南京邮电大学

题　　目	面向连通性和匹配问题的高效图算法研究
专　　业	计算机科学与技术
学生姓名	凌彦达
班级学号	B09040324
指导教师	陈　志 副教授
指导单位	计算机学院软件工程系

日期： 2013 年 3 月 18 日至 2013 年 6 月 21 日

摘　要

图算法是一种通过抽象事物之间的关系进行建模进而求解问题的方法。本论文主要研究图的双连通性、最小割以及二分图的最大匹配和最优匹配等问题。

图的双连通性反应连通性的强度，判断双连通关键是对割点的求解。本文应用 tarjan 算法求解割点，在线性时间内搜索出所有割点，将双联通图模型与词义消歧相结合，通过对词义网络建模求解关键词义，较快解决了词义消歧关键词提取的问题。求解最小割问题有最短增广路和预流的两类算法，本文对其中四种主流算法进行效率分析，通过不同的实际模型比较各个算法的优劣，针对不同的情况得出了相应的结论。二分图模型在匹配问题中有广泛应用，本文就二分匹配在社交网络中的最大匹配和最优匹配应用做了相应的分析，通过模型转换从问题中抽象出二分图模型，采用 KM 算法和网络流算法对模型进行求解。

关键字 双连通性；最小割；最大流；二分图匹配；词义消歧；网络流

目 录

第一章 绪论

1.1 研究背景

我们在生产生活中经常会遇到和需要解决一些和“关系”有关的问题。这些“关系”或者表示了各个量和事物间的联系，或者对各个量之间进行了限定和制约。而图论就是用来解决这一类问题的一种很好的方式。在图论中用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。这样我们就把原来错综复杂的问题转换成一个很简明抽象的模型，进而把研究的重心转移到对图论模型上来。图论算法就是在已有的图论模型的基础上，对这些图的性质进行研究，以找到最准确高效的方式解决问题的算法。在本文中，我们仅就图论中的双连通性、最小割以及二分图的匹配这三个小问题展开相关的研究和讨论。

对于词义消歧的研究国内外已经做了大量的工作，不过由于这个课题太大太深，现在任然有许多可供研究的领域，对于双连通性在词义消歧中的应用笔者还没有找到这方面的先例。对于最小割的研究已经比较成熟，而对于各种算法效率的分析还没有统一的结论

1.2 研究问题概述

图的双连通性属于图的连通性领域的一个子问题，对于无向连通图中的相关问题的求解具有意想不到的效果。

最小割模型具有很强的转换性，最大流问题、最大匹配、最大独立集问题、最小点权覆盖等等问题都可以转换为最小割模型。而对于最小割的求解，我们通常使用网络流的算法。

匹配问题同样是图论中的一个非常重要的问题。然而一般图的匹配是一个 NP 问题，这里我们仅对二分图的最大匹配和最优匹配展开研究。

1.3 主要研究方向

在图的双连通问题中，我们主要尝试将双连通性应用到自然语言处理的无监督词义消歧中去，在文献中已经提到通过对图的连通性研究设计相关的词义消歧算法，不过没有就双连通性展开讨论，这也给我们提供了一个研究的方向。

最小割问题通常使用网络流算法来求解。而网络流的算法主要分为两大类：基于增广路的算法和基于预流推进的算法。每种算法由于实现的手法不同，又有着诸如 ISAP 算法，Dinic 算法等等相关的网络流算法。在本文中我们主要过对这些算法进行分析和研究，比较他们在不同模型上的效率，以便在实际的问题求解中选取最高效的方法。

二分图匹配问题在实际生活中有着很多的应用，而二分图匹配也可以转换成

若干模型，这里我们主要对社交网络中的相关问题进行分析，并尝试把它们转换成二分图的模型进行求解。

1.4 理论基础

1.4.1 双连通性

双连通性主要涉及割点和割边的概念。如果在图 G 中去掉一个顶点(自然同时去掉与该顶点相关联的所有边)后图的连通分支数增加，则称该顶点为 G 的割点(cut-vertex)[1]。在一个无向连通图中，如果有一个边集，删除这个边集后原图变成多个连通块，就称这个边集为割边集合[2]。一个图的最小割点集合中的顶点数称为点连通度[2]。一个图的最小割边集合中的边数称为边连通度[2]。如果一个无向连通图的点连通度大于 1，则称该图是点双连通的，简称双连通。如果一个无向连通图的边连通度大于 1，则称该图是边双连通的，简称双连通。

1.4.2 最小割

最小割模型是建立在网络图之上的。有向加权图 G 中指定两个顶点 S 和 T, 分别称为源点和汇点。边 e 上的权值称为边 e 的容量，又设此有向图为一个严格有向图，则称该有向图为一个网络[3]。网络流中的流是一个实值函数 f 且满足三个性质：

（1）容量限制：对于∨u，v 属于 V，要求 f(u,v)≤c(u,v)

（2）反对称性: 对于∨u，v 属于 V，要求 f(u,v)=-f(v,u)

（3）流量守恒：对于∨u 属于 V-{s,t}，要求∑f(u,v)=0

称 f(u,v)为从 u 到 v 的流[4]。

进一步的，对于网络 G=(V,E,C)，设流 f 是 G 中的流。对于 G 中的每条边<u,v>，可以定义残留网络为在不超过容量限制的条件下，可以通过的额外的网络流量 Cf(u,v)=c(u,v)-f(u,v)称为残量网络。

网络 G=(V,E,C)中的割[S,T]将点集 V 划分成 S、T(S=V-T)两个部分，使得 s 属于 S 且 t 属于 T。符号[S,T]代表边集合{<u,v>|<u,v>∈E,u∈S,v∈T}。穿过割[S,T]的净流量定义为 f(S,T)，割[S,T]的容量定义为 c(S,T)。一个网络的最小割也就是该网络中的容量最小的割。

1.4.3 二分图匹配

对于一个边集 M∈E(G)，∨ei,ej∈M,ei,ej 不存在公共顶点，则 M 是图 G 的一个匹配，M 中的每个端点称为被 M 匹配，而其余顶点式未被 M 匹配，若图 G 中的每个顶点都被 M 匹配了，则称 M 是图 G 的一个完美匹配[5]。图 G=(V,E)的一个极大匹配是一个不能再通过添加边来是其变大的匹配[6]。G 的一个最大匹配是指 G 中的所有匹配中最大的匹配[6]。

1.5 本文组织

本文一共分为 5 章，结构如下：

第一章：绪论，主要介绍课题的研究背景、对研究的问题进行概述，并阐述本文的重点方向

第二章：研究和讨论无向图的双连通性问题

第三章：研究和讨论最小割问题

第四章：研究和讨论二分图匹配的问题

第五章：对全文进行总结，并对未来的工作做一些展望

第二章 图的双连通性与词义消歧

2.1 问题描述

在双连通图中从图中的任意一个节点出发，都可以走到图上的另外的任意一个节点，在判断图的连通性是可以采用 dfs 或者 bfs 的方式来进行求解。而对于双联通的概念则是在图的连通性的基础上，要求这种连通性不需要依赖于任何关键节点。好了，那什么是关键节点呢？对于一张已经连通的图来说（如果都不是连通图了那也没有讨论双连通的必要了），如果我们删除掉一个点以及和这个点相关联的边，剩下的图被分割成多个联通分支[2]（也就是说这些小块只能在自己内部连通而不能和其他小块连通），那么我们就称这个点为关键节点，也称为“割点”。求解双连通分量的关键就是在求解图中的关键节点。这里我们采用到 tarjan 算法[1]对割点进行求解。如图 2-1 所示，是一张包含若干个双连通分量的连通图。

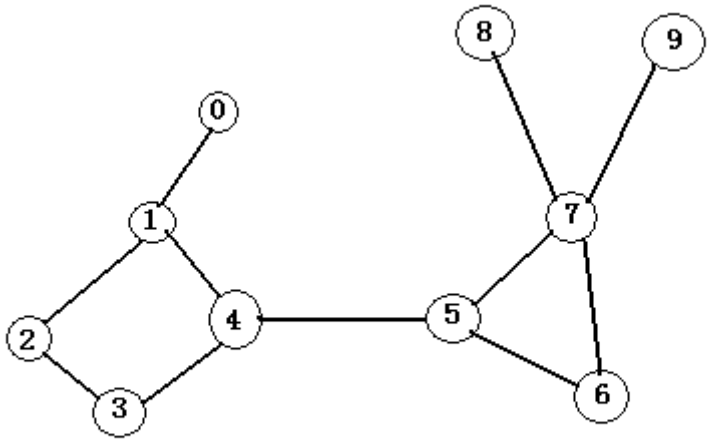


图 2-1 双连通性示例图

首先用 dfs 的方式遍历这张图，并且得到一颗有根树。3 是树根，节点旁边的标号是深度搜索访问节点的顺序，加粗边是图中深度搜索没有访问到的边（因为有的顶点可以多个边到达，深搜只要通过一个边到达顶点，就不再访问该顶点了），称作非树边，也就是树中没有的。细一点的边是树边。

如果两个顶点 u,v ，其中 u 是 v 的祖先或者 v 是 u 的祖先，那么非树边(u,v)叫做回退边。在深搜树中，所有的非树边都是回退边。无向图的深搜树是一棵开放树，如果在其中添加一条回退边，就会形成环，该环路或扩大连通分量的范围，或者导致新的连通分量产生[14]

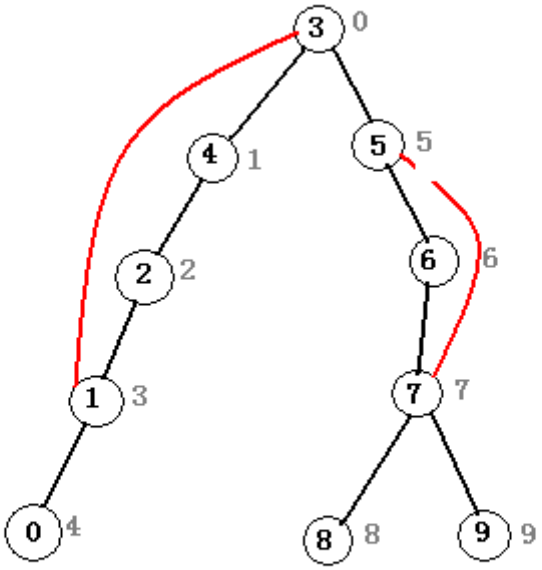


图 2-2 树形展开图

2.2 方法流程

从实例中我们可以发现，当采用 dfs 去将一个图按树形展开之后，最为关键节点的节点有以下的几个特点：

对于一个节点 v 来说，当 v 是树根，如果它有 2 个或者更多儿子，那么它是一个关节点。

当 v 不是树根，当且仅当它有至少一个儿子 w，且从 w 出发，不能通过 w 的后代顶点组成的路径和一条回退边到底 u 的任意一个祖先顶点，此时 v 是一个关节点。 其道理很明显，如果树根包含多个儿子，那么把根节点去掉，整棵树自然被分成多个不相干的部分，图也就断开了。如果 v 是非根顶点，如果其子树中的节点均没有指向 v 祖先的回边，那么去掉 v 以后，将会把 v 及其子树与图的其他部分分割开来，v 自然是关节点。

根据这个性质，我们可以通过在 dfs 的过程中对节点附加一些信息，我们给每个顶点定义一个 low 值，low(u) 表示从 u 出发，经过一条其后代组成的路径和回退边，所能到达的最小深度的顶点的编号。(如果这个编号大于等于 u 的编号，就说明它的后代无法到达比 u 深度更浅的顶点，即无法到达 u 的祖先，那么 u 就是个关节点）low(u) = min{ dfn(u), min{ low(w) | w 是 u 的儿子}, min{dfn(w), | (u,w) 是一条回退边} } dfn(u) 是深搜过程中对顶点的编号值。

因此，我们在深搜过程中计算出 dfn 值和 low 值，如果发现 u 有一个儿子 w ，使得 low(w) >= dfn(u)，那么 u 就是关节点。

求解双连通分量的过程，可以通过深搜完成。 在搜索过程中，如果遇到一个新的边，则压栈，直到找到一个关节点，由于深搜是递归的，在找到一个关节点的同时，必定已经访问完了其子孙节点和其子树的边（包括回退边），而且这些边都在栈中，此时弹出栈中的边直到遇到关节点所在的边即是双连通分支包括的边[15]。流程图如图 2-2 所示。方法流程如下：

1.为节点 u 设定次序编号和 Low 初值

2.将节点 u 压入栈中

3.枚举每一条边，如果节点 v 未被访问则继续向下找 Low[u] = min(Low[u], Low[v])

4.如果节点 v 还在栈内 Low[u] = min(Low[u], DFN[v])

5.如果节点 u 是强连通分量的根则将 v 退栈，为该强连通分量中一个顶点

在双连通性的基础之上我们将结合词义消歧的相关问题对双连通的相关原理和算法进行应用。

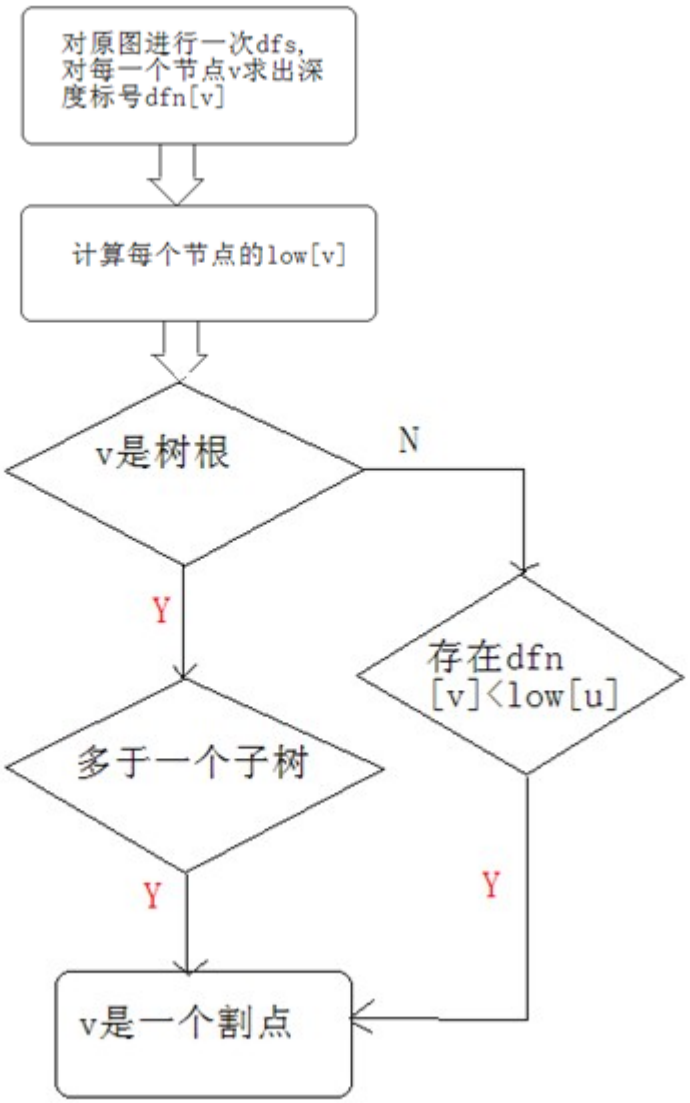


图 2-3　流程图

2.3 词义消歧

词汇的歧义性是自然语言的固有特征.词义消歧根据一个多义词在文本中出现的上下文环境来确定其词义，作为各项自然语言处理的基础步骤和必经阶段被提出来.所谓的词义消歧是指根据一个多义词在文本中出现的上下文环境来确定其词义.形式化地，令词语w具有n个词义，w 在特定的上下文环境C里只有S'是正确的词义，词义消歧的任务就是在这n个词义中确定词义S'.每个词义SK和上下文C都存在或强或弱的联系，记为$R(SK,C)$，其中S'与上下文C的关系应当是最强的.词义消歧技术通过分析和计算W出现的上下文C和每个词义SK之间的关系R，排除干扰词义，最后确定S'。

词义消歧可以分为很多种，每个分类问题都会根据分类依据的不同而得到不

同的分类结果，词义消歧也不例外.根据消歧知识来源的不同，词义消歧方法可分为基于知识的方法和基于统计的方法，基于知识的消歧一般又细分为基于规则的方法和基于词典的方法.基于知识库的消歧方法主要是依赖语言学专家的语言知识构造知识库，通过分析多义词所在的上下文，选择满足一定规则的义项.知识库的类型包括专家规则库、词典、本体、知识库等[18].基于统计的方法则以大型语料库为知识源，从标注或未标注词义的语料中学习各种不同的消歧特征，进而用于词义消歧。

按照消歧过程有无指导，词义消歧分为有导消歧和无导消歧.前者利用已标注了词义的大型语料库来提取特定词义的特征属性，利用机器学习方法生成分类器或分类规则对新实例进行词义判定，后者则从原始的数据文集或机器可读字典中获取词义的相关特征,对新实例进行词义判定。所以，有指导的词义消歧常被看作词义分类问题，无指导词义消歧被看作聚类问题。

按照消歧结果的评价体系，词义消歧分为独立型评估和应用型评估.独立型评估是指不依赖于应用领域，使用一组标准的测试集，独立评价词义消歧性能[19]。应用型评估不单独地评价词义消歧的效果，而是考察其对实际自然语言处理系统最终目标的贡献，比如,词义消歧在机器翻译系统中对翻译性能的影响、在信息检索中对搜索性能的改善情况等等。

2.4 研究问题的应用

2.4.1WordNet

WordNet是从1995年开始，在普林斯顿(Princeton)大学认知科学实验室(Cognitive Science Laboratory)的心理学教授George A. Miller的指导下，由Princeton大学的心理学家、语言学家和计算机工程师联合设计的一种基于认知语言学的在线英语词典。由于它包含了语义信息，所以有别于通常意义下的字典.WordNet根据词条的意义将它们分组,那些具有相同意义的词条称为同义词集(synset),每个synset代表一个潜在的概念(concept)，一个多义词将出现在与其各词义对应的多个同义词集中。它不仅把单词以字母顺序排列,而且按照单词的意义组成一个“单词的网络”。

WordNet的开发有两个目的:其一，它既是一个词典,又是一个辞典，从直觉上讲,它比单纯的词典或辞典更加有用；其二，支持自动的文本分析以及人工智能应用.WordNet是完全免费的资源，其数据库及相应的软件工具的开发都遵从BSD许可协议,可以自由地下载和使用，亦可在线查询和使用.WordNet已经在英语语言处理的研究中得到了广泛应用，几乎成了英语语言知识库的标准。

WordNet中包含了丰富的语义知识,包括词义的定义描述、使用实例、结构化的语义关系、词频信息等，所有这些信息都可以用于词义消岐[20]。

(1) 基于定义描述的词义消岐

WordNet为其中的每个同义词集都提供了简短、概要的定义描述和使用实例，例如，*bus#1: autobus, coach, charabanc, double-decker, jitney, motorbus, motorcoach, omnibus, passenger vehicle--(a vehicle carrying many passengers; used for public transport; "he always rode the bus to work").*

Satanjeev[19]使用WordNet来代替传统的机读词典，对原始Lesk算法进行改进，提出了Adapted Lesk算法。该算法使用WordNet中的多种语义关系来扩展词义的定义描述。在计算两个同义词集的相关度时，不同于原始Lesk算法的简单计数，Satanjeev对多字短语分配了较高的权重(短语字数的平方)，以突出其在相关性判断中的重要性。Satanjeev使用了一种全局消歧策略，即消歧时不是独立确定每个词汇的词义，而是以整个句子作为处理单位，对每种词义的不同组合，计算整体相关性,取相关性最高的组合中的各个词义作为基于概念区域密度的词义消岐的基本思想是：将本体层次结构根据待消歧词的各个词义划分成互相独立的子结构，每个子结构以歧义词的每个词义作为根节点，分别计算各子结构的概念密度来判定目标词的词义。Agirre[22]最先提出使用概念密度进行词义判断，充分利用概念间的概念距离生成定义良好的概念密度公式,基于WordNet中覆盖广泛的名词层次结构对名词进行词义判断。该方法是一个完全自动化的无监督词义消歧方法，给定处于子结构根节点处的词义概念*c*,*nhyp*和*h*分别表示子结构中节点包含的下义概念节点的平均数和子结构的高度，当此子结构中包含*m*个目标词与上下文词汇的词义时*c*的概率密度。

消歧的方法.对一个多义词汇，首先用Lesk方法计算其每个词义的值;然后通过同义词词典或WordNet本体找到每个词义的同义词、上义词、下义词等相关词，再利用Lesk方法计算这些词义的值；最后将这些词义值与其各自的权重相乘，再与源词义值加权求和，得到最终的词义值,词义值最大者为岐义词的确切词义，使用共包含4287词义的872个词语的测试集得到63%的消歧精度,而使用相同的数据集，原始Lesk算法的消歧精度仅为50%。

(2) 基于概念区域密度的词义消岐

基于概念区域密度的词义消岐的基本思想是:将本体层次结构根据待消歧词的各个词义划分成互相独立的子结构,每个子结构以歧义词的每个词义作为根节点，分别计算各子结构的概念密度来判定目标词的词义。Agirre[22]最先提出使用概念密度进行词义判断，充分利用概念间的概念距离生成定义良好的概念密度公式,基于WordNet中覆盖广泛的名词层次结构对名词进行词义判断。该方法是一个完全自动化的无监督词义消歧方法,给定处于子结构根节点处的词义概念*c*,*nhyp*和*h*分别表示子结构中节点包含的下义概念节点的平均数和子结构的高度，当此子结构中包含*m*个目标词与上下文词汇的词义时，*c*的概率密度。

(3) 基于结构化语义关系的图论式词义消岐

WordNet中各同义词集之间通过各种语义关系产生互连。这些语义关系包括：上下位关系(hypernym/ hyponym)、组成成分关系(meronym/part-of)、相似关系(synonym)、反义关系(antonym)等。其中，上下位关系是最常用的语义关系，它将WordNet中的同义词集组织成树状的概念层次体系结构.图论(graph theory)是数学的一个分支，以图为研究对象,用点代表事物，用连接两点的线表示相应的两个事物间具有某种关系，当人们的研究对象在结构上具有内在的、结构化的联系时，常常可以采用图论的方法解决问题。

Mihalcea[25]提出一种在描述语义依赖性的图中使用随机行走策略进行词汇序列消歧的方法。首先将待消歧的词汇序列通过一个语义连接图表示出来，图中的顶点为词汇的语义标签，边为语义节点之间的语义依赖关系;然后在此图中运行一个随机行走算法，使用PageRank算法迭代计算每个节点的重要性值；最后算法将收敛到一组节点标签的静态概率值,这些值用于为词汇序列中的每个词汇确定其最可能的词义，该方法仅使用词典定义，达到了54.2%的消歧精度，远远超过了在这之前的同类方法[26−30]。其主要原因在于，在词汇序列标注的过程中考虑了序列词义之间的整体依赖性。结构化语义互连(structural semantic interconnection,简称SSI)[11]是目前效果最佳的半监督词义消歧技术之一，它属于基于知识的结构化模式识别问题.首先,为消歧词和上下文词汇的每个词义建立一个结构化的图形描述,词义描述的信息来自于多个数据源(WordNet 2.0,Domain labels,WordNet glosses,WordNet usage examples,Dictionaries of collocations),各数据源之间通过人工或自动化的方法进行集成；然后，用一套语法规则来生成各种有意义的关联模式，并为每个关联模式分配权重;最后根据歧义词的每个词义及其上下文信息与这些规则的匹配情况进行消歧.需要指出的是，原SSI方法中的数据源包括了标注文集。

2.4.2 双连通分量的应用

下面通过词义消歧问题对双连通性进行检验，假设现在有 n 种不同的词义在上下文中，这些词义是相互有联系的，现在我们考虑这样一些词义，它们作为一些关键节点连接了上下文，当把这些词义从整个集合中删掉之后会把上下文切断，那么这些词义就很有必要被提取出来特别考虑。我们将单个的词义看成点，若 A 词义和 B 词义之间有关系，就用一条边把他们连接起来，这样我们就得到了一张关于词义的无向图，对这个图做 tarjan，就可以求出关键的词义，然后就可以对这些关键词义进行相应的研究，当然剩下的工作是词义消歧的关键，由于研究的侧重点不同，我们在这里就不做过多的介绍。

2.5 本章小结

在图的双连通性的研究中，我们详细阐述了双连通性的含义以及求解的算法，通过 tarjan 标记的方式将图划分成多个双来南通分量。而这也为词义消歧中词义的建模和划分提供了一个很好的切入点。词汇的歧义性是自然语言的固有特征.词义消歧根据一个多义词在文本中出现的上下文环境来确定其词义，作为各项自然语言处理的基础步骤和必经阶段被提出来.所谓的词义消歧是指根据一个多义词在文本中出现的上下文环境来确定其词义.

我们将单个的词义看成点，若 A 词义和 B 词义之间有关系，就用一条边把他们连接起来，这样我们就得到了一张关于词义的无向图，对这个图做 tarjan，就可以求出关键的词义，然后就可以对这些关键词义进行相应的研究，

第三章 最小割问题

3.1 问题描述

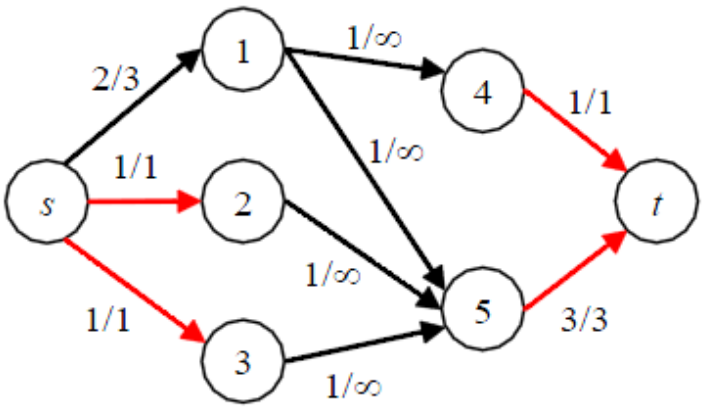
最小割的应用背景是在网络流中的，网络就是一种特殊的有向图，有向加权图 G 中指定两个顶点 S 和 T，分别称为源点和汇点。边上的权值称为边的容量，又设此有向图为严格的有向图，则这个有向图就被称为一个网络[4]。一个网络中有三个要素：点、边和容量，对于这个特殊的有向图，记为 G=（V,E,C）。在图中每条有向边都连接两个特定的点，并且拥有一个权值代表该段路径的最大承载量。当设计一条路线 S,A,C,T 时，该段路线的总承载量受路线中最小容量边制约，而所谓的网络流即为该路线的总承载量，网络中的一个可行流并不是该路线上的所有边容量的简单相加。最小割是最大流的对偶问题。但在实际建模过程中，它不如最大流来的直观，模型也往往隐蔽得很深，不容易找到构图方法。

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 $u,v \in E$，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[5]：$c_f(u,v)=c(u,v)-f(u,v)$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。



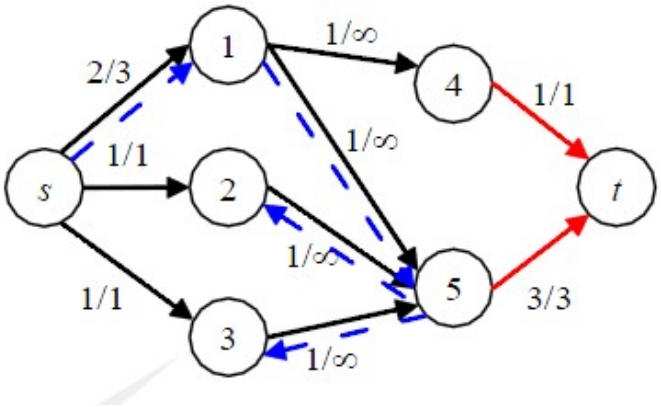


图 3-1 网络图

3.2 最大流算法分析

总体上来说，最大流算法分为两大类：增广路 (Augmenting Path) 和预流推进重标号 (Push Relabel) 。也有算法同时借鉴了两者的长处，如 Improved SAP 。本篇主要介绍增广路类算法，思想、复杂度及实际运行效率比较，并试图从中选择一种兼顾代码复杂度和运行效率的较好方案。以下我们将会看到，有时理论分析的时间复杂度并不能很好的反映一种算法的实际效率。

3.2.1Ford — Fulkerson 方法

所有增广路算法的基础都是 Ford - Fulkerson 方法。称之为方法而不是算法是因为 Ford - Fulkerson 只提供了一类思想，在此之上的具体操作可有不同的实现方案。给定一个有向网络 G(V,E) 以及源点 s 终点 t ，FF 方法描述如下：

(1)初始化网络中的所有边的容量，c<u,v>继承该边的容量，c<v,u>初始化为 0，其中边<v,u>即为回退边。初始化最大流为 0。

(2)在残留网络中找一条从源 S 到汇 T 的增广路 p。如果能找到，则转步骤（3）；如不能找到则转步骤（5）。

(3)在增广路 p 中找到所谓的“瓶颈”边，即路径中容量最小的边，记录下这个值 x,并且累加到最大流中，转步骤（4）

(4)将增广路中所有 c<u,v>减去 x,将所有 c<v,u>加上 x，构成新的残留网络，转步骤(2)

(5)得到网络的最大流，退出

假设有向网络 G 中边 (i,j) 的容量为 c(i,j) ，当前流量为 f(i,j) ，则此边的剩余流量即为 r(i,j) = c(i,j) - f(i,j) ，其反向边的剩余流量为 r(j,i) = f(i,j) 。有向网中所有剩余流量 r(i,j) > 0 的边构成残量网络 Gf ，增广路径 p 即是残量网络中从源点 s 到终点 t 的路径。

沿路径 p 增广流量 f 的操作基本都是相同的，各算法的区别就在于寻找增广路径 p 的方法不同。例如可以寻找从 s 到 t 的最短路径，或者流量最大的路径。

3.2.2Edmonds - Karp 算法

Shortest Augmenting Path (SAP) 是每次寻找最短增广路的一类算法，Edmonds - Karp 算法以及后来著名的 Dinic 算法都属于此。

在无权边的有向图中寻找最短路，最简单的方法就是广度优先搜索 (BFS)，E-K 算法就直接来源于此。每次用一遍 BFS 寻找从源点 s 到终点 t 的最短路作为增广路径，然后增广流量 f 并修改残量网络，直到不存在新的增广路径。

E-K 算法的时间复杂度为 O(VE2)，由于 BFS 要搜索全部小于最短距离的分支路径之后才能找到终点，因此可以想象频繁的 BFS 效率是比较低的。实践中此算法使用的机会较少。

关键代码如下：

```
x <-- 0
while 在残量网络 Gx 中存在增广路 s ~> t
    do 找一条最短的增广路径 P
        delta <-- min{rij:(i,j) 属于 P}
        沿 P 增广 delta 大小的流量
        更新残量网络 Gx
return x
```

流程图如下：

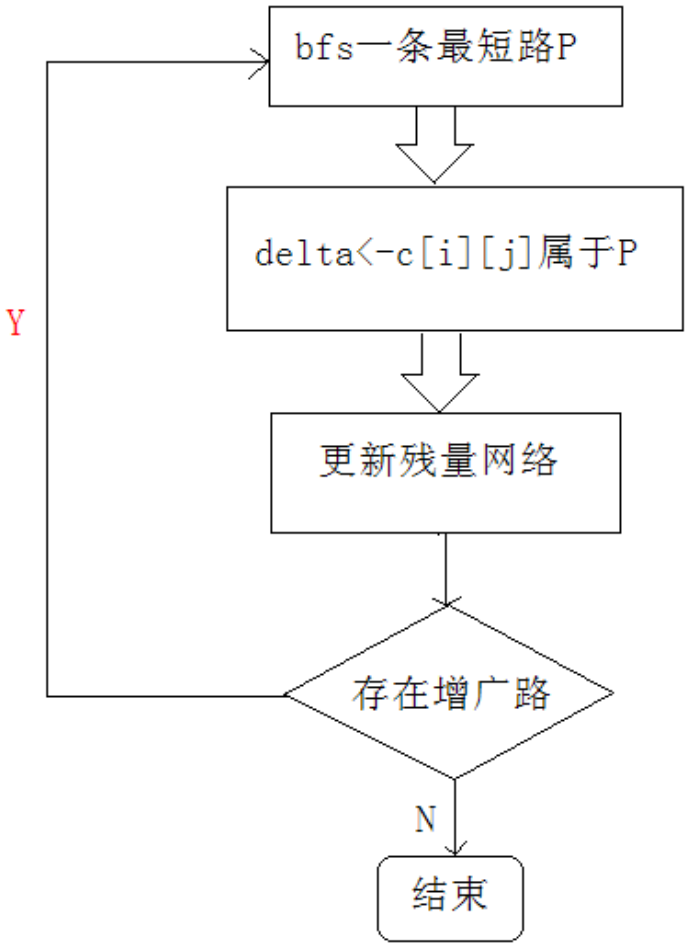


图 3-2 EK 流程图

3.2.3Dinic 算法

BFS 寻找终点相对较慢，而 DFS 又不能保证找到最短路径。1970 年 Dinic 提出一种思想，结合了 BFS 与 DFS 的优势，采用构造分层网络的方法可以较快找到最短增广路，此算法又称为阻塞流算法。

首先定义分层网络 AN(f)。在残量网络中从源点 s 起始进行 BFS，这样每个顶点在 BFS 树中会得到一个距源点 s 的距离 d，如 d(s) = 0，直接从 s 出发可到达的点距离为 1，下一层距离为 2 ... 。称所有具有相同距离的顶点位于同一层，在分层网络中，只保留满足条件 d(i) + 1 = d(j) 的边，这样在分层网络中的任意路径就成为到达此顶点的最短路径。

Dinic 算法每次用一遍 BFS 构建分层网络 AN(f)，然后在 AN(f) 中一遍 DFS 找到所有到终点 t 的路径增广；之后重新构造 AN(f)，若终点 t 不在 AN(f) 中则算法结束。

实际代码中不必真的用一个图来存储分层网络，只需保存每个顶点的距离标号并在 DFS 时判断 dist[i] + 1 = dist[j] 即可。Dinic 的时间复杂度为 O(V2E)。由于较少的代码量和不错的运行效率，Dinic 在实践中比较常用。伪代码如下：

```
p <-- s
 while s 的出度 > 0 do
     u <-- p.top
    if u != t then
        if u 的出度 > 0 then
            设 (u,v) 为 AN(f) 中一条边
            p <-- p, v
        else
            从 p 和 AN(f) 中删除点 u 以及和 u 连接的所有边
    else
        沿 p 增广
        令 p.top 为从 s 沿 p 可到达的最后顶点
     end while
```

流程图如下：

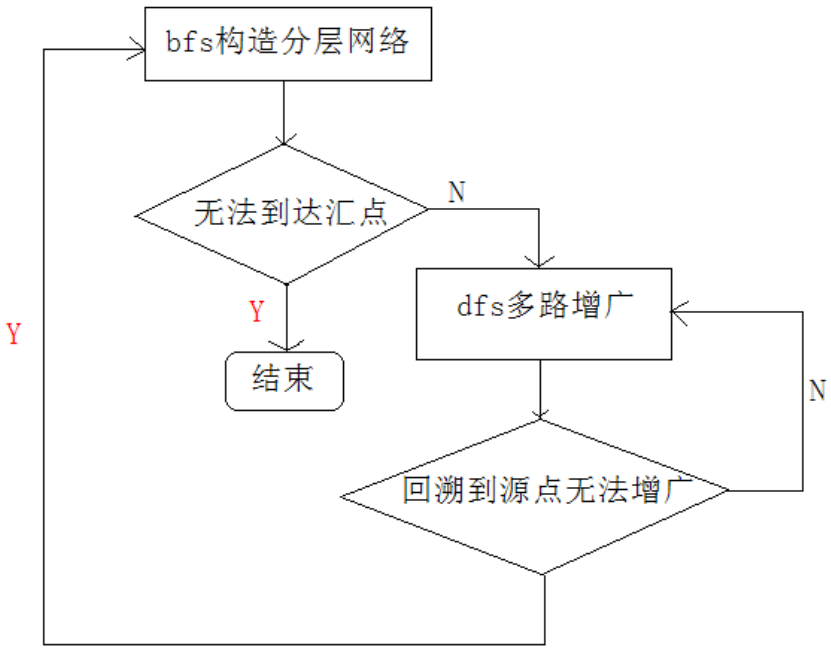


图 3-3　dinic 流程图

3.2.4ISAP 算法

通常的 SAP 类算法在寻找增广路时总要先进行 BFS，BFS 的最坏情况下复杂度为 O(E)，这样使得普通 SAP 类算法最坏情况下时间复杂度达到了 O(VE2)。为了避免这种情况，Ahuja 和 Orlin 在 1987 年提出了 Improved SAP 算法，它充分利用了距离标号的作用，每次发现顶点无出弧时不是像 Dinic 算法那样到最后进行 BFS，而是就地对顶点距离重标号，这样相当于在遍历的同时顺便构建了新的分层网络，每轮寻找之间不必再插入全图的 BFS 操作，极大提高了运行效率。国内一般把这个算法称为 SAP 显然这是不准确的，毕竟从字面意思上来看 E-K 和 Dinic 都属于 SAP，我还是习惯称为 ISAP 或改进的 SAP 算法。

与 Dinic 算法不同，ISAP 中的距离标号是每个顶点到达终点 t 的距离。同样也不需显式构造分层网络，只要保存每个顶点的距离标号即可。程序开始时用一个反向 BFS 初始化所有顶点的距离标号，之后从源点开始，进行如下三种操作：(1)当前顶点 i 为终点时增广 (2) 当前顶点有满足 dist[i] = dist[j] + 1 的出弧时前进 (3) 当前顶点无满足条件的出弧时重标号并回退一步。整个循环当源点 s 的距离标号 dist[s] >= n 时结束。对 i 点的重标号操作可概括为：

dist[i] = 1 + min{dist[j] : (i,j)属于残量网络 Gf}。

算法中的允许弧是指在残量网络中满足 dist[i] = dist[j] + 1 的弧。Retreat 过程中若从 i 出发没有弧属于残量网络 Gf 则把顶点距离重标号为 n 。

虽然 ISAP 算法时间复杂度与 Dinic 相同都是 O(V2E)，但在实际表现中要好得多。要提的一点是关于 ISAP 的一个所谓 GAP 优化。由于从 s 到 t 的一条最短路径的顶点距离标号单调递减，且相邻顶点标号差严格等于 1，因此可以预见如果在当前网络中距离标号为 k (0 <= k < n) 的顶点数为 0，那么可以知道一定不存在一条从 s 到 t 的增广路径，此时可直接跳出主循环。在我的实测中，这个优化是绝对不能少的，一方面可以提高速度，另外可增强 ISAP 算法时间上的稳定性，不然某些情况下 ISAP 会出奇的费时，而且大大慢于 Dinic 算法。

3.2.5 最大容量路径算法 (Maximum Capacity Path Algorithm)

1972 年还是那个 E-K 组合提出的另一种最大流算法。每次寻找增广路径时不找最短路径，而找容量最大的。可以预见，此方法与 SAP 类算法相比可更快逼近最大流，从而降低增广操作的次数。实际算法也很简单，只用把前面 E-K 算法的 BFS 部分替换为一个类 Dijkstra 算法即可。USACO 4.2 节的说明详细介绍了此算法，这里就不详述了。

时间复杂度方面。BFS 是 O(E)，简单 Dijkstra 是 O(V2)，因此效果可想而知。但提到 Dijkstra 就不能不提那个 Heap 优化，虽然 USACO 的算法例子中没有用 Heap ，我自己还是实现了一个加 Heap 的版本，毕竟 STL 的优先队列太好用了不加白不加啊。效果也是非常明显的，但比起 Dinic 或 ISAP 仍然存在海量差距，这里就不再详细介绍了。

3.2.6Capacity Scaling Algorithm

该算法有很多解释，容量缩放算法、容量变尺度算法等。类似于二分查找的思想，寻找增广路时不必非要局限于寻找最大容量，而是找到一个可接受的较大值即可，一方面有效降低寻找增广路时的复杂度，另一方面增广操作次数也不会增加太多。时间复杂度 O(E2logU) 实际效率嘛大约稍好于最前面 BFS 的 E-K 算法，稀疏图时表现较优，但仍然不敌 Dinic 与 ISAP。

通过对最大流的主流算法分析，可以看出不同算法的主要区别在于在做增广路时的处理不同，下面我们将要对这些算法的效率进行比较分析。

3.3 算法效率比较

依次对 EK 算法、Dinic 算法、ISAP 算法和高标预流推进算法进行测试。

（1）蜂窝图

蜂窝图是一种连接情况相对平均的模型，这种情况下 dinic 算法和高标预流推进算法要略优于 ISAP 算法，表 4-1 记录了 4 种算法在同一组大数据下的运行时间和内存使用情况。

表 4-1 蜂窝图效率比较

算法	运行时间	内存使用	代码长度
EK 算法	270ms	32540K	1234Byte
Dinic 算法	5ms	2576K	1747Byte
ISAP 算法	10ms	488K	2122Byte
高表预流推进算法	5ms	720K	2216Byte

（2）完全图

完全图是典型的稠密图，在这种情况下 dinic 算法要明显优于其他算法，dinic 在解决稠密图是效果良好。表 4-2 记录了 4 种算法在同一组大数据下的运行时间和内存使用情况。

表 4-2 完全图效率比较

算法	运行时间	内存使用	代码长度
EK 算法	132ms	32540K	1234Byte
Dinic 算法	15ms	2576K	1747Byte
ISAP 算法	23ms	488K	2122Byte
高表预流推进算法	15ms	720K	2216Byte

（3）团状图

团状图更加近实际模型。这种情况下高标预留推进展现了比较好的时间效率，dinic 算法由于在退流的时候需要不断的做 dfs，所以在团状图的时候效率会受影响。表 4-3 记录了 4 种算法在同一组大数据下的运行时间和内存使用情况。

表 4-3 团状图效率比较

算法	运行时间	内存使用	代码长度
EK 算法	109ms	32540K	1234Byte
Dinic 算法	7ms	2576K	1747Byte
ISAP 算法	23ms	488K	2122Byte
高表预流推进算法	23ms	720K	2216Byte

（4）随机图

从随机图可以看出，dinic 算法和高标预留推进算法应该是效率最优秀的两个算法了。表 4-4 记录了 4 种算法在同一组大数据下的运行时间和内存使用情况。

表 4-4 随机图效率比较

算法	运行时间	内存使用	代码长度
EK 算法	93ms	32540K	1234Byte
Dinic 算法	15ms	2576K	1747Byte
ISAP 算法	23ms	488K	2122Byte
高表预流推进算法	7ms	720K	2216Byte

综合上述 4 种情况绘制出表格如下：

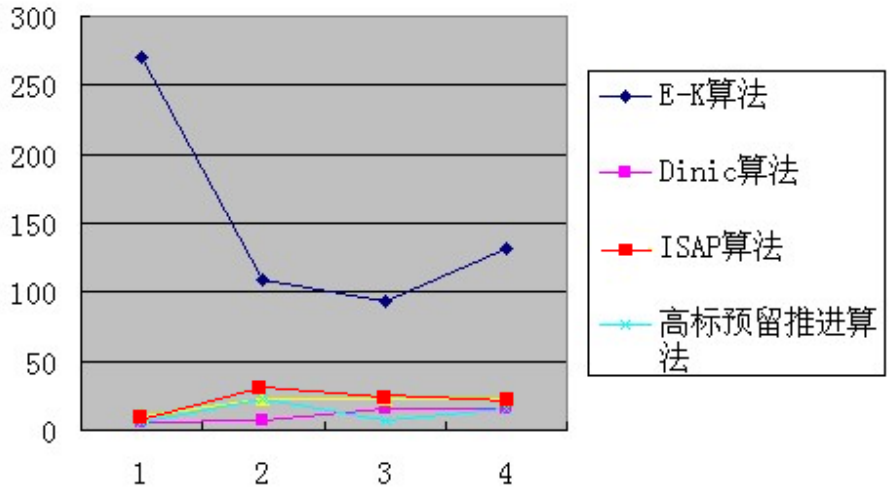


图 3-4 效率统计图

3.4 本章小结

本章首先介绍了网络流的相关概念，详细阐述了最小割的模型以及求解地方式。对于两大类求解最小割的算法：增广路算法和预流推进算法进行了理论上的分析和实际运算效率的比较。其中 EK 算法明显要劣于其他的主流算法，而 dinic 算法在处理稠密图时具有良好的效率，相对的，在处理团状图时高标预留推进算法又显示出了很好的效果。

最大流算法有增广路算法和预流推进算法两个常用的系列。理论时间复杂度一般是 O（N*M^2）或 O（N^2*M），但是最高标号预流推进的理论复杂度逆天到 O（N^2*sqrt（M）），但是我没有实现也没有实测到底有多快。不过网络流的理论复杂度我完全不知道是怎么算的，实际速度也比理论要快上好多的样子

第四章 二分图匹配

4.1 问题描述

二分图是这样一个图，它的顶点可以分类两个集合 X 和 Y，所有的边关联的两个顶点恰好一个属于集合 X，另一个属于集合 Y。给定一个二分图 G，在 G 的一个子图 M 中，M 的边集中的任意两条边都不依附于同一个顶点，则称 M 是一个匹配。进一步，边数最多的匹配称为图的最大匹配。如果所有点都在匹配边上，则称这个最大匹配是完美匹配。

二分图匹配主要围绕这样几个关键点，首先是未盖点。设 I 是 G 的一个顶点，如果 VI 不与任意一条属于匹配 M 的边相关联，就称 VI 是一个未盖点。另外一个概念是交错轨。如果 P 的任意两条相邻的边一定是一条属于 M 而另一条不属于 M，就称 P 是交错轨。最后是可增广轨（增广路）。两个端点都是未盖点的交错轨称为可增广轨。

可增广轨的性质具有这样 3 个性质：

1：P 的路径长度必定为奇数，第一条边和最后一条边都不属于 M。

2：P 经过取反操作可以得到一个更大的匹配 M’。

3：M 为 G 的最大匹配当且仅当不存在相对于 M 的增广路径。

4.2 二分图的最大匹配算法

二分图最大匹配匈牙利算法的思路是不停的找增广轨，并增加匹配的个数，增广轨顾名思义是指一条可以使匹配数变多的路径，在匹配问题中，增广轨的表现形式是一条"交错轨",也就是说这条由图的边组成的路径,它的第一条边是目前还没有参与匹配的，第二条边参与了匹配,第三条边没有最后一条边没有参与匹配,并且始点和终点还没有被选择过。这样交错进行，显然他有奇数条边。那么对于这样一条路径，我们可以将第一条边改为已匹配,第二条边改为未匹配。以此类推，也就是将所有的边进行“取反”，容易发现这样修改以后，匹配仍然是合法的，但是匹配数增加了一对。另外，单独的一条连接两个未匹配点的边显然也是交错轨。可以证明，当不能再找到增广轨时，就得到了一个最大匹配。这也就是匈牙利算法的思路。

二分图最大匹配除了匈牙利算法还有 Hopcroft-Karp 算法，匈牙利算法的复杂的为 O(ne)，而 Hopcroft-Karp 算法的复杂度为 O(en^0.5)。

该算法的精髓在于同时找多条增广路进行反转。先用 BFS 找出可能的增广路，这里用到 BFS 层次搜索的概念，记录当前结点在第几层，用于后面 DFS 沿增广路反转时用，然后再用 DFS 沿每条增广路反转。这样不停地找，直至无法找到增广路为止。

4.3 二分图模型的应用

二分图的模型在很多问题上有高效的应用，主要的方向有 3 点：

(1) 二分图的最大匹配数等于最小覆盖数，即求最少的点使得每条边都至少和其中的一个点相关联，很显然直接取最大匹配的一段节点即可。

(2) 二分图的独立数等于顶点数减去最大匹配数，很显然的把最大匹配两端的点都从顶点集中去掉这个时候剩余的点是独立集，这是|V|-2*|M|，同时必然可以从每条匹配边的两端取一个点加入独立集并且保持其独立集性质。

(3) DAG 的最小路径覆盖，将每个点拆点后作最大匹配，结果为 n-m，求具体路径的时候顺着匹配边走就可以，匹配边 i->j',j->k',k->l'....构成一条有向路径。

二分匹配的模型很适合社交网络的相关处理。在不少社交网站经常会碰到“交友”之类的问题，这类问题大致就是有一些男生和一些女生，他们之前可能相互认识也可能相互不认识，而这个社交网络的作用就是“撮合”他们当中相互不认识的男生和女生进行交往。当然当一对男生和女生开始交往后他们不希望被其他人所打扰，也就是说当一个人已经有了交往的对象之后他会再和其他的任何人进行交往。现在我们希望能够撮合出最多的男生和女生。那么这就是一个二分匹配的问题。

根据社交网站所获取的数据，我们可以进行建模，构建三个集合，其中 X={男生的集合}，Y={女生的集合}，E={关系}，这样就构建了一个二分图模型 G={X,Y,E}

4.4 二分图最优匹配

使二分图的边权和最大的完备匹配称为最优匹配。（如果二分图的两个点集不相等可以通过加 ’点‘ 和 ’权值为 0 的边‘ 实现转化）若 L(x)是顶点 x 对应的顶标值。可行顶标对于图中的每条边(x,y)都有 L(x)+L(y)>=w(x,y)，这个称为可行性顶表。只包含 L(x)+L(y)=w(x,y)的边的子图称为相等子图。然后分析算法流程：设顶点 Xi 的顶标为 a[i]，顶点 Yi 的顶标为 b[i]。初始时，a[i]为与 Xi 相关联的边的最大权值，b[j]=0，保证 a[i]+b[j]>=w(i,j)成立。当相等子图中不包含完备匹配时，就适当修改顶标以扩大相等子图，直到找到完备匹配为止

修改顶标的方法：当从 Xi 开始寻找交错路失败后，得到一棵交错树，对交错树中 X 顶点的顶标减少 d 值， Y 顶点的顶标增加 d 值。

对于图中所有的边 (i,j) 有：

（1）i 和 j 都不在交错树中，边(i,j)仍然不属于相等子图

（2）i 和 j 都在交错树中，边(i,j)仍然属于相等子图

（3）i 不在交错树中，j 在交错树中，a[i]+b[j]扩大，边(i,j)不属于相等子图

（4）i 在交错树,j 不在交错树中,边(i,j)有可能加入到相等子图中

为了使 a[i]+b[j]>=w(i,j)始终成立,且至少有一条边加入到相等子图中,其中 i 在交错树中, j 不在交错树中。由于在算法过程一直保持顶标的可行性，所以任意一个匹配的权值和肯定小于等于所有结点的顶标之和，又因为在扩大相等子图过程中优先加入了权值大大边，所以在相等子图中找到的第一个完备匹配就是最优匹配。

二分图最优匹配的经典算法是 KM 算法，这是一个很神奇的算法，而且较难理解，下面将花一定的篇幅来介绍这个算法：

（1）可行点标：每个点有一个标号，记 lx[i]为 X 方点 i 的标号，ly[j]为 Y 方点 j 的标号。如果对于图中的任意边(i, j, W)都有 lx[i]+ly[j]>=W，则这一组点标是可行的。特别地，对于 lx[i]+ly[j]=W 的边(i, j, W)，称为可行边；

（2）KM 算法的核心思想就是通过修改某些点的标号（但要满足点标始终是可行的），不断增加图中的可行边总数，直到图中存在仅由可行边组成的完全匹配为止，此时这个匹配一定是最佳的（因为由可行点标的的定义，图中的任意一个完全匹配，其边权总和均不大于所有点的标号之和，而仅由可行边组成的完全匹配的边权总和等于所有点的标号之和，故这个匹配是最佳的）。一开始，求出每个点的初始标号：lx[i]=max{e.W|e.x=i}（即每个 X 方点的初始标号为与这个 X 方点相关联的权值最大的边的权值），ly[j]=0（即每个 Y 方点的初始标号为 0）。这个初始点标显然是可行的，并且，与任意一个 X 方点关联的边中至少有一条可行边；

（3）然后，从每个 X 方点开始 DFS 增广。DFS 增广的过程与最大匹配的 Hungary 算法基本相同，只是要注意两点：一是只找可行边，二是要把搜索过程中遍历到的 X 方点全部记下来（可以用 vst 搞一下），以进行后面的修改；

（4）增广的结果有两种：若成功（找到了增广轨），则该点增广完成，进入下一个点的增广。若失败（没有找到增广轨），则需要改变一些点的标号，使得图中可行边的数量增加。方法为：将所有在增广轨中（就是在增广过程中遍历到）的 X 方点的标号全部减去一个常数 d，所有在增广轨中的 Y 方点的标号全部加上一个常数 d，则对于图中的任意一条边(i, j, W)（i 为 X 方点，j 为 Y 方点）：

<1>i 和 j 都在增广轨中：此时边(i, j)的(lx[i]+ly[j])值不变，也就是这条边的可行性不变（原来是可行边则现在仍是，原来不是则现在仍不是）；

<2>i 在增广轨中而 j 不在：此时边(i, j)的(lx[i]+ly[j])的值减少了 d，也就是原来这条边不是可行边（否则 j 就会被遍历到了），而现在可能是；

<3>j 在增广轨中而 i 不在：此时边(i, j)的(lx[i]+ly[j])的值增加了 d，也就是原来这条边不是可行边（若这条边是可行边，则在遍历到 j 时会紧接着执行 DFS(i)，此时 i 就会被遍历到），现在仍不是；

<4>i 和 j 都不在增广轨中：此时边(i, j)的(lx[i]+ly[j])值不变，也就是这条边的可

行性不变。
这样，在进行了这一步修改操作后，图中原来的可行边仍可行，而原来不可行的边现在则可能变为可行边。那么 d 的值应取多少？显然，整个点标不能失去可行性，也就是对于上述的第<2>类边，其 lx[i]+ly[j]>=W 这一性质不能被改变，故取所有第<2>类边的 (lx[i]+ly[j]-W)的最小值作为 d 值即可。这样一方面可以保证点标的可行性，另一方面，经过这一步后，图中至少会增加一条可行边。
<5>修改后，继续对这个 X 方点 DFS 增广，若还失败则继续修改，直到成功为止；
<6>以上就是 KM 算法的基本思路。但是朴素的实现方法，时间复杂度为 O(n4)——需要找 O(n)次增广路，每次增广最多需要修改 O(n)次顶标，每次修改顶标时由于要枚举边来求 d 值，复杂度为 O(n2)。实际上 KM 算法的复杂度是可以做到 O(n3)的。我们给每个 Y 顶点一个“松弛量”函数 slack，每次开始找增广路时初始化为无穷大。在寻找增广路的过程中，检查边(i,j)时，如果它不在相等子图中，则让 slack[j]变成原值与 A[i]+B[j]-w[i,j]的较小值。这样，在修改顶标时，取所有不在交错树中的 Y 顶点的 slack 值中的最小值作为 d 值即可。但还要注意一点：修改顶标后，要把所有不在交错树中的 Y 顶点的 slack 值都减去 d。

4.5 二分匹配与最小割

除此之外，最大流的方式同样也可以求解最优匹配问题。首先我们要知道二分图的最有匹配和二分图的最小点权覆盖是一个对偶的问题。二二分图最小点权覆盖集=二分图最大点权独立集。那么什么是二分图的最大点权独立集呢？

这是一个二分图最大点权独立集问题，就是找出图中一些点，使得这些点之间没有边相连，这些点的权值之和最大。独立集与覆盖集是互补的，求最大点权独立集可以转化为求最小点权覆盖集（最小点权支配集）。最小点权覆盖集问题可以转化为最小割问题解决。结论:最大点权独立集 = 所有点权 - 最小点权覆盖集 = 所有点权 - 最小割集 = 所有点权 - 网络最大流。对于一个网络，除去冗余点（不存在一条 ST 路径经过的点），每个顶点都在一个从 S 到 T 的路径上。割的性质就是不存在从 S 到 T 的路径，简单割可以认为割边关联的非 ST 节点为割点，而在二分图网络流模型中每个点必关联到一个割点（否则一定还有增广路，当前割不成立），所以一个割集对应了一个覆盖集（支配集）。最小点权覆盖集就是最小简单割，求最小简单割的建模方法就是把 XY 集合之间的变容量设为无穷大，此时的最小割就是最小简单割了。

而最小割模型在上一章中已经展开了足够详细的讨论，本章中则不再赘述。

4.6 本章小结

在二分图匹配的研究中，首先对二分图模型的相关性质做了详细的说明，并

且就社交网络中常遇到的问题进行了实际的求解，体现了二分图模型的重要性。在二分图的最优匹配中我们详细分析了 KM 算法，并且通过模型转化的方式，利用网络流在求解二分图的最优匹配，实现了知识的互联，也印证了“一切皆网络流”的传世名言。

当然本文由于时间的限制和本人认知水平的局限性，所做的还远远不够，在词义消歧方面本文仅仅就词义的建模阶段提出了一种划分的方式，而对于整个词义消歧来说还有非常多的工作要做。而词义消歧本身和图论的联系相当紧密，若有时间和机会将是一个很好的研究方向。

在最小割的研究中，还有诸如限定上下界的最大流，最小费用最大流等问题值得探讨，并且最小割模型有很多精妙的应用，也是一个很好的研究方向。

二分图的匹配其实还是有自身的局限性的，而对于一般图的匹配，目前有“带花树”算法被提出，这也是今后可以研究的一个方向。

第五章 总结

回顾本文，一共研究了三个方面的内容：图的双连通性在词义消歧中的应用，最小割算法的研究和比较，二分图的最大匹配和最优匹配以及相关的应用。

在图的双连通性的研究中，我们详细阐述了双连通性的含义以及求解的算法，通过 tarjan 标记的方式将图划分成多个双连通分量。而这也为词义消歧中词义的建模和划分提供了一个很好的切入点。词汇的歧义性是自然语言的固有特征.词义消歧根据一个多义词在文本中出现的上下文环境来确定其词义,作为各项自然语言处理的基础步骤和必经阶段被提出来.所谓的词义消歧是指根据一个多义词在文本中出现的上下文环境来确定其词义.

本文将单个的词义看成点，若 A 词义和 B 词义之间有关系，就用一条边把他们连接起来，这样我们就得到了一张关于词义的无向图，对这个图做 tarjan，就可以求出关键的词义，然后就可以对这些关键词义进行相应的研究，

在最小割这一章中，详细阐述了最小割的模型以及求解地方式。对于两大类算法：增广路算法和预流推进算法进行了理论上的分析和实际运算效率的比较。其中 EK 算法明显要劣于其他的主流算法，而 dinic 算法在处理稠密图时具有良好的效率，相对的，在处理团状图时高标预留推进算法又显示出了很好的效果。

最大流算法有增广路算法和预流推进算法两个常用的系列。理论时间复杂度一般是 O（N*M^2）或 O（N^2*M），但是最高标号预流推进的理论复杂度逆天到 O（N^2*sqrt（M）），但是我没有实现也没有实测到底有多快。不过网络流的理论复杂度我完全不知道是怎么算的，实际速度也比理论要快上好多的样子这就为以后我们在处理不同的图论模型是提供了一种很好的参照。

在二分图匹配的研究中，我们队二分图模型的相关性质做了详细的说明，并且就社交网络中常遇到的问题进行了实际的求解，体现了二分图模型的重要性。在二分图的最优匹配中我们详细分析了 KM 算法，并且通过模型转化的方式，利用网络流在求解二分图的最优匹配，实现了知识的互联，也印证了“一切皆网络流”的传世名言。

当然本文由于时间的限制和本人认知水平的局限性，所做的还远远不够，在词义消歧方面本文仅仅就词义的建模阶段提出了一种划分的方式，而对于整个词义消歧来说还有非常多的工作要做。而词义消歧本身和图论的联系相当紧密，若有时间和机会将是一个很好的研究方向。

在最小割的研究中，还有诸如限定上下界的最大流，最小费用最大流等问题值得探讨，并且最小割模型有很多精妙的应用，也是一个很好的研究方向。

结束语

在这个分析问题日益科学高效，设计方案日益求精求快的时代，对问题的抽象建模变得越来越重要，而模型的建立总离不开事物以及不同事物之间的关系，因此我们可以广泛运用图来进行建模。图论作为离散数学的重要分支之一，在数学领域和计算机科学领域有着不可替代的重要作用。图论以图为研究对象。图论中的图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。

图的连通性和匹配问题是一种应用非常广泛的模型，在本文中，我们对词义消歧和社交网络中的相关问题进行了研究，当然这只是沧海一粟，还有许多问题等待我们去解决。

参考文献

[1] 贾进章,刘剑,宋寿森.基于邻接矩阵图的连通性判定准则[J].辽宁工程技术大学学报,2003,22(2)：158-160.

[2] 何林儒.基于网络流割集理论的路网容量研究[D].长沙：长沙理工大学,2010-4-1

[3] 维斯.数据结构与算法分析:C 语言描述(原书第 2 版)[M].北京：机械工业出版社,2004：112-113.

[4] Sanjoy Dasgupta.算法概论[M].北京：清华大学出版社,2008：97-98.

[5] 乌迪 曼博.算法引论:一种创造性方法[M].北京：电子工业出版社,2010：66-68

[6] 刘会静.关于图的边连通性[D].北京：北京交通大学 2012-12-8

[7] Thomas H.Cormen 等.算法导论(原书第 3 版)[M].北京：机械工业出版社,2012：76-78.

[8] Robert Sedgewick 等.算法(第 4 版)[M].北京：人民邮电出版社,2012：126-127.

[9] 吴艳.基于网络流矩阵求解网络最大流[D].西安：西安交通大学,2007-10-28

[10] 毛华,史田敏,李斌.补图方法在二部图最大匹配中的应用[J].黑龙江大学自然科学学报,2012,29(3)：289-293.

[11] 邓毅雄.图的相对结合数[J].华东交通大学学报,1995,12(1)：92-96.

[13] 屈婉玲,袁崇义.一个图增广问题的 NC 算法[J].北京大学学报(自然科学版),1998,15(3)：223-226

[14] 陈星.几类图的连通性和控制集[D].新疆：新疆大学,2011

[15] 郭知熠.极小 n-棱连通图的最小度点数[J].工程数学学报,2007,3(2)：97-98

[16] 王世英,王瑞霞,王晓丽等.有向图边接通度的下界(英文)[J].山西大学学报(自然科学版),2009,32(004)：516-520.

[17] 刘凤霞.一些重要图类的条件连通度[D].新疆：新疆大学,2006

[18] 杨陟卓,黄河燕.基于异构关系网络图的词义消歧研究[J].计算机研究与发展,2013,50(2)：437-444.

[19] 龚永恩,袁春风,武港山.基于语义的词义消歧算法初探[J].计算机应用研究,2006,23(3)：41-43.

[20] 龚永恩,袁春风,武港山.基于语义的词义消歧算法初探[J].计算机应用研究,2006,23(3)：41-43.