网络流建图技巧与模型

周佳乐 2017/1/12

基本知识

求解网络流问题的算法

最大流问题

增广路算法

·Ford-Fulkerson //dfs寻找增广路

·Dinic //bfs建立分层图+dfs寻找增广路，速度较快

……

预流推进算法

~~我没学过ovo~~

费用流问题

·最短增广路算法，基本思路：在残余网络上总是沿着最短路增广. 求最短路的方法：无负权边时直接用Dijkstra求最短路，有负边时可以用Bellman-Ford、SPFA，或者用重赋权值的技巧消去负边，然后用Dijkstra求最短路(见《挑战程序设计竞赛》)

·还有一种消圈算法，大致是先求个最大流，接着不断spfa消负环，效率较低？**~~总之我不会~~**

几类问题

求解最大流

求解可行流

最x费用xx流：最(小/大)费用(最大/可行)流

对偶/互补问题

最大流 ⇔ 最小割

|最大匹配|+|最小边覆盖| = |V|

|最大独立集|+|最小顶点覆盖|=|V|

经典模型

有上下界的网络流：

1.无源汇有上下界的可行流

问题：
在一个有上下界的流网络G中，不设源和汇，但要求任意一个点i都满足流量平衡条件：$\sum f(u, i) = \sum f(i, v)$
且每条边(u, v)都满足容量限制$B(u, v) \leqslant f(u, v) \leqslant C(u, v)$的条件下，寻找一个可行流f，
或指出这样的可行流不存在

做法：
设一个附加网络G'，每条边的容量$C'(u, v) = C(u, v) - B(u, v)$，增设附加源s、汇t
对于每个结点i，从s向i连一条流量为流入边下限之和的边$c'(s, i) = \sum B(u, i)$，从i向t连一条容量
为流出边下限之和的边$c'(i, t) = \sum B(i, v)$
求G'的最大流，若最大流中，从源点流出的弧都满载，则该流对应原问题的一个可行流，否
则原问题无可行流

//实际上就是先把下界强制满流
//费用流中消除负圈也是类似的做法

有上下界的网络流：

2.有源汇有上下界的可行流/最大流

问题：

在一个容量有上下界的流网络G 中，求源点s到汇点t的一个可行流/最大流

做法：

可行流：

从t向s连一条容量正无穷的边(使得源汇点成为流量平衡的普通节点)，然后求无源汇可行流

最大流：

二分最大流量a，每次从t到s连一条弧(t, s)，其流量下界B(t, s) = a

判断改造后的图是否存在无源汇的可行流，存在可行流 <=> a<=max_a

//也可以不断找可行流增广。。

3.有源汇有上下界的最小流

与上一题类似，只是在加入弧(t, s)后二分(t, s)的容量上界C(t, s)即可

其他扩展，如最小费用最大流，方法类似，二分参变量

最小路径覆盖

- 给定有向无环图 G=(V,E)。设 P 是 G 的一个简单路（顶点不相交）的集合。如果 V 中每个顶点恰好在 P 的一条路上，则称 P 是 G 的一个路径覆盖。P 中路径可以从 V 的任何一个顶点开始，长度也是任意的，特别地，可以为 0。G 的最小路径覆盖是 G 的所含路径条数最少的路径覆盖。

建模方法

- 转化成二分图最大匹配问题
- 构造二分图，把原图每个顶点i拆分成二分图X，Y集合中的两个顶点Xi和Yi。对于原图中存在的每条边(i,j)，在二分图中连接边(Xi,Yj)。

二分图的最大独立集、最小顶点覆盖

独立集：在G中两两不相邻的顶点集合S

顶点覆盖：顶点集合S，满足G中任意一条边都至少有一个端点属于S

|最大独立集|+|最小顶点覆盖|=|V|，且X⊂V是G的独立集<=>V\X是G的顶点覆盖

证：任意e=(u, v)，对于独立集X，u和v不同时在X中，所以u和v必有一个在V\X中(充分性得证，反之亦然)

- 但最大独立集和最小点覆盖都是NP困难的，但对于二分图有下面的等式
- |最大匹配|=|最小顶点覆盖|
- 所以也就有|最大独立集| = |V| - |最大匹配|

- 设二分图G=(U∪V, E)，在通过最大流求解最大匹配得到的残余网络中，令S=(从s不可达的属于U的顶点)∪(从s可达
- 的属于V的顶点)//也就是(U中已匹配的点)∪(V中已匹配且与U中未匹配点相邻的点)，则S就是G的一个最小顶点覆盖
- 因此二分图的最大独立集、最小顶点覆盖都可以高效解决

二分图的最大点权独立集、最小点权覆盖集

·建模：

·增加源点s和汇点t，将每条二分图的边<u, v>的容量替换为+∞

·增加s到X部的点u的有向边，容量即为c(s, u)=Wu

·增加Y部到t的点v的有向边，容量为c(v, t) = Wv

·求解最小割即可

（独立集与覆盖集为互补问题）

最大权闭合子图

- 闭合图

定义一个有向图G=(V,E)的闭合子图(closure)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。

点权之和最大的闭合子图即为该图的最大权闭合子图。

还有一种等价定义为:满足对于任意<u,v>∈E,若有u∈V'成立,必有v∈V'成立

- 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

建模方法

- 增设一个超级源点和一个超级汇点
- 将原图中每条有向边替换为容量为∞的有向边
- 增加连接源点s到原图每个正权点$v(w>0)$的有向边，容量为w
- 增加连接原图每个负权点$v(w<0)$到汇点t的有向边，容量为$-w$

- 此时,最大权=正权点的总权和-最小割

- 简单证明

最大密度子图

- 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
- 即，Maximize $D = f(\mathbf{x}) = \dfrac{\sum_{e \in E} x_e}{\sum_{v \in V} x_v} = \dfrac{|E'|}{|V'|}$
- 这是0-1分数规划问题

分数规划 ~~以下是我写的笔记不保证对，建议直接看论文~~

- 分数规划的一般形式：
- 有两个连续的实值函数a(x), b(x)，问解向量x在解空间S内 a(x)/b(x) 的最值λ （保证在解空间内b(x)>0）
- 没有办法直接求解
- 解决问题的方法是进行参数搜索(parametric search)，即不断猜测答案，将优化问题转为判定问题或
- 其他的优化问题，利用单调性每次缩小搜索范围，从而逼近答案

- 以求解最小值为例：(最大值就把下面结论全部倒一下) 构造函数 $g(\lambda) = \min\{a(x)-\lambda\cdot b(x)\}$ $x\in S$ 容易看
- 出 $g(\lambda)$ 严格递减
- 设 $\lambda^* = a(x^*) / b(x^*)$ 为原规划的最优解，得到重要结论：
- //为了方便考虑，设λi=a(xi)/b(xi)，把a(xi)-λb(xi)变形为(λi-λ)·b(xi)
- $g(\lambda) = 0 \iff \lambda = \lambda^*$ (这个0在x=x*时取到，其他的λi-λ*>0)
- $g(\lambda) < 0 \iff \lambda > \lambda^*$ (当x=x*时λi-λ就已经能取到负数)(不过最小化g(λ)的x不一定是x*)
- $g(\lambda) > 0 \iff \lambda < \lambda^*$ (对于所有λi，都有λi>λ*>λ)

- 因此，解决分数规划问题等同于找出λ=λ*使得g(λ)=0.有了这个结论，就可以对最优解λ*进行二分查找
- 逼近，每次需要计算g(λ)，而g(λ)不是分数规划，就可以想办法求解，这就把问题化简了
- 算法复杂度：二分复杂度*解决g(λ)的复杂度

0-1分数规划

- 0-1分数规划(0-1 fractional programming)
- 分数规划的一个特例，解向量x满足任意$x_i \in \{0,1\}$，求 $f(x) = (\sum a_i \cdot x_i) / (\sum b_i \cdot x_i)$ 的最值

- 经典例题：最优比率生成树

最大密度子图

- 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

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

具有单调性，与x轴的交点即为目标解

↑发现这个问题可以用最大权闭合子图来解决，具体优化见论文

~~总之就是这样~~

参考资料

- http://wenku.baidu.com/link?url=byEDv4rsueYcD1JODHKRrc_oihkAwh40_IhfcYjweh8ZCQW_j0_4VcP_QduvFZmQ_NFyeQCDSYxERcKQjY0AEEkOWlyxcRXIZu_6JT36SaG