二部图所有极大匹配的求解算法

徐凤生

（德州学院计算机系，山东　德州 253023）

【摘　要】 二部图是一种十分重要的数据结构。在对二部图及匹配的概念进行了阐述后，给出了求二部图所有极大匹配的算法，该算法也可用于求二部图的所有最大匹配和完全匹配。用 C 语言程序验证了此算法的有效性。

【关键词】 二部图；匹配；极大匹配；最大匹配；完全匹配；算法

1　引言

二部图[1,2]是一种十分重要的数据结构，在许多科学领域有着重要的应用。在现实生活中，许多问题可以用二部图来刻画，例如："资料分配"、"工作安排"、"课程表安排"等。二部图也可用于计算机网络，用以实现分布式站点的信息管理。利用二部图解决这些问题则需要涉及匹配算法。目前求最大匹配的常用算法是"匈牙利算法"，其原理是把顶点到顶点的特定通路称为交替链，采用标记顶点和寻找交替链的方法求最大匹配。该算法的缺陷是不能求出所有最大匹配。文献[3]给出了一种求所有最大匹配的算法，但其求解过程比较繁琐，且较难理解。至于求所有极大匹配的算法，目前还未见报道。本文将给出一种求二部图所有极大匹配的算法，该算法也可用于求二部图的所有最大匹配和完全匹配，因而克服了匈牙利算法的不足。文中给出了算法的 C 语言源程序，验证了此算法的有效性。

2　二部图的相关概念

下面的概念和使用的符号可以参见文献[1,2]。

定义 1 简单无向图 $G=<V,E>$，　$V=V_1\cup V_2$ 且 $V_1\cap V_2=\phi$，G 中任一条边的两个端点一个属于 V_1，另一个属于 V_2，则称 G 为二部图，记为 $G=<V_1,V_2,E>$。在二部图 $G=<V_1,V_2,E>$ 中，若 $|V_1|=m$，$|V_2|=n$，且 $\forall u\in V_1, v\in V_2$ 均有 $[u,v]\in E$，称 G 为完全二部图，记为 $K_{m,n}$。

定理 1 简单图 G 为二部图的充要条件是 G 中所有基本回路的长度为偶数。

本文不讨论如何判断一个图是否为二部图的算法实现，只讨论二部图中的匹配算法，所以，下面讨论的图都是指二部图。

定义 2 设二部图 $G=<V_1,V_2,E>$，$M\subseteq E$，若 M 中任意两条边都不相邻，称 M 为二部图 G 的一个匹配。若在 M 中再加入任何一条边就都不是匹配了，称 M 为极大匹配。边数最多的极大匹配称为二部图 G 的最大匹配。

定义 3 设二部图 $G=<V_1,V_2,E>$，M 是 G 中一个最大匹配，若 $|M|=\min\{|V_1|,|V_2|\}$，称 M 为 G 中的一个完备匹配。若 $|V_1|=|V_2|$，M 为 G 的完全匹配。

由上述定义可知，最大匹配和完全匹配一定是极大匹配，但反之不一定成立。因此，只要求出所有极大匹配，则所有最大匹配和完全匹配也就求出来了。

例：根据定理 1 可以判定图 1 中的图 G 为二部图。在 G 中，{u0v0，u1v1，u2v2}，{u0v0，u1v2，u2v1}，{u0v2，u1v0，u2v1}，{u0v2，u1v1}是极大匹配，{u0v0，u1v2，u2v1}，{u0v2，u1v0，u2v1}是最大匹配，但{u0v2，u1v1}不是最大匹配。

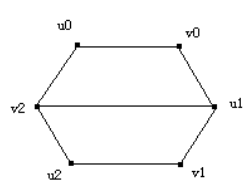

图 1　G

3　求所有极大匹配的算法及其实现

对于二部图 $G=<V_1,V_2,E>$，我们采用矩阵形式来表示。设 $V_1=\{u_0,u_1,\ldots,u_{N-1}\}$，$V2=\{v_0,v_1,\ldots,v_{M-1}\}$，则表示二部图 G 的矩阵为 $A=(a_{ij})$，它是一个 N×M 矩阵，其中

$$a_{ij}=\begin{cases}1 & \text{若 } u_i \text{ 和 } v_j \text{ 之间有边}\\ 0 & \text{否则}\end{cases}$$

定义一个全局变量 B[M]来存储求出的一个极大匹配，其初值元素全为−1。全局变量 P 作为计数器，用以统计求出的极大匹配数。设置局部变量 flag 用以检验矩阵 A 的某一行元素是否全为 0，flag=0 表示该行元素不全为 0，flag=1 表示该行元素全为 0。

根据二部图极大匹配的定义可知，若 M 是二部图 G 的一个匹配，则向 M 中添加边，并保证 M 仍然是一个匹配，直到不能再添加边为止，则 M 一定是二部图 G 的一个极大匹配。本文给出的求二部图所有极大匹配的算法就是基于这种思想。

由于二部图 G 采用矩阵来 A 表示，因此，求 G 的极大匹配的过程实际上是从矩阵 A 的不同行、不同列选取元素为 1 的极大元组。该算法采用搜索、试探、前进、回溯等几种运算。

求二部图所有极大匹配算法的 C 语言源程序如下：

```
#include <stdio.h>
#include <conio.h>
#define N 3//V1 中元素的个数
#define M 3//V2 中元素的个数
int B[N];//存储求出的极大匹配
int P=0; //统计求出的极大匹配数
int A[N][M]={
  {1,0,1},
  {1,1,1},
  {0,1,1},
};//二部图的矩阵表示，可根据实际情况设定
void hs(int i)
{
  int j,k,flag;
  if(i>N-1)//求出一个极大匹配
    {
P++;//计数器加 1
printf("%d:",P);
for(j=0;j<N;j++)
if(B[j]! =-1)printf("(%d,%d)",j,B[j]); //输出该极大匹配
      printf("\n");
    }
  else
  {
    flag=1;
    for(j=0;j<M;j++)//对第 i 行的各个元素进行试探
      if(A[i][j]==1)
 {
    B[i]=j;
    k=0;
          while(B[i]! =B[k])k++;//试探是否与已选元素冲突
   if(k==i) //不冲突，保存该元素并设置 flag 为 0，继续试探下一行；否则恢复 B[i]的初值，试探下一列
   {
flag=0;
hs(i+1);
}
        else
          B[i]=-1;
```

（下转第 47 页）

所有关键词和文档之间的对应关系，从而降低系统的负载，提高检索的效率。下面就给出这种分类的算法。

3　基于文本相似度的关键词分类算法

为了进行关键词分类，系统必须记录用户输入的历史关键词。通过整理超过查询频度 ρ 的关键词，得到高频关键词列表，阈值 ρ 用来控制高频词的数量。假设元搜索引擎已以词为单位缓存了该列表中所有关键词对应的网页快照内容，即建立了词和网页快照的映射关系，考虑到缓存的内容已经是网络全文搜索引擎处理过的结果，我们采用相对简单的相似度算法对这些文档进行分类，进而得到关键词分类，算法步骤如下：

1）采用 DF 算法整理缓存文档的特征信息，提取计算使用的特征词集；

2）将缓存的每篇文档表示为特征词集的文档矢量，为使计算简化，将矢量中各维的值规范化为 1 或 0；

3）给定一个相似度阈值 μ，采用基于海明距离的文本相似度计算公式：

$$sim(M_1,M_2)=1-\frac{\sum_{k=1}^{n} x_k \oplus y_k}{n}$$

其中，⊕表示模 2 加运算，x_k ，y_k 分别表示文档矢量 M_1，M_2 对应的第 k 位特征词的分量，n 为特征词集中所包含词条的数目。计算各篇文档之间的相似度，将高于阈值 μ 的文档归为一类，重复执行步骤 3）至所有缓存文档遍历完，得到对缓存文档的分类结果；

4）在计算文档相似度的同时，记录文档之间共现的关键词，从而在得到文档分类的同时得到与各文档类相对应的共现关键词列表，或称关键词类；

5）对照高频关键词列表，从各个关键词类中剔除未出现在高频关键词列表中的其它词，算法结束。

该算法采用了对数据集进行划分的思想对文档分类，其本质是通过计算缓存文档之间的相似度得到造成这种相似度的共现关键词，从而揭示缓存的文档在关键词上的近似程度，进而为基于关键词的优化查询做准备。

在得到文档的分类和与各个文档类对应的关键词列表后，当用户提交关键词串进行查询时，则将原来的遍历关键词与文档的映射关系改为遍历各个关键词类。若用户所输入的关键词串全部出现在某个关键词类中时，则直接返回该关键词类对应的文档列表；若用户输入的关键词串部分出现在某个关键词类中，则还需检查该类文档和未出现在对应关键词类的关键词间的映射关系，若这种映射关系存在，则该文档也作为结果的一部分，若这种关系不存在，则放弃该文档；若用户输入的关键词串不出现在任何一个关键词类中，则可以认定为低频词，此时缓存策略不起作用，仍然交给网络搜索引擎处理。

若不想过多地强调高频词的作用，则第 5）步是可以省略的，这样能在一定程度上提高用户输入的关键词在缓存中的命中率。

4　分析与结论

该算法对于用户有侧重点的、返回结果信息较少的查询效果较好。因为有侧重点的查询导致输入的某些关键词有高频出现的特征，这种特征为关键词分类奠定了基础。分类实际上揭示了这些关键词在文档中的抱团性质，因此，这种分类操作能有效改善对这些关键词的重复查询效率。且若返回结果的形式采用与搜索引擎类似的分页显示，则获取前几页结果的计算量较小，因为系统可优先提取某关键词类对应的网页快照列表，该关键词类满足完全包含了用户输入的关键词串的条件，因而亦起到了加速查询的作用。

在该算法过程中，阈值 μ 的选取尤为重要，μ 设置太小导致文档分类过于粗糙，所提取的共现关键词很有可能趋近于空集，无法简化查询；μ 设置太大则导致分类太细，每个类中仅包含数量很少的相近似的文档，不能揭示关键词在文档中出现的普遍性，同样无法提高查询的效率。

考虑到元搜索引擎提供的是规模相对较小的查询应用，我们采用较为简单的基于海明距离的相似度算法，且计算过程与用户的查询分开，因此计算的效率和查询影响不大，但该算法的计算复杂度还有待于进一步改善。

参考文献

[1] 张焕炯,李玉鉴,钟义信.文本相似度计算的一种新方法.计算机科学.2002,29(7):92-93
[2] 卜东波,白硕,李国杰.文本聚类重权重计算的对偶性策略.软件学报. 2002,13(11):2083-07
[3] 翟静.可扩展的文本分类系统的核心 API 的设计与实现.四川大学硕士学位论文.2003
[4] 谷波,张永奎.文本聚类算法的分析与比较.电脑开发与应用.2003(11):4-6
[5] 陶跃华,赵波,杨秀国.搜索引擎的文档预处理技术研究.计算机科学. 2002,29(7):111-112

（上接第 45 页）

```
    }
      if(flag)hs(i+1);//若第 i 行元素全为 0,则继续试探下一行
    }
  }
  main()
  {
    int i;
    for(i=0;i<N;i++)B[i]=-1;
    hs(0);
  }
```

该算法采用递归形式来实现，具有结构简练、清晰、可读性强等优点，但递归算法在执行过程中会耗费太多的时间和空间。为了追求算法的时空效率，可将该递归算法转化为非递归算法，限于篇幅，在此不在讨论。

4　结束语

本文对二部图的匹配问题进行了讨论，给出了二部图所有极大匹配的求解算法。由于最大匹配和完全匹配都是极大匹配，所以该算法也可用于求二部图的所有最大匹配和完全匹配。如果只要求给出所有最大匹配或完全匹配，只需对算法进行稍微改动即可。用该算法求最大匹配，可以克服“匈牙利算法”不能求出所有最大匹配的不足。给出的 C 语言源程序已在 Turbo3.0 环境下运行通过，验证了此算法的有效性。

参考文献

[1] 李盘林,李丽双,李洋,王春立等.离散数学[M].北京：高等教育出版社，2001.5
[2] Bernard Kolman,Robert C.Busby,Sharon Cutler Ross 等等.Discrete Mathematical Structures[M].北京：高等教育出版社，2002.4
[3] 田晓明,朱绍文.关于无向二部图最大匹配集矩阵算法的研究[J].湛江师范学院学报，2000.2,69-73
[4] 严蔚敏,吴伟民 编著.数据结构[M].北京：清华大学出版社，2002