无向图最小割

基本定义

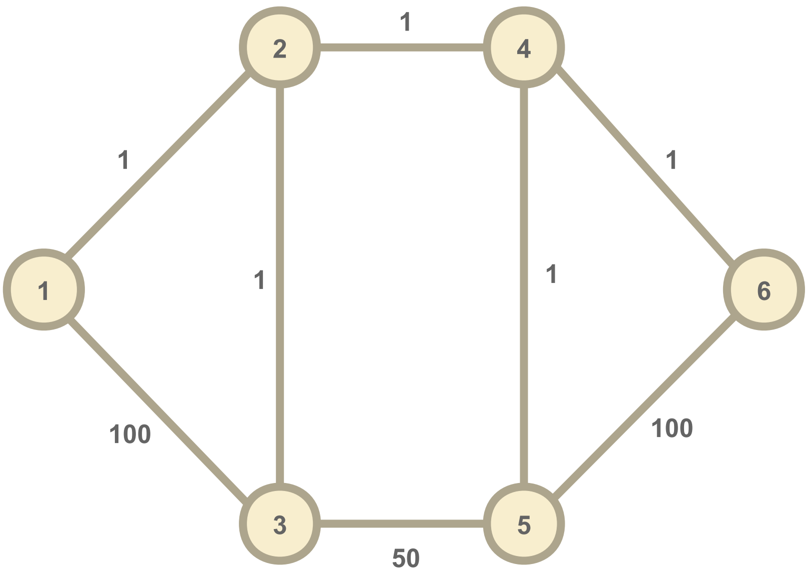
对于一张无向带权图$G=(V,E,w)$，将点集V分为两个不相交的集合S和T，满足$S\cup T=V$，那么称$C=(S,T)$是图G的**一个割**。**割的权值**定义为两个端点不在同一集合内的边的权值之和，记做$W(C)$：

$$W(C)=\sum_{u\in S,v\in T,(u,v)\in E}w(u,v) \tag{1.1}$$

对于指定的s和t，如果存在一个割$C=(S,T)$满足$s\in S,t\in T$或者$s\in T,t\in S$，那么称这个割为**s-t割**。

对于一张图，权值最小的割称为**最小割**。多数情况下，最小割不是唯一的。

例如，在下图中：



$(\{2\},\{1,3,4,5,6\})$和$(\{4\},\{1,2,3,5,6\})$均是最小割，其权值为3。而$(\{1,2,3\},\{4,5,6\})$是一个1-6最小割，其权值为51。

为了方便，我们对于任意的不相交集合U、V，要求$U\cup V\subseteq V$，定义它们之间的权值为：

$$W(U,V)=\sum_{u\in U,v\in V,(u,v)\in E}w(u,v) \tag{1.2}$$

换言之。权值是横跨U和V的边的权值之和。

现在来考虑一个比较浅显的定理：

> 对于任意四个不相交集合S_1、T_1、S_2和T_2，满足$S_1\cup T_1\cup S_2\cup T_2\subseteq V$，那么一定有：

$$W(S_1,T_1)+W(S_2,T_2)\leqslant W(S_1\cup S_2,T_1\cup T_2) \tag{1.3}$$

证明：

$$\begin{aligned}&W(S_1\cup S_2,T_1\cup T_2)=W(S_1,T_1)+W(S_1,T_2)+W(S_2,T_1)+W(S_2,T_2)\\&\because W(S_1,T_2),W(S_2,T_1)\geqslant 0\\&\therefore W(S_1\cup S_2,T_1\cup T_2)\geqslant W(S_1,T_1)+W(S_2,T_2)\end{aligned}$$

求解s-t最小割

利用最大流最小割定理，一个图的s-t最小割实际上就是将原图建成一个源点为s，汇点为t的网络上的最大流，因此可以采用各种最大流算法来解决。这里不再详细介绍。

求解全局最小割

所谓全局最小割就是无向图本身的最小割，没有限定s和t。考虑到一个s-t最小割可能成为全局最小割，所以可以想出以下的过程：

1. 令$i=1,w_0=\infty$。
2. 如果$|V|\leqslant 1$，返回$\min\{w_0,w_1,\dots,w_{n-1}\}$。
3. 随便选定两个点s和t，求出其s-t最小割，记权值为w_i。
4. 将s和t合并，即将E中的每条边的t改成s，并从V中删去t。
5. $i=i+1$，跳回第二步。

这样做的理由是：假设全局最小割为(S,T)，那么分两种情况：

1. 如果s和t分别在S和T中，那么s-t的最小割就是全局最小割。
2. 如果s和t同在S或T内，那么将s和t合并为一个点后，不会影响割的权值。

这样我们可以在$\Theta(n)$次网络流内求出最小割。但是由于网络流算法时间复杂度比较高，所以这并不是一个很优的过程。

考虑到在第三步中，我们实际上并不关心s和t是谁，我们只在意得到一个s-t最小割。那么是不是可以实现一个更加高效的方法来求出任意一个s-t最小割呢？

现在来考虑以下这个算法：

```
function ST-MINCUT(G):
    assert |V| > 1

    A = {}
    B = V
    s = 0
    t = 0
    while |B| > 0:
        p = 0
        for u in B:
            if W(A, {u}) > W(A, {p}):  // Assume that W(A, 0) = 0
                p = u

        A = A ∪ p
        B = B - {p}
        s = t
        t = p

    return (s, t)
```

该算法会返回一个二元组(s,t)，表示$(V-t,t)$是一个s-t最小割。
下面我们将证明这是对的：

> 设s和t为ST-MINCUT中A集合倒数第二个加入的点和最后加入的点，那么$(V-\{t\},\{t\})$是一个s-t最小割。

证明[1]：
我们令$S'=V-\{t\}$、$T'=\{t\}$。对于图中任意一个s-t割$C=(S,T)$，假设$t\in T$并且$s\in S$。设$t'=v_i$为T中除了t之外最后被加入A的点。令$V_i=\{v_1,v_2,\ldots,v_i\}$。此外我们注意到$V_i$实际上是ST-MINCUT算法在$G$关于$V_i$的导出子图上的运行结果。
这几个集合内的点的一种可能如下表所示：

$$\begin{array}{cccccccc} S & = & & v_2 & & \ldots & s & \\ T & = & v_1 & & t' & & & t \\ S' & = & v_1 & v_2 & v_3 & \ldots & s & \\ T' & = & & & & & & t \end{array}$$

进行归纳假设，对于$|V|=2$的图，以上结论显然成立。现在假设对于$|V|<n$的图均满足，尝试证明对于$|V|=n$的图也满足。

现在我们要证明$W(S',T')\leqslant W(S,T)$，就可以得出算法的正确性。首先注意到：

$$W(S,T)\geqslant W(V_i\cap S,V_i\cap T)+W(V_{n-1}-V_{i-1},\{t\}) \tag{3.1}$$

以及：

$$W(S',T')=W(V_{i-1},\{t\})+W(V_{n-1}-V_{i-1},\{t\}) \tag{3.2}$$

根据归纳假设，我们有：

$$W(V_{i-1},\{t'\})\leqslant W(V_i\cap S,V_i\cap T) \tag{3.3}$$

考虑到算法的A变为V_{i-1}时，选择了t'而不是t加入A集合，那么说明：

$$W(V_{i-1},\{t\})\leqslant W(V_{i-1},\{t'\})\leqslant W(V_i\cap S,V_i\cap T) \tag{3.4}$$

综上可知：

$$\begin{aligned} W(S',T') &= W(V_{i-1},\{t\})+W(V_{n-1}-V_{i-1},\{t\}) \\ &\leqslant W(V_i\cap S,V_i\cap T)+W(V_{n-1}-V_{i-1},\{t\}) \\ &\leqslant W(S,T) \end{aligned}$$

这样，我们可以通过执行$\Theta(n)$次ST-MINCUT算法来找出全局最小割。ST-MINCUT的直接实现是$\Theta(n^2)$的，如果使用二叉堆这种数据结构来实现，那么可以做到$\Theta((n+m)\log n)$的复杂度。这个算法就是Stoer-Wanger算法。

---

1. 这个证明是原论文中的证明方式，个人认为其中有一些不妥之处，因为在最后一步时，好像忽略了$w(t',t)$，导致不等式会不成立。不知道是不是个人的理解上的偏差还是确实有误，如果有看过原证明的大神希望能指点我一下。 ↩

赞 [Issue 页面]

\# NULL #

编辑 预览

GitHub 登录

可以D人了......

可以使用 Markdown 编辑 评论

Powered by Gitment

标签: 最小割 网络流 图论 组合优化

创建时间: 2017.02.12

上次修改: 2017.02.12

统计: 4539 字 / 约 18 分钟

目录

- [求解s-t最小割]
- [求解全局最小割]

数学公式渲染引擎：
- MathJax (推荐)
- KaTeX
- Mixed

效率很高，但是目前 KaTeX 容错性不强，因此使用 KaTeX 时可能会存在一些数学公式无法渲染的情

先使用 KaTeX 渲染，再使用 MathJax 渲染
vertical_align_top

RITEME.SITE

一个从不乱说话的博客...

POWERED BY

- Python Markdown
- Material Design Lite
- Tipuesearch
- MathJax & KaTeX
- Gitment

友情链接

- ruanxingzhi
- Haogram
- HJWJBSR
- MicDZ
- Linyxus
- memset0