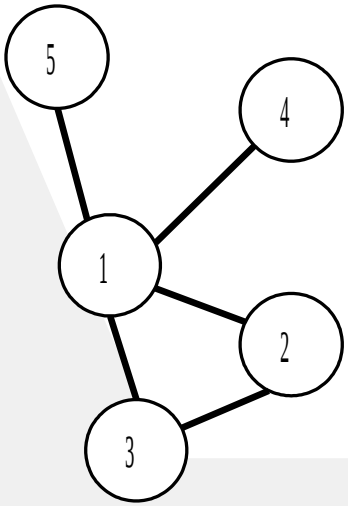
$$d_u = \sum_{(u,v) \in E} 1$$

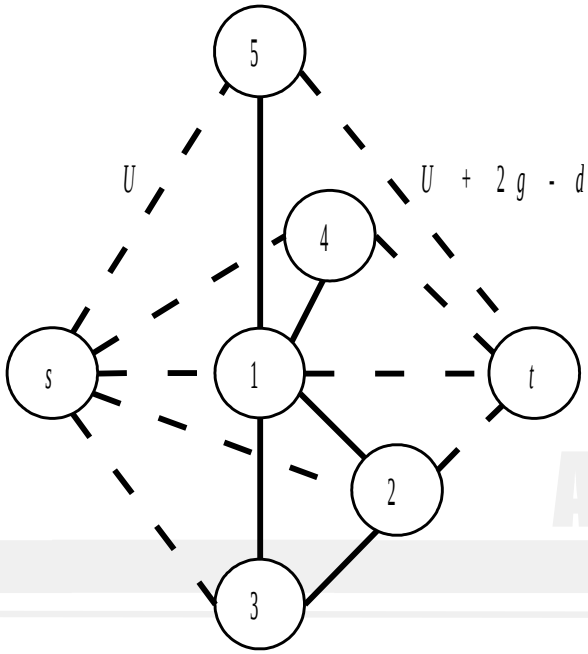
×2

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

最大密度子图 － 改进算法

- 将上面的思路整理一下
- 在原图点集的基础上增加源和汇；将每条原无向边替换为两条容量为 1 的有向边；连接源 s 到每个点，容量为 U ；连接汇 t 到每个点，容量为 $U+2g-d_v$ 。
- U 为一个足够大的数。





最大密度子图 — 改进算法

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

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

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

改进算法证明

$$h(g)=\frac{U\times n-c[S,T]}{2}$$

复杂度为

$$O(h(g))=O(\mathrm{MaxFlow}(n,n+m))$$

推广 1：改进算法的点权和边权推广

- 定义带边权无向图的密度为该图的边权和与该图的点数的比值：

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

$$c(u,v) = 1 \qquad (u,v) \in E$$

$$c(u,v) = w_e \qquad (u,v) \in E$$

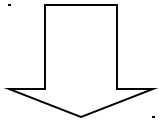
$$d_u = \sum_{(u,v) \in E} 1 \quad \Rightarrow \quad d_u = \sum_{(u,v) \in E} w_e$$

推广 2：改进算法的点权和边权推广

- 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与该图的点数的比值，即

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

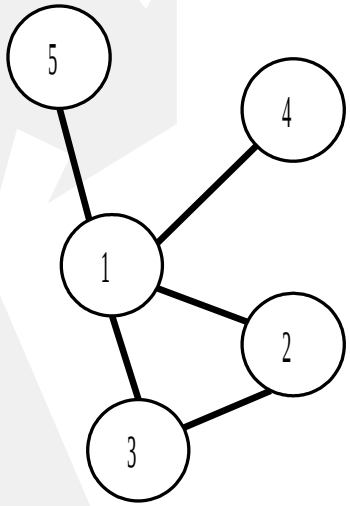
$$c(v,t) = U + 2g - d_v$$

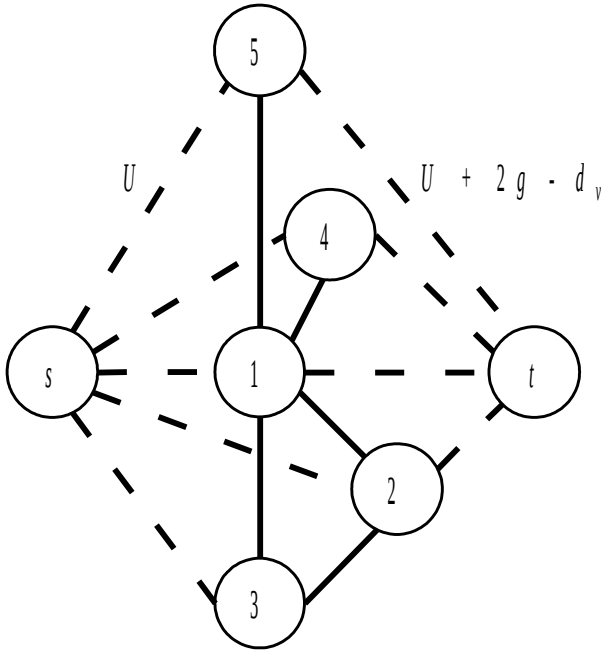


$$c(v,t) = U + 2g - d_v - 2p_v$$

重新解决 NOI 2006 最大获利 (Profit)

点有权，边也有权。想到可以利用推广 2 来解决 .





复杂度为

$$O(\mathrm{MaxFlow}(n, n+m))$$