网树求解有向无环图中具有长度约束的简单路径和最长路径问题

李 艳[1)] 孙 乐[2)] 朱怀忠[2)] 武优西[2)]

[1)](河北工业大学经济管理学院 天津 300401)

[2)](河北工业大学计算机科学与软件学院 天津 300401)

摘 要 具有长度约束的简单路径(Simple Paths with Length Constraint, SPLC)问题是指求解图中任意两点间路径长度为 m 的简单路径数,是 k-path 问题的一种特殊情况.该文基于网树数据结构提出了在有向无环图中求解SPLC问题的算法(Nettree for SPLC in Directed Acyclic Graphs, NSPLCDAG).网树是一种多树根多双亲的数据结构.NSPLCDAG算法将该问题转化为一棵网树后,利用树根路径数这一性质对其进行求解.对NSPLCDAG算法进行改造,可以求解有向无环图中最长路径问题并形成网树求解最长路径算法(Nettree for the Longest Path in DAGs, NLPDAG),NLPDAG算法可找到所有最长路径,对NLPDAG算法做进一步改进形成改进的NLPDAG算法,改进的NLPDAG算法可在线性时间复杂度内给出有向无环图中的一条最长路径.实验结果验证了NSPLCDAG和改进的NLPDAG算法的正确性与有效性.

关键词 有向无环网络;简单路径;长度约束;最长路径;网树

中图法分类号 TP18 **DOI号**: 10.3724/SP.J.1016.2012.02194

A Nettree for Simple Paths with Length Constraint and the Longest Path in Directed Acyclic Graphs

LI Yan[1)] SUN Le[2)] ZHU Huai-Zhong[2)] WU You-Xi[2)]

[1)](*School of Economics and Management, Hebei University of Technology, Tianjin* 300401)

[2)](*School of Computer Science and Software, Hebei University of Technology, Tianjin* 300401)

Abstract The problem of Simple Paths with Length Constraint (SPLC) is to calculate the number of simple paths between two vertices under the length constraint m in the graph. It is a special case of the k-path problem. In order to solve the problem in Directed Acyclic Graphs (DAGs), this paper proposes an algorithm named Nettree for Simple Paths with Length Constraint in DAGs (NSPLCDAG) based on a data structure of Nettree which may have more than one root and one parent. First NSPLCDAG transforms the graph into a Nettree. Then the concept of the number of root paths of Nettree is used to solve the problem. Based on NSPLCDAG, a new algorithm named Nettree for the Longest Path in DAGs (NLPDAG) is constructed which can find all the longest paths. Then NLPDAG is improved and the improved NLPDAG can find one of the longest paths with linear time complexity. The experimental results verify the correctness and effectiveness of the two algorithms of NSPLCDAG and the improved NLPDAG.

Keywords directed acyclic graphs; simple path; length constraint; the longest path; nettree

收稿日期:2012-06-30;最终修改稿收到日期:2012-08-17.本课题得到国家自然科学基金(61072100,51107029)、河北省自然科学基金(G2010000165)及河北省高等学校青年基金项目(SQ121006)资助.**李 艳**,女,1975年生,博士,副教授,主要研究方向为数据挖掘与智能计算. E-mail: lywuc@163.com. **孙 乐**,男,1986年生,硕士研究生,主要研究方向为数据挖掘与智能计算. **朱怀忠**,男,1978年生,硕士,讲师,主要研究方向为数据挖掘与智能计算. **武优西**(通信作者),男,1974年生,博士,教授,主要研究领域为数据挖掘与智能计算. E-mail: wuc567@163.com.

1 引　言

图是一种比线性表和树更为复杂的数据结构.在线性表中,结点间为单前驱单后继的线性关系;在树结构中,结点间为单前驱多后继的非线性关系;而在图结构中,结点间则为多前驱多后继的非线性关系.图中任意两个结点之间都可能相关,因此图已经被广泛应用于诸如语言学、逻辑学、物理、化学、电气工程和计算机科学等学科中.

在图论中,最长路径(the longest path)问题[1]是指在给定的图中找出一条最长的简单路径.在无向图中求解最长路径问题是著名的 NP 难问题,现有的研究主要基于近似算法[2]、参数化算法[3].另外一类研究是针对特殊图进行求解,并给出多项式时间复杂度的求解算法,如 Mertzios 和 Corneil① 采用一种深度优先搜索策略对伴相似图(cocomparability graphs)中最长路径问题进行了求解;Uehara 和 Valiente[4] 在二部置换图(bipartite permutation graphs)中对最长路径问题建立了线性时间复杂度的求解算法,其求解的二部置换图既是一个置换图也是一个二部图;Ioannidou 等人[5] 在区间图(interval graphs)中运用动态规划思想,对最长路径问题给出了时间复杂度为 $O(n^4)$ 的求解算法;Edmonds 和 Chakraborty[6] 在有向无环图(Directed Acyclic Graphs,DAGs)所有边的长度非负的情况下,对最长路径的方差和期望的边界进行了计算.

k-path 问题是指在给定的图中找出一条长度为 k 的简单路径,是最长路径问题的一种特殊情况.Chen 等人[7] 基于分治思想对一般图中的 k-path 问题进行了研究,提高了该问题的计算速度.k-path 问题在诸多领域具有非常重要的应用.Scott 等人[8] 在生物学约束下,采用彩色编码技术,在酵母蛋白质相互作用网络中查找蛋白质传导路径,其求解方法是将蛋白质作为结点,蛋白质之间的相互作用关系作为边,以此构成的生物蛋白质作用网络,权值最大的 k-path 代表信号传导路径;Desaulniers 等人[9] 利用推广 k-path 的不均衡性(generalized k-path inequalities)对 k 个车辆的路由选择问题进行了研究.与 k-path 问题相近的是求解图中指定两点间路径长度为 k 的最大不相交路径问题,Itai 等人[10] 给出该问题的判定问题是一个 NP-Complete 问题的证明.求解两点间路径长度为 k 的所有简单路径数是具有长度约束的简单路径(Simple Paths with Length Constraint,SPLC)问题.本文在有向无环图中求解该问题形成了 SPLC in DAGs.SPLC in DAGs 可在具有周期间隙约束的序列模式挖掘和具有间隙约束的模式匹配等方面得到应用.Zhang 等人[11] 提出了具有周期间隙约束的序列模式挖掘问题,并将该模式挖掘方法应用在 DNA 序列挖掘中;Tanbeer 等人[12] 将具有周期间隙约束的序列模式挖掘方法应用于购买模式的挖掘.尽管 Zhang 等人采用了空间变换的方法进行序列模式挖掘,但是此方法的基础是具有间隙约束的模式匹配问题[13].文献[14]研究了具有间隙约束和一次性条件的模式匹配问题的求解方法,给出了将具有间隙约束的模式匹配问题转换为有向无环图的算法,这使得具有间隙约束的模式匹配问题与 SPLC in DAGs 问题建立了实质性联系.

为了解决 SPLC in DAGs 问题,本文利用网树数据结构(简称网树,Nettree)[13,15] 进行求解.网树是一种多前驱多后继的非线性数据结构,是对树结构的拓展.网树不仅具有树结构的所有概念,如根结点、叶子结点、双亲、孩子、路径、层等,而且除根结点外,网树中任意结点可以有多个双亲结点.本文提出了网树求解有向无环图中具有长度约束的简单路径算法(Nettree for Simple Paths with Length Constraint in DAGs, NSPLCDAG),NSPLCDAG 算法将该问题转化为一棵网树,然后利用网树的树根路径数这一特殊性质对该问题进行求解.NSPLCDAG 算法的时间复杂度和空间复杂度分别为 $O(m\times n\times t)$ 和 $O(n+|E|)$,这里 m,t,n 和 $|E|$ 分别表示两点间的路径长度约束、顶点最大出度以及顶点数和边数.对 NSPLCDAG 算法做进一步改造,可以形成求解最长路径问题的算法(Nettree for the Longest Path in DAGs, NLPDAG).对 NLPDAG 算法做进一步改进形成改进的 NLPDAG 算法,该算法使得网树退化为一棵树,并使算法的时间复杂度和空间复杂度分别为 $O(|E|)$ 和 $O(n+|E|)$,这里 n 和 $|E|$ 分别表示图的顶点数和边数.

本文第 2 节对 SPLC in DAGs 问题进行定义;第 3 节介绍网树的概念和性质;第 4 节提出 NSPLCDAG 算法,同时分析该算法的时间复杂度和空间复杂度,并通过示例来说明算法的工作原理;第 5 节提出 NLPDAG 算法和改进的 NLPDAG 算法并给出求解示例;第 6 节给出实验结果及分析;第 7 节得

① Mertzios G, Corneil D. A simple polynomial algorithm for the longest path problem on cocomparability graphs. Technical report available at http://arxiv.org/abs/1004.4560

出本文结论.

2 问题的定义

定义 1. 图 $G=(V,E)$,其中 V 称为顶点集,E 称为边集.从顶点 v 到顶点 v' 的路径是一个有序顶点序列 $S=\{v=v_0,v_1,\cdots,v_m=v'\}$,其中顶点序列应满足 $\langle v_{j-1},v_j\rangle\in E(1\leq j\leq m)$.路径长度是路径中有向边的数目.如果序列 S 中任何两个顶点不重复出现,则称此路径为简单路径.

定义 2. 具有长度约束的简单路径 SPLC 问题是指给定图 $G=(V,E)$ 中任意两点 $u,v\in V$ 和正整数 m,求解从 u 到 v 路径长度为 m 的简单路径数.SPLC in DAGs 是指在给定的有向无环图中求解 SPLC 问题.

定义 3. 邻接矩阵是表示顶点之间相邻关系的矩阵,图 G 采用邻接矩阵存储.二维数组元素 $g[i][j]=1(1\leq i,j\leq n,n=|V|)$ 表示顶点 i 到顶点 j 之间存在一条有向边,否则表示顶点 i 到顶点 j 之间无边.

顶点 v_i 的出度(用 $OD(v_i)$ 表示)是第 i 行的元素之和,计算公式如下:

$$OD(v_i)=\sum_{j=0}^{n-1}g[i][j] \tag{1}$$

顶点 v_i 的入度(用 $ID(v_i)$ 表示)是第 i 列的元素之和,计算公式如下:

$$ID(v_i)=\sum_{j=0}^{n-1}g[j][i] \tag{2}$$

例 1. 如图 1 所示的有向无环图,其顶点个数 $n=9$,求从顶点 1 至顶点 7 路径长度约束 $m=4$ 的简单路径数.

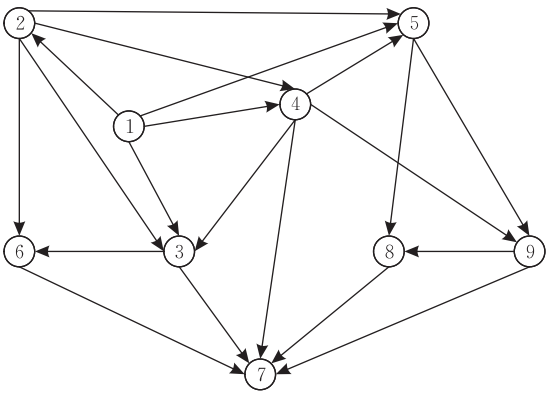

图 1 一个有向无环图

从图 1 可以看出,顶点 1 至顶点 7 路径长度约束为 4 的路径数为 10,即{1,2,4,3,7}、{1,2,3,6,7}、{1,4,3,6,7}、{1,2,5,8,7}、{1,4,5,8,7}、{1,4,9,8,7}、{1,5,9,8,7}、{1,2,4,9,7}、{1,2,5,9,7}和{1,4,5,9,7}.

SPLC in DAGs 问题的求解难度在于,顶点 u 和 v 之间的路径数呈现指数形式,因此不能采用穷举法列出所有可能的路径并判定路径长度是否满足约束条件来进行求解,本文采用网树这一数据结构进行求解.

3 网树的定义及性质

本节给出了网树的定义和性质.

定义 4. 网树数据结构是结点的集合,这个集合可以为空集,或可由若干不同的根结点 $r_1,r_2,\cdots,r_m$ 和 0 或多个非空子网树 $T_1,T_2,\cdots,T_n$ 构成,这些子网树的树根至少存在一条边与网树的根结点 r_i 相连接,这里 $1\leq m,1\leq n$ 且 $1\leq i\leq n$.

图 2 给出了一棵一般意义的网树.

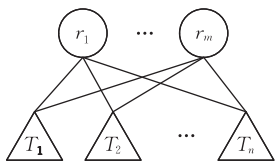

图 2 一棵一般意义的网树

网树具有如下 5 个性质[13,15]:

(1)网树是树结构的拓展,它具有很多与树相似的概念,如根结点、叶子结点、层、双亲、孩子等;

(2)一棵网树可以有 n 个根结点,其中 $n\geq 1$;

(3)除了根结点之外,网树的其它结点可以有多个双亲结点;

(4)从一个结点到达网树的一个根结点的路径不唯一;

(5)同一结点可以在网树的不同层上多次出现.

定义 5. 由于同一结点可以在网树的不同层上多次出现,这里用 n_j^i 来表示第 j 层的结点 i.

定义 6. 从结点 n_j^i 到达根结点的路径数称为树根路径数(the number of root paths),用 $N_r(n_j^i)$ 来表示.

树根路径数具有如下 2 个性质:

(1)根结点 n_1^i 的树根路径数为 1,即 $N_r(n_1^i)=1$;

(2)结点 n_j^i 的树根路径数是其所有双亲结点的树根路径数之和,即

$$N_r(n_j^i)=\sum_{k=1}^{h}N_r(n_{j-1}^{i_k}) \tag{3}$$

这里 $n_{j-1}^{i_k}$ 是结点 n_j^i 的第 k 个双亲，h 是结点 n_j^i 的双亲数.

图 3 给出了一棵网树. 在这棵网树中一些结点在不同层中多次出现. 如结点 3 既出现在第 2 层又出现在第 3 层，本文采用 n_2^3 和 n_3^3 来分别描述第 2 层和第 3 层的结点 3. 结点 n_1^1 和 n_1^2 是网树的两个根结点；n_3^6，n_3^5 和 n_3^3 是网树的 3 个叶子结点. 结点 n_3^6 有两个双亲结点，分别是 n_2^2 和 n_2^3. 结点 n_2^3 的树根路径数 $N_r(n_2^3)=2$ 是因为结点 n_2^3 有两个双亲结点 n_1^1 和 n_1^2 且 $N_r(n_1^1)=N_r(n_1^2)=1$；同理可知，$N_r(n_2^2)=1$；$N_r(n_3^6)=N_r(n_2^2)+N_r(n_2^3)=3$. 网树的另一特征是从一个结点到达网树的一个根结点的路径可能不唯一. 如叶子结点 n_3^6 访问根结点 n_1^1 有两条不同的路径，分别为 $\{n_3^6,n_2^3,n_1^1\}$ 和 $\{n_3^6,n_2^2,n_1^1\}$.

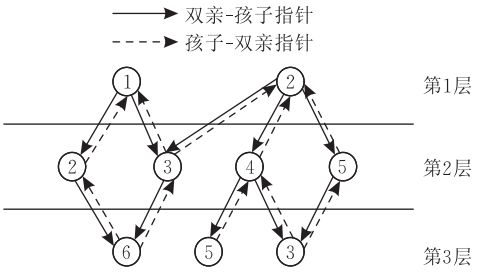

图 3　一棵网树

4　NSPLCDAG 算法描述及复杂度分析

4.1　算法描述

采用 NSPLCDAG 算法求解图 G 中 u，v 两点间长度为 m 的简单路径数问题的基本思想为：将有向无环图 G 转化为一棵 $m+1$ 层网树，然后利用网树的树根路径数这一性质进行求解. 在转化过程中，图 G 的顶点编号 i 即为网树结点名称 i，网树采用从树根层至 $m+1$ 层逐层创建的方式. 将图 G 转化为一棵网树及求解的流程如下：

首先，将顶点 u 作为网树的根结点 n_1^u，该结点为网树的第 1 层且其树根路径数 $N_r(n_1^u)=1$.

其次，依据网树第 $j-1$ 层结点创建网树第 j 层结点. 具体方法是：取网树的第 $j-1$ 层结点 n_{j-1}^i，如果 $g[i][l]=1(1\le l\le n)$ 且在第 j 层未创建结点 n_j^l，则在网树的第 j 层创建结点 n_j^l 并在结点 n_{j-1}^i 和结点 n_j^l 间建立父子关系，并使得 n_j^l 的树根路径数与结点 n_{j-1}^i 的树根路径数一致；若 $g[i][l]=1(1\le l\le n)$ 且在第 j 层已创建结点 n_j^l，则在结点 n_j^l 和结点 n_{j-1}^i 间建立父子关系，并使 n_j^l 的树根路径数累加上结点 n_{j-1}^i 的树根路径数.

最后，该网树第 m 层中与顶点 v 相连接的结点的树根路径数之和即为问题的解. NSPLCDAG 算法的描述如下：

算法 1.　NSPLCDAG 算法.

输入：有向无环图的顶点数 n，有向无环图的邻接矩阵，顶点 u，v 和两点间的路径长度约束 m

输出：路径数 $pathnum$

```
1. 读取存储图的二维数组 g[i][j] (1≤i,j ≤n);
2. 依据顶点 u 初始化网树的根结点;
3. FOR (j=2;j<=m;j++)
4.   FOR (a=0;a<网树第 j-1 层结点数;a++)
5.    i=网树第 j-1 层第 a+1 个结点的名称;
6.    FOR (b=0;b<OD(V_i);b++)
7.     l=顶点 i 的第 b+1 个弧的弧尾顶点;
8.     IF (n_j^l 未被创建)
9.       创建 n_j^l 结点，N_r(n_j^l)=N_r(n_{j-1}^i)，
                          第 j 层结点数自增;
10.     ELSE
11.       N_r(n_j^l)+=N_r( n_{j-1}^i);
12.     END IF
13.    END FOR
14.  END FOR
15. END FOR
16. FOR (d=0;d<网树第 m 层结点数; d++)
17.  IF (g[d][v]==1) pathnum+=N_r(n_m^d);
18. END FOR
19. RETURN pathnum;
```

4.2　NSPLCDAG 算法复杂度分析

NSPLCDAG 算法的空间复杂度是 $O(m\times n\times t+n^2)$ 且可进一步优化为 $O(n+|E|)$，这里 m，t，n 和 $|E|$ 分别表示两点间的路径长度约束、顶点最大出度、顶点数和边数. NSPLCDAG 算法占用的空间是由网树和图 G 的邻接矩阵两部分组成的. 第 1 部分网树的开销为 $O(m\times n\times t)$，这是因为网树的深度是 m，网树中每层最多有 n 个结点，且每个结点最多有 t 个孩子. 对 NSPLCDAG 算法可以做进一步改进，使网树的空间开销为 $O(n)$，其改进方法如下：由于 NSPLCDAG 算法仅依据网树中上一层结点信息来创建下一层结点，因此在存储网树的过程中仅保留一层网树结点即可. 此外，在解决 SPLC in DAGs 问题时，可以不必存储网树的父子关系，而仅计算当前结点的树根路径数，这样网树的开销可以缩减为 $O(n)$. 第 2 部分存储图 G 邻接矩阵的开销为 $O(n^2)$，也可以将图 G 存储为三元组形式，其开销为 $O(|E|)$，因此 NSPLCDAG 算法优化后的空间复杂度为 $O(n+|E|)$.

NSPLCDAG 算法的时间复杂度为 $O(m\times n\times t)$.

这是因为算法的第 3 行循环次数为 $O(m)$，第 4 行的最坏循环次数为 $O(n)$，第 6 行的最坏循环次数为 $O(t)$，算法的第 5，7，9 和 11 行均在 $O(1)$ 时间内即可完成，算法的第 16～18 行的最坏时间复杂度为 $O(n)$. 综上，NSPLCDAG 算法的时间复杂度为 $O(m\times n\times t)$.

4.3 NSPLCDAG 算法运行实例

以例 1 为例来说明 NSPLCDAG 算法的工作原理.

根据 NSPLCDAG 算法的思想，将例 1 中的有向无环图转化为一棵 5 层网树，所建立网树如图 4 所示. 在图 4 中箭头方向代表网树的创建方向，白色圆圈内数字代表网树结点名称，灰色圆圈内数字代表该结点的树根路径数. 网树的创建和问题的求解过程如下：

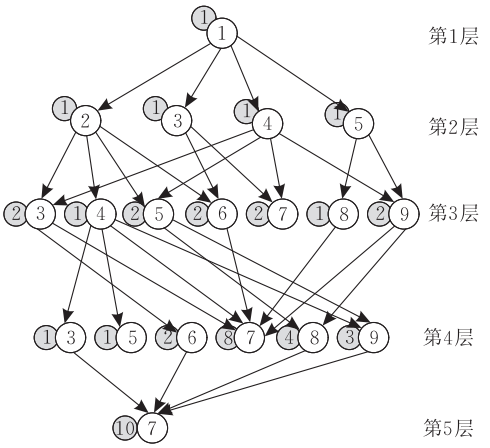

图 4 求解图 1 中从顶点 1 至顶点 7 路径长度约束为 4 的网树

将顶点 1 作为网树的根结点 n_1^1，树根路径数 $N_r(n_1^1)=1$. 将满足 $g[1][l]=1(1\le l\le n)$ 条件的所有顶点创建为网树第 2 层结点 n_2^l，因此网树的第 2 层结点分别是 n_2^2，n_2^3，n_2^4 和 n_2^5，且这些结点的树根路径数均为 1. 之后，对结点 n_2^2 创建孩子结点，创建依据为 $g[2][l]=1(1\le l\le n)$，这样可以创建第 3 层的结点 n_3^3，n_3^4，n_3^5 和 n_3^6，且这些结点的树根路径数均为 1. 依据 $g[3][l]=1(1\le l\le n)$ 对结点 n_2^3 创建孩子结点，易知结点 n_2^3 有两个孩子结点，分别为 n_3^6 和 n_3^7，此时 n_3^6 的树根路径数为 2. 以此类推可以建立网树第 3 层和第 4 层全部结点. 由于路径长度约束为 4，因此根据第 4 层结点与顶点 7 的连接情况，即 $g[l][7]=1(1\le l\le n)$ 来建立父子关系，并求得问题的解为 $1+2+4+3=10$.

5 网树求解最长路径问题

求解最长路径问题是图论中经典问题之一，其是指在给定的图 G 中找到路径长度最长的一条简单路径. 将 NSPLCDAG 算法中根据顶点 u 创建网树的根结点变为依据图 G 中所有入度为 0 的顶点来创建网树的根结点，同时将创建网树深度为指定的长度约束变为网树不再有新的结点产生，即可对最长路径问题进行求解，形成网树求解最长路径问题的算法(Nettree for the Longest Path in DAGs，NLPDAG). NLPDAG 算法从网树最深的一个叶子结点回溯至网树根结点以产生一条最长路径，故 NLPDAG 算法需要对 NSPLCDAG 算法的第 2～3 行和第 16～19 行进行修改，其余各行均保持不变，具体修改如下：

算法 2. NLPDAG 算法.

输入：有向无环图的邻接矩阵

输出：最长路径 *path*

NLPDAG 算法的第 2～3 行：

2. 依据图 G 中所有入度为 0 的顶点来创建网树的根结点；

3. FOR ($j=2$；第 $j-1$ 层结点个数>0；$j++$)

NLPDAG 算法的第 16～19 行：

16. $K=j-2$；

17. *path*[K]=第 $j-1$ 层的第 1 个结点名；

18. 由 *path*[K]回溯至根结点形成最长路径 *path*；

19. RETURN *path*；

由于 NLPDAG 算法必须由最深层叶子结点回溯至网树根结点，因此 NLPDAG 算法不能在求解过程中删除任何网树结点且必须存储网树结点的父子关系，由 NSPLCDAG 算法的空间复杂度和时间复杂度分析可知，NLPDAG 算法的时间复杂度和空间复杂度分别为 $O(K\times n\times t)$ 和 $O(K\times n\times t+|E|)$，这里 K，t，n 和 $|E|$ 分别表示图中最长路径长度、顶点最大出度、顶点数和边数. 由于 NLPDAG 算法是由 NSPLCDAG 算法改进而来，因此 NLPDAG 算法不但可以找到一条最长路径，同时根据所建立的网树可以描述所有最长路径.

以图 1 为例来说明最长路径是如何获得的. 由于图 1 中入度为 0 的顶点只有顶点 1，因此以顶点 1 为网树的根结点来创建网树，网树的前 4 层与图 4 一致，依据第 4 层结点创建第 5 层结点 n_5^6，n_5^7，n_5^8 和 n_5^9，依据第 5 层结点创建第 6 层结点 n_6^7 和 n_6^8，依据第 6 层结点仅能够创建第 7 层结点 n_7^7，由于顶点 7 的出度为 0，因此第 8 层结点个数为 0，循环结束，所创建的网树如图 5 所示，故图 1 的最长路径长度为 6，且最长路径的最终顶点为 7. 结点 n_7^7 的第 1 个双亲结点为 n_6^8；以此类推，回溯至第 1 层，就可以得到

一条最长路径：{1,2,4,5,9,8,7}. 从图 5 可以看出，从顶点 1 至顶点 7 最长的路径只有一条，但在实际问题中，通常最长路径并不唯一.

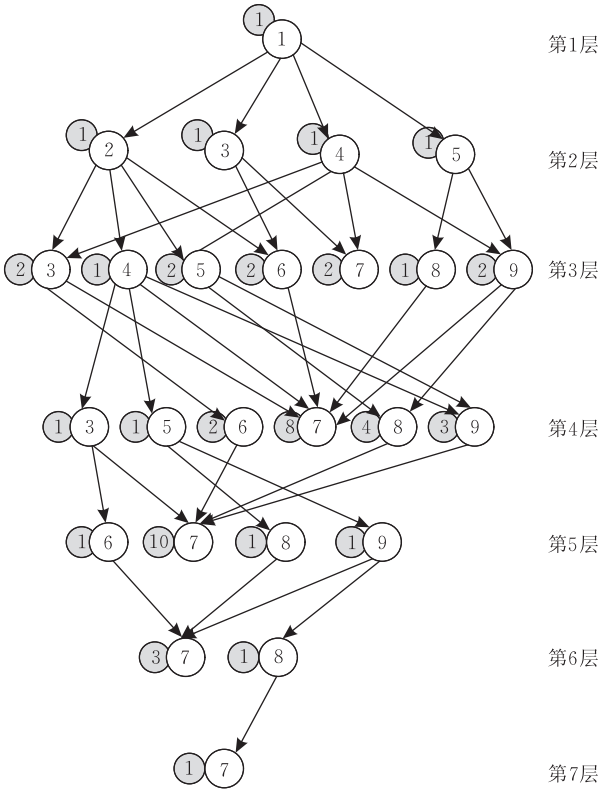

图 5　求解图 1 中最长路径的网树

如果仅需在有向无环图中找到一条最长路径，而无需计算最长路径的路径数，则可以对 NLPDAG 算法进行改进. NLPDAG 算法在有向无环图中求解最长路径时，对很多冗余信息进行了计算，这是因为 NLPDAG 算法将顶点 i 所指向的所有顶点均作为网树中结点 i 的孩子结点. 在求解最长路径时，应仅选择满足删除有向边$\langle i,j\rangle$后，顶点 j 的入度为 0 的点作为网树中顶点 i 的孩子结点，从而形成改进的 NLPDAG 算法. 由于此时不存在一个结点具有多个双亲的情况，因此网树也就退化为一棵树或一个森林（我们可以将树看成是网树的特例）. 具体算法如下：

算法 3.　改进的 NLPDAG 算法.

输入：有向无环图的邻接矩阵

输出：最长路径 *path*

1. 读取存储图的二维数组 $g[i][j]$ $(1\leq i,j\leq n)$；

2. 依据图 G 中入度为 0 的所有顶点来创建网树的根结点；

3. 依次从已创建的网树中取一个未处理的结点 i，所有有向边$\langle i,j\rangle$的顶点 j 的入度自减；

4. IF $(id(v_j)==0)$ 为结点 i 创建孩子结点 j；

5. 重复步 3 和 4 直至不再有新的结点产生；

6. 最长路径长度＝网树的最大深度－1，由该叶子结点回溯至根结点以获得最长路径 *path*；

7. RETURN *path*；

由于改进的 NLPDAG 算法对有向无环图中所有有向边仅处理一次，而每次处理均在 $O(1)$ 内完成，因此改进的 NLPDAG 算法时间复杂度为 $O(|E|)$. 易知，有向无环图若以三元组形式存储，改进的 NLPDAG 算法空间复杂度为 $O(n+|E|)$.

按照改进的 NLPDAG 算法对图 1 进行求解，生成的网树如图 6 所示.

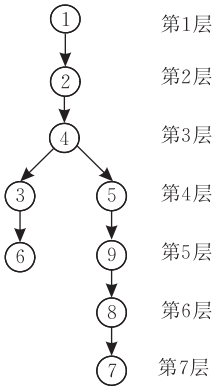

图 6　改进 NLPDAG 算法求解最长路径的网树

6　实验结果及分析

6.1　实验结果

文献[16]采用了链式队列数据结构对有向无环图中从 u 到 v 两点间全部简单路径问题进行了研究，并以产品族零部件关系网络为例，对实际有向无环网络中的应用加以说明，该算法的空间复杂度和时间复杂度均为 $O(n^3)$. 由于该算法是用来求解两点间全部简单路径，因此亦可用于求解本文的 SPLC in DAGs，该算法的时间复杂度与空间复杂度均保持不变. 而本文 NSPLCDAG 算法亦可用于求解文献[16]的问题，求解的方法是以最长路径 K 为 SPLC in DAGs 的长度约束，所创建的网树则可以表示文献[16]中问题的解，显然 NSPLCDAG 算法的时间复杂度和空间复杂度也没有发生任何变化. 图 5 的网树可描述文献[16]中问题在图 1 上的解. 因此本文的 NSPLCDAG 算法在求解相同问题时，算法的时间复杂度和空间复杂度均优于文献[16]的算法. 两种算法的时间复杂度和空间复杂度对比如表 1 所示.

表 1　SPLC in DAGs 的算法复杂度对比

算法	时间复杂度	空间复杂度
文献[16]算法	$O(n^3)$	$O(n^3)$
NSPLCDAG 算法	$O(m\times n\times t)$	$O(n+\|E\|)$

对最长路径问题本文给出了 NLPDAG 算法和改进的 NLPDAG 算法,这两种算法的时间复杂度和空间复杂度对比如表 2 所示.

表 2 最长路径问题算法复杂度对比

算法	时间复杂度	空间复杂度
NLPDAG 算法	$O(K\times n\times t)$	$O(K\times n\times t+\|E\|)$
改进的 NLPDAG 算法	$O(\|E\|)$	$O(n+\|E\|)$

为了获得较大规模的有向无环图,我们采用文献[14]中的算法,将具有间隙约束的模式匹配问题转化为有向无环图.本文采用了文献[15]中使用的前 4 种模式串 $P1$,$P2$,$P3$ 和 $P4$ 来分别生成 DAG1 至 DAG4,并且忽略了文献[15]中的全局约束.序列串采用美国国家生物计算信息中心公布的猪流感 H1N1 病毒的一种候选 DNA 序列(A/Managua/2093.01/2009(H1N1))①中的第一个片段的前 510 个字符.在增加源点和汇点后,使得 DAG1 到 DAG4 的大小均为 512.为了避免序列串对问题求解的影响,DAG5 到 DAG8 则采用文献[15]中的 $P2$ 模式,而序列串则采用前述 DNA 序列的第 1 个~第 4 个序列片段的全部字符,在增加源点和汇点后,使得 DAG5,DAG6,DAG7 和 DAG8 的大小分别为 2288,2301,2171 和 1722.表 3 给出了生成本文 8 个有向无环图的模式串.

表 3 生成有向无环图的模式串

序号	模式串
$P1$	$a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a$
$P2$	$g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a$
$P3$	$g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t$
$P4$	$g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a[1,9]g[1,9]t$

实验运行的软硬件环境为:Intel® Core™ 2 Duo CPU T7100 处理器、主频 1.80 GHz、内存 1GB、Windows XP 操作系统.由于算法的运行速度较快,运行时间较短,为此本文全部实验均采用运行 100 次获得总的运行时间,然后单次运行时间为总时间除以 100 的方法以便较为准确地获得算法在各个实例上的运行时间.为了测试有向无环图的大小对于算法运行时间的影响,对 DAG1,DAG2,DAG3 和 DAG4 分别在 256、320、384 和 448 个结点的子图以及原图中路径长度约束分别为 12,10,12 和 12 的条件下进行了测试,并与文献[16]算法的运行时间进行了对比,结果见表 4.

表 4 不同大小有向无环图的算法运行时间对比

图名称	路径长度约束 m	顶点数	问题解/个	文献[16]算法运行时间/ms	NSPLCDAG 算法运行时间/ms
DAG1	12	256	40	4.84	0.94
	12	320	40	5.32	1.25
	12	384	386	12.50	2.03
	12	448	386	17.19	2.50
	12	512	386	26.09	2.97
DAG2	10	256	12012	132.50	1.41
	10	320	24684	288.75	2.03
	10	384	31278	460.15	2.81
	10	448	42762	740.78	3.43
	10	512	55923	904.53	4.22
DAG3	12	256	31373	355.31	1.56
	12	320	61608	901.25	1.56
	12	384	89704	1308.28	2.81
	12	448	99198	1647.97	4.22
	12	512	135193	2526.10	4.69
DAG4	12	256	35539	357.03	0.87
	12	320	74458	1080.47	1.44
	12	384	109258	1548.12	3.08
	12	448	123782	2101.09	2.91
	12	512	167794	2901.40	2.83

为了对比算法在不同路径长度约束下解的大小和运行时间的长短,对 DAG1 和 DAG2 在长度约束分别为 16,17,18,19,20 和 21 以及 DAG3 和 DAG4 在长度约束分别为 18,24,30,36,42 和 48 的情况下进行了测试,结果见表 5.

表 5 不同长度约束下算法运行时间

图名称	路径长度约束 m	问题解/个	NSPLCDAG 算法运行时间/ms
DAG1	16	884	1.97
	17	0	1.96
	18	1092	2.58
	19	0	2.33
	20	360	2.03
	21	0	2.08
DAG2	16	3281059	4.47
	17	0	4.09
	18	0	4.30
	19	26537036	4.48
	20	0	4.70
	21	0	4.97
DAG3	18	9466972	5.63
	24	790664231	5.62
	30	56485170822	5.94
	36	4321739185142	7.35
	42	272754257287024	7.81
	48	12638966539700128	9.06
DAG4	18	13947750	4.53
	24	1340042909	6.41
	30	119833139429	7.18
	36	10906693648838	8.44
	42	756148905584048	8.44
	48	51060826962548128	8.60

表 6 给出了采用 NLPDAG 算法和改进的 NLPDAG 算法求得的 DAG1 至 DAG8 的最长路径长度、最长路径及求解时间.

① A/Managua/2093.01/2009(H1N1)可在 http://www.ncbi.nlm.nih.gov/genomes/FLU/SwineFlu.html 下载

表 6　最长路径及求解时间

图名称	NLPDAG 算法所求的最长路径长度及最长路径	改进的 NLPDAG 算法所求的最长路径长度及最长路径	NLPDAG 算法运行时间/ms	改进的 NLPDAG 算法运行时间/ms
DAG1	20 {1,320,322,325,328,330,332,333,337,339,342,343,344,345,346,350,353,354,357,361,512}①	20 {1,320,322,325,328,330,332,333,337,339,342,343,344,345,346,350,353,354,357,361,512}	1953	5.47
DAG2	73 {1,131,135,137,142,146,149,154,159,169,179,…,470,475,484,488,493,498,508,512}②	73 {133,138,141,144,151,153,158,162,172,182,186,…,460,470,476,486,496,500,504,510,512}	1515	4.38
DAG3	75 {1,131,135,137,142,146,149,151,153,157,160,164,169,179,189,191,193,198,200,206,214,216,221,231,239,248,250,256,260,267,269,271,273,277,283,288,290,294,296,299,303,307,313,321,327,330,340,342,349,359,366,369,372,382,386,392,397,402,408,412,414,416,418,420,426,435,440,450,460,470,475,484,488,493,498,512}③	75 {126,131,135,140,142,146,149,151,153,158,162,172,182,186,192,197,199,202,209,211,214,220,222,231,241,251,257,261,264,267,269,272,275,277,284,288,292,294,296,302,304,307,317,326,332,339,341,346,354,364,366,371,376,382,386,393,399,402,408,412,414,416,423,433,437,439,444,450,460,470,476,486,496,500,504,512}	1891	5.16
DAG4	78 {1,131,135,137,142,146,149,151,153,157,160,164,169,179,189,190,193,198,200,206,214,216,221,231,239,248,250,256,260,267,269,271,273,274,283,288,289,294,296,299,303,307,313,321,327,330,340,342,349,359,366,369,372,374,377,383,385,386,392,397,402,408,412,414,4 16,418,419,426,435,440,450,460,470,475,484,488,493,498,512 }	78 {126,131,135,140,142,146,149,151,153,158,162,172,182,186,192,197,199,202,209,211,214,220,222,231,241,251,257,261,264,267,269,272,275,277,284,288,292,298,301,302,304,307,317,326,332,339,341,346,354,364,366,371,373,375,381,383,385,386,393,399,402,408,412,414,416,423,433,437,439,444,450,460,470,476,486,496,500,504,512}	1485	6.56
DAG5	—④	331 {133,138,141,144,151,153,158,162,172,182,186,…,1825,1831,1835,1838,1848,1852,2288}	—	74.06
DAG6	—	281 {946,950,959,967,977,981,985,993,1002,1011,1016,…,2282,2284,2288,2296,2300,2301}	—	72.19
DAG7	—	203 {454,459,463,465,475,479,489,498,506,513,518,…,1447,1455,1457,1461,1468,1477,2171}	—	70.47
DAG8	—	343 {1,8,12,19,25,35,41,50,60,65,69,73,75,79,86,…,1695,1700,1704,1709,1713,1721,1722}	—	43.91

①“{}”前面的数字代表最长路径长度；“{}”内的数字串代表求解出的最长路径.
②由于 DAG2 以及 DAG5～DAG8 的最长路径长度较大，限于篇幅，我们仅给出了最长路径的开始和结束的部分值，中间的绝大部分采用“…”进行了省略.
③由于 DAG3 和 DAG4 的最长路径的解仅在 370 附近至 385 附近有所差异，其他部分较为相似，因此本文对这两个实例给出了完整的最长路径.
④由于 DAG5～DAG8 图相对较大，NLPDAG 算法因消耗空间过大，在这些实例上未能给出运行结果.

6.2　实验结果分析

(1) 本文所提出的 NSPLCDAG 算法的效率要大大优于文献[16]所提出的算法.

通过表 4 的全部 20 个实例可以看出，文献[16]算法的运行时间均显著长于本文算法. 如在 DAG3 中，当路径长度约束 m 为 12，顶点数为 512 时，文献[16]算法的运行时间为 2526.1 ms，而 NSPLCDAG 算法的运行时间为 4.69 ms. 这些实验充分地说明了本文所提出的 NSPLCDAG 算法的效率要大大优于文献[16]算法.

(2) 文献[16]算法的求解时间与顶点数的大小和解的大小相关，特别是与解的大小相关，解越大，求解时间增加越显著.

在表 4 的 DAG1 中，当路径长度约束 m 为 12，顶点数分别为 384、448 和 512 时，问题解的大小均为 386，算法运行时间分别为 12.50 ms、17.19 ms 和 26.09 ms，这说明了文献[16]算法的运行时间与顶点数的大小相关. 而在 DAG2，DAG3 和 DAG4 中，当解显著增加时，问题的求解时间也显著增大. 如在 DAG3 中，当路径长度约束 m 为 12，顶点数分别为 384、448 和 512 时，问题解的大小分别为 89704、99198 和 135193，其运行时间分别为 1308.28 ms、1647.97 ms 和 2526.1 ms，这些实例充分地说明了当解显著增加时，文献[16]算法的求解时间也显著增大. 产生上述现象的原因是，文献[16]的算法是从汇点出发，对每条简单路径进行逐一回溯，通过枚举所有可能的解来获得所有的简单路径，因而文献[16]算法的求解时间与解的大小相关.

(3) NSPLCDAG 算法的求解时间与图的大小和长度约束大小相关，更为重要的是当问题的解显著增大时，求解时间并不显著增加.

通过表 4 可以看出，NSPLCDAG 算法的求解时间与图的尺寸未呈现绝对线性变化，但是当图的尺寸增大时，运行时间也相应增加，因此 NSPLCDAG

算法的求解时间与图的大小呈正相关.表 5 也呈现出同样的特点,虽然运行时间与路径长度 m 未呈现绝对线性变化,但是随着路径长度约束 m 的增大,运行时间也相应增加,因此 NSPLCDAG 算法的求解时间与长度约束呈正相关.此外,通过表 4 和 5 还可以看出,当问题的解快速增大时,问题的求解时间并未显著增加,这一特点在表 5 中体现尤为明显.如在 DAG4 中,当路径长度约束由 18 变为 48 时,问题的解增大了近 5×10^9 倍,然而问题的求解时间增加不到一倍,这充分地说明了当问题的解显著增大时,求解时间并不显著增加.当路径长度约束为 18 时,DAG3 解的大小小于 DAG4 解的大小,但是在 DAG3 中的运行时间却略长于 DAG4,这说明问题的求解速度与解的大小无关.NSPLCDAG 算法是一个高效求解算法是因为其采用网树结构,该结构将同一层结点名称相同的结点合并为一个网树结点,有效地避免了组合爆炸现象的发生,大大地提高了问题的求解速度.

(4)表 5 中部分实例的解为 0 的分析与说明.

在表 5 的 DAG1 中,当长度约束为 17,19 和 21 时,问题的解均为 0,造成这一现象的原因有两种:①通过表 6 可以看出,DAG1 的最长路径长度为 20,因此长度约束在大于最大路径长度时,问题的解为 0,所以 DAG1 中长度约束为 21 时,问题的解为 0;②本文的全部有向无环图均采用文献[15]的算法转化而来,对于 DAG1 来说,指向汇点的所有有向边均由模式串 $P1$ 的最后一个字符'a'来生成.模式串 $P1$ 中'a'与't'交替出现,因此当路径长度约束为偶数时,问题的解均不为 0;而为奇数时,如为 17 和 19 时,问题的解均为 0.在 DAG2,DAG3 和 DAG4 中存在同样的现象,由于模式串 $P2$,$P3$ 和 $P4$ 中'a','t'和'g'三者交替出现,由模式串 $P2$ 可知,DAG2 在长度约束为 3 的倍数+1 时,结果均不为 0,而在其它情况下均为 0.由模式串 $P3$ 和 $P4$ 可知,DAG3 和 DAG4 在长度约束为 3 的倍数时,结果均不为 0,而其他情况均为 0.表 5 的 DAG2,DAG3 和 DAG4 在不同长度约束下的求解结果验证了当路径长度约束小于最长路径长度时,两点间的简单路径数也可能为 0.

(5)改进的 NLPDAG 算法适用于求解大规模有向无环图的最长路径问题.

通过表 6 可以看出,在求解 DAG1 至 DAG4 的 4 个大小为 512 的有向无环图的最长路径时,最长路径长度达到 78,由于 NLPDAG 算法占用空间较大,因此在求解这 4 个实例时,运行时间接近 2 s;而改进的 NLPDAG 算法运行时间小于 10 ms.而在 DAG5 至 DAG8 的大小达到 2000 数量级且最长路径长度达到 300 数量级时,NLPDAG 算法因占用空间过大,未能给出运行结果;而改进的 NLPDAG 算法的求解时间小于 100 ms.这充分地说明了改进的 NLPDAG 算法具有较快的求解速度,适用于求解大规模有向无环图的最长路径问题.

(6)有向无环图中最长路径不唯一.

在表 5 的 DAG1 中路径长度约束为 20(即为该图最长路径长度)时,SPLC in DAGs 问题的解为 360,这说明在 DAG1 中,从顶点 1 到顶点 512 共有 360 条不同的最长简单路径.此外,表 6 中 NLPDAG 算法和改进的 NLPDAG 算法在 DAG1 至 DAG4 中给出相同的最长路径长度,相互验证了算法的正确性,并在 DAG2,DAG3 和 DAG4 中给出了不同的最长路径,充分地说明了有向无环图中最长路径不唯一.

(7)对其它现象的分析与说明.

对于表 4 的 DAG1,当其尺寸发生变化时,问题的解可能并不发生变化,即 DAG1 在 384 和 448 个结点的子图以及原图上问题的解均为 386,这说明在 DAG1 中从顶点 1 至顶点 512 在路径长度约束为 12 的情况下,没有经过顶点 384～511 的路径.

由于 NSPLCDAG 算法采用了动态分配内存的方式,因此运行时间会有一定的扰动,故算法在表 4 的 DAG3 中顶点数为 256 和 320 时,取得了相同的运行时间 1.56 ms;另外在表 5 的 DAG4 中,长度约束为 12 和 18 时,运行时间都为 4.53 ms;在表 5 的 DAG1 中,在长度约束为 18 时,算法运行时间比长度约束为 19,20 和 21 的运行时间都长;在表 5 的 DAG2 中,在长度约束为 16 时,算法运行时间比长度约束为 17 和 18 的运行时间都长,这说明算法在某些实例上运行时间会有一定的扰动.

7 结 论

本文研究了求解 SPLC in DAGs 问题的 NSPLCDAG 算法,该算法将此问题转化为一棵网树,并利用网树的树根路径数性质对该问题进行求解.网树的多前驱多后继性有效地避免了组合爆炸现象的发生,大大地提高了问题的求解速度,使得 NSPLCDAG 算法的时间复杂度和空间复杂度分别为 $O(m\times n\times t)$ 和 $O(n+|E|)$,这里 m,t,n 和 $|E|$ 分别表示两点间的路径长度约束、顶点最大出度以及顶点数和边数.本文通过对 NSPLCDAG 算法进行适当修改,形成了有向无环图中求解最长路径问题

的 NLPDAG 算法，在对 NLPDAG 算法进行改进后，形成了改进的 NLPDAG 算法. 改进的 NLPDAG 算法的时间复杂度和空间复杂度分别为$O(|E|)$和$O(n+|E|)$，这里 n 和$|E|$分别表示图的顶点数和边数. 对比性实验验证了 NSPLCDAG 算法和改进的 NLPDAG 算法的正确性和有效性.

致　谢　匿名审稿人对本文提出了宝贵修改意见，在此表示感谢！

参　考　文　献

[1] Wang Jian-Xin, Yang Zhi-Biao, Chen Jian-Er. Algorithms for longest path: A survey. Computer Science, 2009, 36(12): 1-4, 31(in Chinese)
(王建新，杨志彪，陈建二. 最长路径问题研究进展. 计算机科学，2009，36(12)：1-4，31)

[2] Kim S K. Optimal algorithms for finding the longest path with length and sum constraints in a tree. IEICE Transactions on Information and Systems, 2011, E94-D(6): 1325-1328

[3] Williams R. Finding paths of length k in $O^*(2^k)$ time. Information Processing Letters, 2009, 109(6): 315-318

[4] Uehara R, Valiente G. Linear structure of bipartite permutation graphs and the longest path problem. Information Processing Letters, 2007, 103(2): 71-77

[5] Ioannidou K, Mertzios G, Nikolopoulos S. The longest path problem is polynomial on interval graphs. Mathematical Foundations of Computer Science, 2009, (5734): 403-414

[6] Edmonds J, Chakraborty S. Bounding variance and expectation of longest path lengths in DAGs//Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, USA, 2010: 766-781

[7] Chen J, Lu S, Sze S, Zhang F. Improved algorithms for path, matching, and packing problems//Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, USA, 2007: 298-307

[8] Scott J, Ideker T, Karp R M, Sharan R. Efficient algorithms for detecting signaling pathways in protein interaction networks. Journal of Computational Biology, 2006, 13(2): 133-144

[9] Desaulniers G, Lessard F, Hadjar A. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science, 2008, 42(3): 387-404

[10] Itai A, Perl Y, Shiloach Y. The complexity of finding maximum disjoint paths with length constraints. Networks, 1982, 12(3): 277-286

[11] Zhang M, Kao B, Cheung D, Yip K. Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data, 2007, 1(2): 7-39

[12] Tanbeer S K, Ahmed C F, Jeong B, Lee Y. Mining regular patterns in transactional databases. IEICE Transactions on Information and Systems, 2008, E91-D(11): 2568-2577

[13] Wu Y, Wu X, Min F, Li Y. A nettree for pattern matching with flexible wildcard constraints//Proceedings of the 2010 IEEE International Conference on Information Reuse and Integration (IRI2010). Las Vegas, USA, 2010: 109-114

[14] He D, Arslan A N, He Y, Wu X. Iterative refinement of repeat sequence specification using constrained pattern matching//Proceedings of the IEEE 7th International Symposium on Bioinformatics & Bioengineering (BIBE 2007). Cambridge, Massachusetts, USA, 2007: 1199-1203

[15] Wu You-Xi, Wu Xin-Dong, Jiang He, Min Fan. A heuristic algorithm for MPMGOOC. Chinese Journal of Computers, 2011, 34(8): 1452-1462(in Chinese)
(武优西，吴信东，江贺，闵帆. 一种求解 MPMGOOC 问题的启发式算法. 计算机学报，2011，34(8)：1452-1462)

[16] Liu Fu-Yun, Qi Guo-Ning, Che Hong-An. Research on algorithm for detecting simple path in complex network and its application. Systems Engineering- Theory & Practice, 2006, 26(4): 9-13, 84(in Chinese)
(刘夫云，祁国宁，车宏安. 复杂网络中简单路径搜索算法及其应用研究. 系统工程理论与实践，2006，26(4)：9-13，84)

LI Yan, born in 1975, Ph. D., associate professor. Her research interests include data mining and intelligent computation.

SUN Le, born in 1986, M. S. candidate. His research interests include data mining and intelligent computation.

ZHU Huai-Zhong, born in 1978, M. S., assistant professor. His research interests include data mining and intelligent computation.

WU You-Xi, born in 1974, Ph. D., professor. His research interests include data mining and intelligent computation.

Background

The problems of simple paths with length constraint and the longest path are classical problems in graph theory. The problem of simple paths with length constraint in directed acyclic graphs has very important application since some pattern matching problems with gap constraints can transform into this problem. In this paper, we propose two algorithms to deal with the two problems in directed acyclic graphs based on the data structure of Nettree which was put forward for solving pattern matching problems with gap constraints. The experimental results verify the correctness and effectiveness of the two algorithms.

This research is supported by the National Natural Science Foundation of China under Grant No. 61072100, No. 51107029, the Natural Science Foundation of Hebei Province under Grant No. G2010000165, and the Young Scholar Foundation of Colleges and Universities of Hebei Province under grant No. SQ121006.