doi:10.11959/j.issn.1000-436x.2015145

网树求解有向无环图中具有长度约束的最大不相交路径

李艳[1]，武优西[2]，黄春萍[1]，张志颖[1]，曾珍香[1]
（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：对有向无环图中具有长度约束的最大不相交路径问题进行研究，该问题是求解图中两点间路径长度为 k 的最大不相交路径。为了对该问题进行求解，提出了贪婪搜索算法(GP, greedy path)，该算法先将一个有向无环图转化为一棵深度为 k+1 的网树，然后计算每个网树节点的树根叶子路径数，并以此计算图中每个顶点的总路径数，之后从网树的第 k+1 层节点出发，在当前节点的双亲节点中选择未被使用且总路径数最小的双亲，以此形成一条优化的不相交路径，最后迭代这一过程，直到不再有新的不相交路径为止。GP 算法的时间和空间复杂度分别为 $O(wkn(p+q))$和 $O(kn(p+q)+n^2)$。为了测试 GP 算法的近似性，又建立了一种能够生成人工数据的算法，该算法能够准确地控制有向无环图中最大不相交路径的数量。通过该算法生成了大量测试用数据，实验结果表明 GP 算法较其他对比性算法具有良好的近似性且实际求解时间较短，验证了该方法的有效性和可行性。

关键词：有向无环图；长度约束；不相交路径；网树
中图分类号：TP301 **文献标识码**：A

Nettree for maximum disjoint paths with length constraint in DAG

LI Yan[1], WU You-xi[2], HUANG Chun-ping[1], ZHANG Zhi-ying[1], ZENG Zhen-xiang[1]
(1. School of Economics and Management, Hebei University of Technology, Tianjin 300401, China;
2. School of Computer Science and Software, Hebei University of Technology, Tianjin 300401, China)

Abstract: The problem of the maximum disjoint paths in directed acyclic graphs (DAG) was researched which is to find the maximum disjoint paths with length k between two given vertices. A greedy algorithm named greedy path (GP) was proposed to solve the problem. GP transformed a DAG into a nettree with depth k+1 at first. Then the number of root-leaf paths for each node of the nettree was calculated to achieve the number of total paths for each vertex of the DAG. In order to obtain an optimized disjoint path, GP selected the node in the (k+1)th level of the nettree as the current node, and searched for the optimized parent in the usable parents whose number of total paths was minimal. This process was iterated, until there was no disjoint path. The space and time complexities of GP are $O(wkn(p+q))$ and $O(kn(p+q)+n^2)$. To evaluate the performance of GP, an algorithm which can create artificial DAG with known maximum disjoint paths was also proposed. Experimental results show that GP can get better performance than other competitive algorithms.

Key words: directed acyclic graph; length constraint; disjoint path; nettree

1 引言

不相交路径问题在诸多领域中都有极其重要的应用。文献[1,2]研究了无线传感器网络中节点不相交路径路由算法；Hashiguchi 等[3]研究了节点不对称网络中的节点不相交路径问题；包学才等[4]基于不相交路径数对网络的抗毁性进行了评价。对于不相交路径的研究存在多种划分，有针对节点（或顶

收稿日期：2014-07-07；**修回日期**：2014-11-19
基金项目:国家自然科学基金资助项目(61370144)；国家社会科学基金资助项目(12CGL112)；河北省自然科学基金资助项目(F2013202138，G2012202068)；河北省教育厅重点基金资助项目(ZH2012038)；河北省科技支撑计划基金资助项目(14210102D)
Foundation Items: The National Natural Science Foundation of China (61370144); The National Social Science Foundation of China (12CGL112); The Natural Science Foundation of Hebei Province (F2013202138, G2012202068); The Key Project of the Educational Commission of Hebei Province (ZH2012038); The Science-Technology Support Plan of Hebei Province (14210102D)

点）不相交路径的研究[1]，也有针对边不相交路径的研究[5]。在节点不相交路径的研究中，有针对 k 个顶点之间是否存在不相交路径的研究[6]，也有对 2 个顶点间是否存在长度约束的不相交路径的研究[7]。在不相交路径问题的研究中很多是限定了具体图的类型，诸如平面图[8]、有向图[9]、二部图[10]和有向无环图[11]等。此外，也有针对具体实体的 Ad Hoc 网络的研究[12]。

Itai 等[7]理论证明了当路径长度大于等于 4 的情形下，在图中求解具有长度约束的最大不相交路径问题是 NP-HARD 问题。尽管 Yu 等[11]对有向无环图的不相交路径进行了研究，但其算法的时间复杂度和空间复杂度高；之后 Wu[13]继续对该问题进行深入研究，并进一步降低了问题的计算复杂度，该算法的时间和空间复杂度分别为 $O(n^{2k}(\frac{1}{\varepsilon})^{k-1})$和 $O(n^{2k-1}(\frac{1}{\varepsilon})^{k-1})$。这类研究侧重于近似率确定情况下对算法的时间和空间复杂度界的研究。从上述研究结果可以看出，这类算法的时间和空间复杂度都较高，难以应用在实际问题的求解过程中。有向无环图中最大不相交路径数的计算是一个异常复杂的问题，随着有向无环图规模的扩大和结构的复杂化，其计算复杂度呈几何级数增长，因此提出一种有效的求解算法尤为重要。在有向无环图中研究具有长度约束的不相交路径具有重要的实际意义。例如对网络抗毁性度量中，目前通常采用最短路径数、网络结构熵或凝聚度等方法，但是上述方法均有自身的局限性[4]。而基于不相交路径的抗毁性指标更能反映网络抗毁性的物理特征。在某些特定网络中，网络是有级数限定的，如供应链网络。因此在这类网络中研究抗毁性时，需要采用具有长度约束的不相交路径的抗毁性指标进行研究。

本文贡献在于：1）提出了一种贪婪搜索算法（GP, greedy path），该算法将 DAG 图转化为一棵网树，之后通过网树计算 DAG 图中每个顶点的总路径数，依据顶点的总路径数在网树上生成优化的不相交路径；2）为了测试 GP 算法的近似性，建立了一种能够生成具有长度约束的最大不相交路径的有向无环图算法，该算法能够准确地控制图中最大不相交路径的数量；3）大量实验结果说明 GP 算法较其他算法具有更高的近似性，且运行时间大体相当，验证了 GP 算法的可行性与有效性。

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 $<v_{j-1}, v_j>\in E(1\leqslant j\leqslant m)$，路径长度是路径中有向边的数目。如果序列 S 中任何 2 个顶点不重复出现，则称此路径为简单路径。

定义 2　有向无环图 G 中从 s 到 t 两点间路径长度为 k 的所有简单路径数称为两点间具有长度约束的路径数问题，其解用 $N(G,s,t,k)$来表示。

定义 3　若 A 和 B 是有向无环图 G 中从 s 到 t 两点间的 2 条简单路径，如果除去 s 和 t 顶点，A 与 B 路径再无公共顶点，则称路径 A 与 B 是 2 条不相交的路径。

定义 4　设路径 P 是一条从 s 到 t 两点间长度为 k 的简单路径，路径 P 对图 G 的影响是 $N(G,s,t,k)$和 $N(G',s,t,k)$的差值，这里 G'是从图 G 中移除路径 P 中除 s 和 t 以外所有的顶点以及与其相关所有边的剩余子图，即 $G'=G-P$。

为了有效地寻找最大不相交路径，本文采用两点间路径数作为对图影响最小评价函数，这是因为通常路径数越多，存在不相交的路径数则可能越多。因此在生成路径的过程中，如果当前顶点连接多个顶点时，在备选顶点中选择两点间经过该顶点路径数最小的顶点，以此进行贪婪搜索。为了能够快速计算出两点间路径数以及经过每个顶点的路径数，并方便地构造求解算法，本文采用网树结构进行求解。

3　网树定义及性质

文献[14]最早给出了网树的概念，并用于求解具有间隙约束的模式匹配问题。此外，文献[15]采用网树结构对 $N(G,s,t,k)$进行了有效求解。尽管 $N(G,s,t,k)$问题可以采用矩阵乘法进行求解，但是采用矩阵乘法难于有效地表示这些路径。而采用网树结构不但可以有效地求解这个问题，而且可以在线性空间内表示这些路径，因此这里依然采用网树进行求解，下面给出网树的定义及相关概念。

定义 5　网树[14, 15]数据结构是节点的集合，这个集合可以为空集，或可由若干根节点 $r_1, \cdots, r_m$ 以及 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与一个网树树根节点 r_i 相连接，这

里 $1 \leqslant m$，$1 \leqslant n$ 且 $1 \leqslant i \leqslant m$。

从定义 5 可以看出，网树是树结构的拓展，它具有很多与树相似的概念，如根节点、叶子节点、层、双亲、孩子等；网树与树结构的区别在于如下 4 个方面。

1) 一棵网树可以有 n 个根节点，其中 $n \geqslant 1$；

2) 除了根节点之外，网树的其他节点可以有多个双亲节点；

3) 从一个节点到达网树的一个根节点的路径不唯一；

4) 同一节点标签可以在网树的不同层上多次出现，用 n_j^i 来表示第 j 层的节点 i。

定义 6 节点 n_j^i 到达树根节点的路径数目称为节点 n_j^i 的树根路径数 NRP（number of root paths），用 $N_r(n_j^i)$来表示。

树根路径数具有如下 3 个性质[15]。

1) 若节点 n_j^i 为不可用节点，则其树根路径数为 0，即 $N_r(n_j^i)=0$；

2) 若根节点 n_1^i 为可用节点，则其树根路径数为 1，即 $N_r(n_1^i)=1$；

3) 若节点 n_j^i 为可用节点，则树根路径数是其所有双亲节点的树根路径数之和

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

其中，$n_{j-1}^{i_k}$ 是节点 n_j^i 的第 k 个双亲，h 是节点 n_j^i 的双亲数。

定义 7 节点 n_j^i 达到第 m(m 表示网树深度)层叶子节点的路径数目，称为节点 n_j^i 的叶子路径数 NLP (number of leaf paths)，用 $N_l(n_j^i)$来表示。

叶子路径数具有如下 3 个性质。

1) 若节点 n_j^i 为不可用节点，则其叶子路径数为 0，即 $N_l(n_j^i)=0$；

2) 若第 m 层叶子节点 n_m^i 为可用节点，则其叶子路径数为 1，即 $N_l(n_m^i)=1$；

3) 若节点 n_j^i 为可用节点，则叶子路径数是其所有孩子节点的叶子路径数之和，即

$$N_l(n_j^i)=\sum_{k=1}^{h} N_l(n_{j+1}^{i_k}) \tag{2}$$

其中，$n_{j+1}^{i_k}$ 是节点 n_j^i 的第 k 个孩子，h 是节点 n_j^i 的孩子数。

定义 8 所有经过节点 n_j^i 的树根叶子路径的总和称为节点 n_j^i 的树根叶子路径数 NRLP(number of root-leaf paths)，用 $N_p(n_j^i)$来表示。

显然节点 n_j^i 的树根叶子路径数等于其树根路径数和叶子路径数的乘积，计算公式如下

$$N_p(n_j^i)=N_r(n_j^i)N_l(n_j^i) \tag{3}$$

定义 9 两点间经过顶点 i 且长度为 k 的路径总数称为顶点 i 的总路径数，用 $T(i)$来表示。

由于网树节点都是依据DAG图中顶点生成的，因而 DAG 图中顶点 i 在网树上生成节点的标签都为 i。故此顶点 i 的总路径数是由分布在网树中各层节点 n_j^i 的树根叶子路径所组成的，顶点 i 的总路径数计算公式如下

$$T(i)=\sum_{j=1}^{k} N_p(n_j^i) \tag{4}$$

其中，k 是网树最大深度。

定义 10 给定一棵网树，节点 n_j^i 可能具有 t 个双亲，分别为 $n_{j-1}^{a_1}$，$n_{j-1}^{a_2}$,…，$n_{j-1}^{a_t}$，节点 n_j^i 的最左双亲是指双亲中节点标签最小的节点，即 $\min(a_1,a_2,\cdots a_t)$；节点 n_j^i 的最右双亲是指双亲中节点标签最大的节点，即 $\max(a_1,a_2,\cdots,a_t)$；节点 n_j^i 的优化双亲节点是指双亲中总路径数最小的节点。

为了充分地说明网树结构与树结构的区别，下面对一棵网树（如图 1 所示）进行说明。在图 1 中，可以看出很多节点在网树上多次出现，如节点 2 既在第一层出现，也在第二层出现，因此用 n_1^2 和 n_2^2 来分别表示第一层和第二层的节点 2。此外，这棵网树具有 2 个树根节点，分别是 n_1^1 和 n_1^2；很多节点不但具有多个孩子节点，而且还具有多个双亲节点，如节点 n_2^3 具有 2 个双亲节点，分别是 n_1^1 和 n_1^2。从树根节点 n_1^2 到达叶子节点 n_3^5 存在 2 条彼此不同的路径，分别是$<n_1^2,n_2^3,n_3^5>$和$<n_1^2,n_2^4,n_3^5>$，由于节点所在层数可以由其在路径中位置表示出来，因此在表示路径的过程中可以简记为<2,3,5>和<2,4,5>。

图 1 中所有节点均为可用节点的情况下，节点 n_2^3 的树根路径数为 2，因为节点 n_2^3 具有 2 个双亲节点，分别是 n_1^1 和 n_1^2，根据树根路径数的性质 2 可知节点 n_1^1 和 n_1^2 的树根路径数都为 1；进而由式(1)可知节点 n_2^3 的树根路径数为 2。若网树中节点标签为 2 的所有节点均不可用情况下，则此时节点 n_2^3 的树根路径数为 1。由于节点 n_2^3 仅仅有 1 个孩子节点，其

孩子为叶子 n_3^5，根据叶子路径数的性质可知节点 n_3^5 和 n_2^3 的叶子路径数均为 1，这表明从树根节点到达第三层的叶子节点中，在路径的第 2 个位置经过节点 3 有 2 条不同路径，这与实际存在 2 条树根叶子路径<1, 3, 5>和<2,3,5>的情况一致。同理，可以计算节点 n_3^3 的树根叶子路径数为 1。由于在图 1 的网树上，节点标签为 3 的节点只有 2 个，分别是节点 n_2^3 和 n_3^3，根据定义 9 可知，节点 3 在图 1 的网树中的总路径数为 2+1=3。节点 n_3^5 具有 3 个双亲节点，分别是 n_2^2、n_2^3 和 n_2^4，依据最左双亲和最右双亲的定义可知，其最左双亲和最右双亲分别为 n_2^2 和 n_2^4。

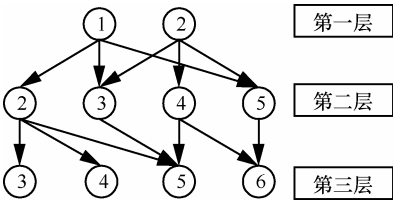

图 1　一棵网树

由于网树结构较为复杂，将图转化为一个网树的过程中，需要频繁地判定网树中节点 n_j^l 是否存在，并在节点 n_{j-1}^i 与节点 n_j^l 之间建立父子关系，而且节点 n_j^l 在第 j 层中并无固定次序出现，因此选择用结构体数组 level 存储各层的网树节点，level 结构定义如下。

```
struct level{
    node *r;        //存储本层节点
};
```

在网树的节点中，增加了判断是否为网树节点的属性 *flag*，当时 *flag* 为真时，表示该节点是一个网树节点；当为假时，表示该节点目前不是一个网树节点。为此网树节点 node 结构的定义如下。

```
struct node{
    int name; //存储节点的标签
    int NRP, NLP, NRLP;    //分别存储该节点的树根路径数、叶子路径数和树根叶子路径数
    int pn,cn; //分别存储该节点的双亲及孩子节点数
    int *ps,*cs;    //分别存储该节点的双亲及孩子节点的位置
    bool flag; //存储该节点是否创建为网树节点
};
```

这样就可以通过 level 结构来描述一棵网树了，其定义如下。

level *nettree;

4　求解算法及其分析

4.1　算法

下面首先介绍将一个有向无环图转化为一棵等价的网树的原理。

4.1.1　有向无环图转化为网树

在读取图的邻接矩阵 $\boldsymbol{G}$ 后，首先，将顶点 s 作为网树的根节点 n_1^s；其次，依据网树第 $j-1$ 层节点创建网树第 j 层节点。具体方法是：取网树的第 $j-1$ 层节点 n_{j-1}^i，如果 $g[i][l]=1(1\leqslant l\leqslant n)$且在第 j 层未创建节点 n_j^l，则在网树的第 j 层创建节点 n_j^l 并在节点 n_{j-1}^i 和节点 n_j^l 间建立父子关系；若 $g[i][l]=1(1\leqslant l\leqslant n)$且在第 j 层已创建节点 n_j^l，则在节点 n_{j-1}^i 和节点 n_j^l 间建立父子关系；最后，创建叶子节点 n_{k+1}^t，并使其与该网树第 k 层中与顶点 t 相连接的节点之间建立父子关系。将图转化为网树的 Transforming 算法的描述如下。

算法 1　Transforming 算法

输入：有向无环图 G，源点 s, 汇点 t 和长度 k

输出：一棵深度为 k+1 的网树 *nettree*

1) 读入图的邻接矩阵 $g[i][j]$ $(1\leqslant i,j\leqslant n)$;
2) 依据顶点 s 初始化网树的根节点；
3) for j=2 to k step 1 do
4)　for a=0 to 网树第 $j-1$ 层节点数−1 step 1 do
5)　　i=网树第 $j-1$ 层第 a+1 个节点的标签；
6)　　for b=0 to $OD(V_i)-1$ step 1 do
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 节点；
10)　　end if
11)　　在节点 $N(n_{j-1}^i)$和 $N(n_j^l)$之间建立父子关系；
12)　end for
13) end for
14)end for
15)　if ($g[d][t]$==1) 在节点 $N(n^d{}_k)$和 $N(n^t{}_{k+1})$之间建立父子关系；
16)end for
17) return *nettree*;

由于节点 n_j^i 到达树根节点的路径数是用 $N_r(n_j^i)$

来表示，而这些路径的长度为 $j-1$，因此 $k+1$ 层叶子节点 n_{k+1}^t 到达树根的路径数就为 $N_r(n_{k+1}^t)$，此时路径的长度为 k，而树根节点为 s，则 $N_r(n_{k+1}^t)$就可以表示 s 和 t 两点间具有长度约束的路径数问题，即 $N(G,s,t,k)=N_r(n_{k+1}^t)$。

为了求解 s 和 t 之间最大不相交路径问题，在每次获得一条长度为 k 的路径时，应选择对有向无环图影响最小的路径 P，为了找到这样一条路径 P，本文提出了优化路径策略生成 PO 路径，为了对比 PO 路径的好坏，本文又给出了最左路径策略产生 PL 路径和最右路径策略产生 PR 路径。

4.1.2 优化路径策略

该策略是指从节点 n_{k+1}^t 向上回溯出一条长度为 k 的路径过程中，需要查找 k 次当前节点 n_j^i 的优化双亲节点。优化双亲节点的选择原则是，在节点 n_j^i 的未被使用的双亲节点中选择一个总路径数最小的节点作为优化双亲节点，由此形成路径 PO。OptimizedMost 算法如下。

算法 2 OptimizedMost 算法

输入：网树 *nettree*, 顶点使用情况 *used*，t 和 k

输出：路径 PO

1) for j=2 to k step 1 do

2) 计算第 j 层每个节点的 $N_r(n_j^i)$值;

3) end for

4) for $j=k$ to 2 step −1 do

5) 计算第 j 层每个节点的 $N_l(n_j^i)$值;

6) 计算第 j 层每个节点的 $N_p(n_j^i)$值;

7) end for

8) for i=1 to n step 1 do

9) 依据式(4)，计算图中每个顶点 $T(i)$值;

10) end for

11) $PO[k]=t$;

12) for $j=k-1$ downto 1 step −1 do

13) $PO[j]=PO[j+1]$的未被使用的最小 $T(i)$值的双亲节点;

14) end for

15) return PO

4.1.3 最左路径策略

最左路径策略是指从节点 n_{k+1}^t 向上回溯一条长度为 k 的路径过程中，每次均选择当前节点的未被使用的最左双亲节点，由此形成路径 PL。LeftMost 算法如下。

算法 3 LeftMost 算法

输入：网树 *nettree*, 顶点使用情况 *used*，t 和 k

输出：路径 PL

1) $PL[k]=k$;

2) for $j=k-1$ downto 1 step−1 do

3) $PL[j]$=依据 *used* 选择 $PL[j+1]$的未被使用的最左双亲节点;

4) end for

5) return PL

4.1.4 最右路径策略

最右路径策略与最左路径策略极为相似，即在回溯产生长度为 k 的路径过程中，每次选择当前节点未被使用的最右双亲节点，并形成路径 PR。RightMost 算法如下。

算法 4 RightMost 算法

输入：网树 *nettree*, 顶点使用情况 *used*，t 和 k

输出：路径 PR

1) $PR[k]=k$;

2) for $j=k-1$ downto 1 step−1 do

3) $PR[j]$=依据 *used* 选择 $PR[j+1]$的未被使用的最右双亲节点;

4) end for

5) return PR

4.1.5 求解算法

现在存在 OptimizedMost、LeftMost 和 RightMost 这 3 种不同策略，可以分别形成 PO、PL 和 PR 共 3 种路径。GP、LP 和 RP 算法是分别对应采用上述 3 种策略而形成的求解算法，另外一个简单而合理的想法是从这 3 种路径中选择对图影响最小路径 P，并形成 SP 算法，该算法如下。

算法 5 SP

输入：有向无环图 G、s、t 和 k

输出：最大不相交路径集合 C

1) 依据 Transforming 算法，将有向无环图 G 转化为一棵网树;

2) 计算 $N(G,s,t,m)$的值;

3) do while $N(G,s,t,m)>0$

4) PL=依据 LeftMost 策略生成的路径;

5) 依据定义 4 计算 PL 对图的影响值 NL;

6) PR=依据 RightMost 策略生成的路径;

7) 依据定义 4 计算 PR 对图的影响值 NR;

8) PO=依据 OptimizedMost 策略生成的路径;

9) 依据定义 4 计算 PO 对图的影响值 NO;

10) $P=PL$、PR 和 PO 中对图的影响最小的路径；

11) 依据 P 更新 *used* 数组；

12) $C=C\cup P$

13) $G=G-P$

14) 更新 $N(G,s,t,m)$的值；

15) loop

16) return C；

通过 SP 算法，很容易获得 GP、LP 和 RP 算法，获得方法如下。

GP 算法

SP 算法中略去第 4)至 7)行及第 10)行，同时将第 11)行改为依据 PO 更新 *used* 数组。

LP 算法

SP 算法中略去第 6)至 10)行，同时将第 11)行改为依据 PL 更新 *used* 数组。

RP 算法

SP 算法中略去第 4)至 5)行及第 8)至 10)行，同时将第 11)行改为依据 PR 更新 *used* 数组。

这样形成了 4 种不同的求解算法，即 SP、GP、LP 和 RP 算法。

4.2　算法分析

定理 1　LP、RP、GP 和 SP 算法的空间复杂度均为 $O(kn(p+q)+n^2)$，这里 k, n, p 和 q 分别是长度约束、图中顶点数、图 G 中顶点的最大入度和最大出度。

证明　图 G 的邻接矩阵的空间复杂度为 $O(n^2)$。网树的深度为 k+1 层，每层最多有 n 个网树节点，每个节点最多有 p 个双亲和 q 个孩子，这里 p 和 q 分别是图 G 中顶点的最大入度和最大出度，因此存储一棵网树需要空间复杂度为 $O(kn(p+q))$。无论是 LeftMost 策略和 RightMost 策略还是 OptimizedMost 策略都需要 $O(k)$来存储生成的路径，而 LP、RP、GP 和 SP 算法都需要使用 *used* 数组来标识图 G 中的某个顶点是否已经使用了，*used* 数组的大小为 $O(n)$。综上，LP、RP、GP 和 SP 算法的空间复杂度均为 $O(kn(p+q)+n^2)$。证毕。

定理 2　LP、RP、GP 和 SP 算法的时间复杂度均为 $O(wkn(p+q))$，这里 w 是问题的解。

证明　首先分析 Transforming 算法时间复杂度，显然算法第 3)行的循环次数为 $O(k)$；第 4)和 15)行网树第 $j-1$ 层和第 k 层节点数最多不超过 n 个，因此循环次数为 $O(n)$；第 6)行的循环次数为 $O(q)$，这里 q 为图的最大顶点出度；而第 5)行和第 7)~11)行以及第 16)行均在 $O(1)$时间复杂度内完成，因此 Transforming 算法的时间复杂度为 $O(knq)$。

然后分析 LeftMost 策略的时间复杂度，LeftMost 策略第 2)行的时间复杂度为 $O(k)$；每个节点最多有 p 个双亲，因此在选择最左双亲过程中，最多有 $O(p)$种选择，这里 p 为图的最大顶点入度；而判断某个节点是否使用仅仅需要在 *used* 数组中查看该双亲节点是否使用，因此检测的时间复杂度为 $O(1)$；这样 LeftMost 策略的时间复杂度为 $O(kp)$。同理，RightMost 策略的时间复杂度也为 $O(kp)$。

而 OptimizedMost 策略中，第 1)行、第 4)行和第 12)行的循环次数均为 $O(k)$；网树的每层最多有 n 个节点，每个节点最多有 p 个双亲和 q 个孩子，因而第 2)行和第 5)行的时间复杂度分别为 $O(nq)$和 $O(np)$；第 6)行的时间复杂度为 $O(n)$；第 13)行的时间复杂度为 $O(p)$，因此算法中除了第 8)~10)行之外的时间复杂度为 $O(k(p+q))$，而第 9)行时间复杂度为 $O(k)$，因为一个顶点最多在 k+1 层的网树上出现 $O(k)$次，而第 8)行的循环次数为 n 次，因此 OptimizedMost 策略总体时间复杂度为 $O(kn(p+q))$。

计算或更新 $N(G,s,t,m)$的值的时间复杂度为 $O(knq)$，这是因为网树有 $O(k)$层网树节点，每层最多有 $O(n)$个节点，每个节点的孩子是 $O(q)$。所以 SP 算法的第 1)、2)、5)、7)、9)和 14)行的时间复杂度都为 $O(knq)$。而第 10)行的时间复杂度为 $O(1)$；第 12)和 13)行的时间复杂度都为 $O(k)$。SP 算法的第 3)行最大循环次数为 w 次，这里 w 为图中最大不相交路径的实际值。这样 SP 算法的时间复杂度为 $O(wkn(p+q))$。进而 LP、RP 和 GP 算法的时间复杂度也都为 $O(wkn(p+q))$。证毕。

尽管这 4 个算法的时间复杂度表示形式上都相同，但是由于 SP 中蕴含了 LP、RP 和 GP 算法，因此 SP 算法将是最慢的算法，而且 GP 算法也因为比 LP 和 RP 略微复杂，则 GP 算法将是次慢算法。而 LP 和 RP 算法的时间复杂度则完全相同。

4.3　运行实例

给定如图 2 所示的有向无环图，求从顶点 1 到顶点 8 路径长度为 4 的最大不相交路径。

GP 算法的工作过程如下：首先将该问题用 Transforming 算法转化为如图 3 所示的深度为 5 层的一棵网树。由于图 3 中为灰色的网树节点均不能到达第 5 层节点 8（即 n_5^8），因此这些节点对计算结果不

产生任何影响，属于无效节点。为了避免干扰，图 4 为简化后的深度为 5 层的一棵网树。下面介绍如何按照优化路径策略获得一条从 1 到 8 长度为 4 的路径。

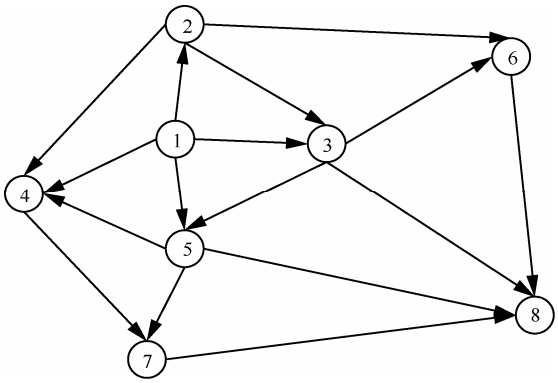

图 2　一个有向无环图

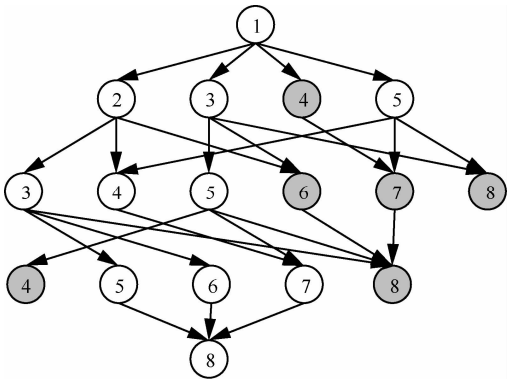

图 3　深度为 5 层的一棵网树

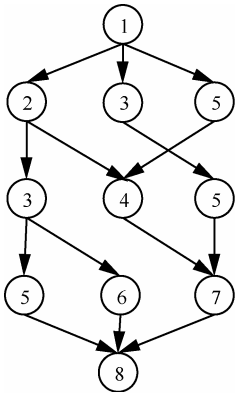

图 4　简化后的网树

对该化简后的网树，按照式(1)和式(2)计算第 2 层至第 4 层各个节点的树根路径数和叶子路径数，其结果标识在图 5 中每个节点的上方，“×”号前面的数值表示该节点的树根路径数；“×”号后面的数值表示该节点的叶子路径数。按照式(3)不难计算出，节点 n_2^2 的树根叶子路径数为 1×2=2，这就是说，从顶点 1 到达顶点 8 在路径长度为 4 的情况下，在路径中第 2 个位置经过顶点 2 的路径共有 2 条，这与实际有 2 条路径（即<1,2,3,5,8>和<1,2,3, 6,8>）在第 2 位置为顶点 2 相一致。同理，可以计算出节点 n_2^3 和 n_2^5 的树根叶子路径数均为 1；节点 n_3^3 、n_3^4 和 n_3^5 的树根叶子路径数分别为 2、1 和 1；节点 n_4^5 、n_4^6 和 n_4^7 的树根叶子路径数分别为 1、1 和 2。按照式(4)不难计算出各个顶点的树根叶子路径数，由于顶点 3 在图 5 的网树中 2 次重复出现，因此顶点 3 的总路径数 $T(3)=N_p(n_2^3)+N_p(n_3^3)=1+2=3$；而顶点 5 在网树中 3 次重复出现，所以顶点 5 的总路径数 $T(5)=N_p(n_2^5)+N_p(n_3^5)+N_p(n_4^5)=1+1+1=3$。而顶点 2、4、6 和 7 的树根叶子路径数分别为 2、2、1 和 7。按照优化路径策略可知，节点 n_5^8 在选择双亲的过程中要选择节点树根叶子路径数最小的双亲节点，因此选择节点 n_4^6。而节点 n_4^6 仅有 1 个双亲节点 n_3^3，因此选择节点 n_3^3；通过节点 n_3^3 又会选择其唯一的双亲节点 n_2^2，这样就按照 OptimizedMost 策略形成了一条路径<1,2,3,6,8>。

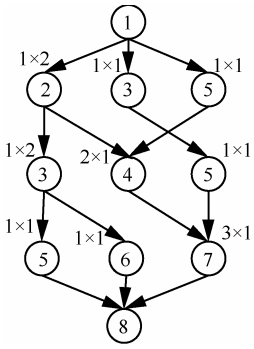

图 5　标出节点的树根路径数和叶子路径数的网树

在顶点 2、3 和 6 为不可以用的情况下，新的网树如图 6 所示。在顶点 2、3 和 6 不可用的情况下，按照式(1)和式(2)计算第 2 层至第 4 层各个节点的树根路径数和叶子路径数，其结果如图 7 所示。针对图 7，优化路径策略能获得<1,5,4,7,8>路径。所以采用 GP 算法能够获得 2 条路径，分别为<1,2,3, 6,8>和<1,5,4,7,8>。

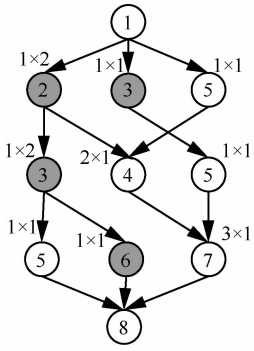

图 6　部分节点不可用时的网树

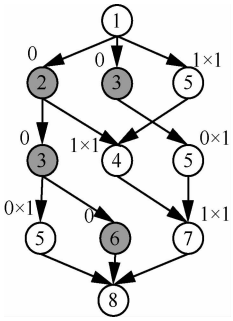

图 7　更新后的网树

依据图 4，节点 n_5^8 具有 3 个双亲节点，分别为 n_4^5、n_4^6 和 n_4^7，由于采用最左路径策略在选择节点 n_5^8 双亲节点过程中，选择了节点 n_4^5，而节点 n_4^5 仅有 1 个双亲节点，因此最左路径策略再次选择了节点 n_3^3，之后选择了节点 n_2^2，这样采用 LeftMost 策略可以获得路径<1,2,3,5,8>，因此 LP 算法的结果就仅为<1,2,3,5,8>。同理，按照最右路径策略，可以获得路径<1,3,5,7,8>，故 RP 算法的结果就仅为<1,3,5,7,8>，故 SP 算法会与 GP 算法结果相同，均为<1,2,3,6,8>和<1,5,4,7,8>。

通过此运行实例可以看出 GP 算法具有如下 3 个优点。

1) 从第 k+1 层树叶节点向树根节点回溯的树根叶子路径一定能够抵达树根节点且长度满足约束 k 的要求，反之则不一定。从图 3 可以看出，从树根节点向下寻找树根叶子路径，有时并不能抵达第 5 层叶子节点 n_5^8，例如一旦树根节点向下选择了节点 n_2^4，则最深抵达第 4 层的节点 8，即 n_4^8，此时路径长度为 3，不满足长度 4 的约束，如果采用此方式寻找一条长度约束路径，不可避免地采用递归查找方法，易知递归查找的时间复杂度为指数形式。从图 3 可以清晰地看出，从叶子节点 n_5^8 向上回溯树根的所有路径长度均为 4。这样在生成一条长度约束的路径过程中，避免了指数形式的递归查找的问题，在 $O(k)$时间复杂度内即可解决。

2) GP 算法采用网树结构不但可以方便地计算出 $N(G,s,t,k)$，而且可以计算出图中每个顶点的总路径数。尽管采用矩阵乘法可以解决这些问题，但是开销过大，因为包含了大量无用的计算。从图 4 中可以看出，GP 算法仅对网树中有限的节点计算其树根叶子路径数，而采用矩阵乘法则需要对所有顶点都计算。更为重要的是，采用网树结构易于从第 k+1 层树叶节点回溯形成一条长度为 k 的树根叶子路径，而矩阵乘法则难于实现。

3) 依据图中顶点的总路径数信息易于构造求解算法。GP 算法就是利用顶点的总路径数信息，从当前节点出发，每次选择优化双亲节点，以此进行求解。

5　实验结果及分析

5.1　实验数据

为了获得较大规模的有向无环图，并获得该图中最大不相交路径数，本文建立了相应的生成算法 DPC(disjoint paths creation)。该算法可以依据用户输入的 n, k 和 d 来随机生成从顶点 1 到顶点 n 路径长度为 k 情况下，最大不交路径数为 w 的有向无环图，这里 d 表示图中边密度，w 为一个大于 0.7 倍理论最大不相交路径数（该数为 $\frac{n-2}{k-2}$）的随机数（注：因为如果最大不相交路径太少且在边密度较高情况下，任何一种算法都相对较容易获得较好质量的解，因此难于用来评价算法的好坏，这一点在后面的实验中也有所体现)。该算法采用的原理如下。

第 1 步　随机生成 $w(k-1)$个不同数字，其范围为 2 到 $n-1$，每个数字表示一个顶点名。

第 2 步　建立一条长度为 k 的路径。每 $k-1$ 个数字与顶点 1 和 n 一起构成一条长度为 k 的路径，具体做法是从前一个数字顶点向后一个数字顶点建立一条有向边，然后顶点 1 向这 $k-1$ 个数字中第一个数字顶点建立一条有向边，最后一个顶点向顶点 n 建立一条有向边。

第 3 步　迭代第 2 步，直到建立 w 条长度为 k 的不相交路径。

第 4 步　随机加边。随机生成 2 个介于 1 到 $n-1$ 之间的数字 a 和 b，判断从 a 到 b 的有向边是否存在，如果存在重新生成；如果 a 与 b 相同，也重新生成；如果会产生回路也重新生成（从顶点 b 出发，依据当前图 G 的状态，采用广度优先搜索策略，如果能够回到顶点 a，说明存在回路），直到可以生成一组满足要求的数字，然后在顶点 a 和 b 之间建立一条有向边。

第 5 步　迭代第 4 步，直到图中边的密度为用户指定的 d 值后停止。

由于第 2 步和第 3 步保证了顶点 1 到顶点 n 之间有 w 条路径长度为 k 的不相交路径，并且第 4 步在生成边的过程中 b 最大为 $n-1$，不能取到 n，这样顶点 n 的入度就仅仅为 w，这样就确保随机生成的有向无环图从顶点 1 到顶点 n 之间最多仅有 w 条

路径长度为 k 的不相交路径。

5.2 实验结果

为了验证本文算法的可行性，将顶点数固定为 n=200。表 1 给出了 k 分别为 5、6 和 7 的情形下，边密度分别为 0.06、0.08、0.1、0.12、0.14、0.16、0.18、0.2、0.25、0.3、0.35、0.4、0.45、0.5、0.55 和 0.6 的情形下，系统随机生成的最大不相交路径数 w。本实验中全部 48 幅有向无环图以及 GP 等 4 种算法的源代码均可以在 http://wuc.scse.hebut.edu.cn/djp/index.htm 下载。

表 1 有向无环图中实际最大不相交路径数 w/个

d	k=5	k=6	k=7
0.06	36	30	27
0.08	41	30	24
0.1	44	34	26
0.12	36	30	25
0.14	44	35	28
0.16	39	32	28
0.18	44	29	28
0.2	42	35	29
0.25	35	28	32
0.3	39	29	28
0.35	40	31	27
0.4	43	29	29
0.45	42	30	26
0.5	42	28	28
0.55	40	31	29
0.6	43	32	28

表 2~表 4 分别给出了在 k=5、k=6 和 k=7 时各个算法找出的最大不相交路径数。

表 2 k=5 时各个算法找出的最大不相交路径数/个

d	LP	RP	SP	GP
0.06	27	29	33	33
0.08	32	30	36	39
0.1	34	36	41	42
0.12	32	32	36	36
0.14	36	36	43	43
0.16	34	37	39	39
0.18	35	37	43	44
0.2	38	35	39	40
0.25	31	34	35	35
0.3	36	35	37	37
0.35	39	38	40	40
0.4	40	37	42	42
0.45	38	40	41	41
0.5	39	42	41	41
0.55	40	39	40	40
0.6	42	38	43	43

表 3 k=6 时各个算法找出的最大不相交路径数/个

d	LP	RP	SP	GP
0.06	21	21	26	26
0.08	23	24	28	28
0.1	24	26	32	32
0.12	23	26	28	28
0.14	28	27	35	34
0.16	28	26	32	32
0.18	26	26	28	28
0.2	28	29	35	35
0.25	24	25	28	28
0.3	27	27	29	29
0.35	31	29	31	31
0.4	27	29	29	29
0.45	28	29	30	30
0.5	28	28	28	28
0.55	30	28	31	31
0.6	31	32	32	32

表 4 k=7 时各个算法找出的最大不相交路径数/个

d	LP	RP	SP	GP
0.06	20	20	24	23
0.08	19	19	24	24
0.1	20	19	25	25
0.12	20	19	24	25
0.14	21	22	27	27
0.16	23	21	27	26
0.18	24	21	27	27
0.2	23	25	28	28
0.25	28	25	31	31
0.3	23	23	27	27
0.35	25	26	27	27
0.4	27	27	28	28
0.45	24	24	25	25
0.5	26	27	28	28
0.55	28	27	28	28
0.6	26	26	27	28

为了清晰地刻画在不同边密度下几种算法识别的正确率，这里将边密度分为较低和较高 2 种，较低的边密度是指边密度在 0.06~0.20 之间；而较高的边密度是指在 0.25 以上，即 0.25~0.60。表 5 给出了在路径长度为 5、6 和 7 以及全部实例下，不同边密度下 4 种算法的平均近似率。

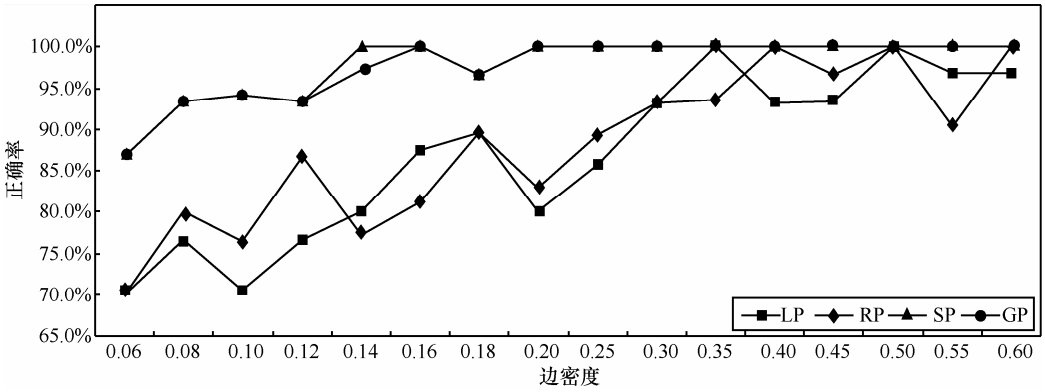

图 8　k=6 时各个算法的正确率

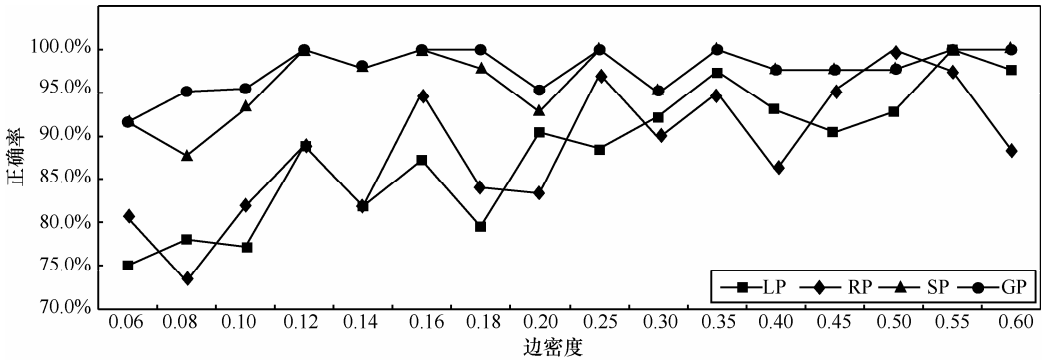

图 9　k=5 时各个算法的正确率

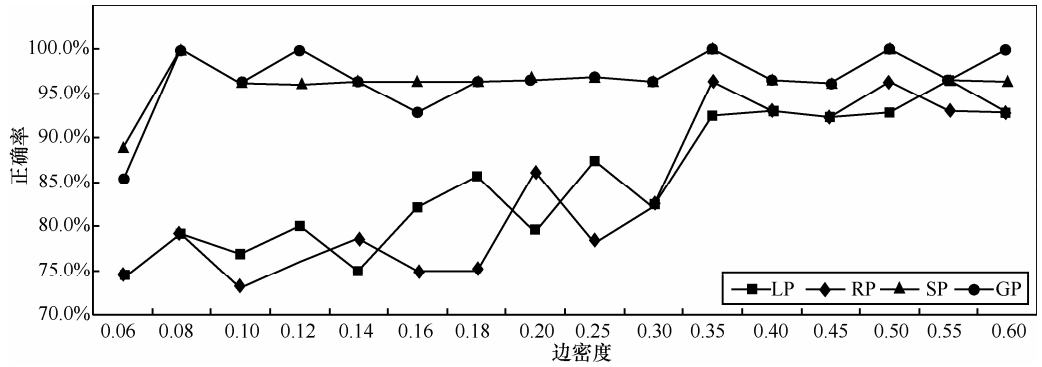

图 10　k=7 时各个算法的正确率

表 5　　**各种情形下各个算法的平均近似率**

算法	k=5		k=6		k=7		全部	
	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60
LP	82.2%	94.1%	78.8%	95.0%	79.1%	91.2%	80.3%	93.5%
RP	83.4%	93.5%	80.4%	95.4%	77.2%	90.3%	80.8%	93.2%
SP	95.1%	98.5%	95.7%	100.0%	95.8%	97.4%	95.5%	98.6%
GP	96.9%	98.5%	95.3%	100.0%	95.3%	97.8%	96.0%	98.7%

表 6 给出了 LP、RP、GP 以及 SP 算法在各种情形下的平均运行时间。

5.3　实验结果分析

1) 通过表 2~表 4，可以很容易地发现 GP 和

表 6 各种情形下各个算法的平均运行时间/ms

算法	k=5		k=6		k=7		全部	
	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60	d=0.06~0.20	d=0.25~0.60
LP	13.3	20.5	15.6	24.4	20.1	28.9	16.3	20.5
RP	13.9	20.1	16.2	24.6	19.0	29.1	16.3	20.5
SP	46.3	103.9	53.3	110.9	55.5	124.2	51.7	82.4
GP	17.4	34.0	20.8	36.5	25.8	41.4	21.3	29.3

SP 算法的实验结果明显好于 LP 和 RP 算法。从这 48 组实例中，可以发现其中的 47 组实例 GP 和 SP 算法的实验结果都显著地好于或等同于 RP 和 LP 算法，例如在表 2 中，当边的密度为 0.08 时，RP 算法仅仅找到 30 条路径，LP 算法找到了 32 条路径，而 GP 算法找到了 39 条路径，SP 算法也找到了 36 条路径，问题解的质量得到了显著的提高。仅有 k=5，边密度为 0.5 这一组实例，RP 算法恰好取得了 42，对比表 1，可知该实例的解是 42，而 GP 和 SP 算法都仅取得了 41。因此通过实验结果可以充分说明 GP 和 SP 算法解的质量优于 LP 和 RP 算法。造成这样现象的原因是 LP 和 RP 算法都是近似随机性算法，随机性算法有可能恰好找到一个实例的最优解。而 GP 和 SP 算法都贪婪算法，是有目的的选择性算法，因此解的质量能够显著优于 LP 和 RP 算法。

2) 伴随着边密度的增高，4 种算法识别正确率也呈现提高趋势。例如在图 8 中，当边的密度为 0.06 时，LP 和 RP 算法的近似率约在 70%，而当边密度达到 0.30 时，这 2 个算法识别正确率接近 95%，而当边密度达到 0.35 时，LP 算法识别近似率竟然达到了 100%，而在图 9 和图 10 中很容易观察到这样的现象。此外，表 5 更是清晰地以数字形式将这一现象刻画出来，可以看到，当边密度较低时，LP 和 RP 算法的识别近似率大约在 80%左右，而当边密度较高时，LP 和 RP 算法都达到了 95%左右。而 GP 和 SP 也有相似的情况，在边密度较低时，评价近似率在 95%以上；当边密度较高时，这 2 个算法近似率都达到或接近 98%以上，特别是当 k=6 时，这 2 个算法识别的正确率都是 100%。

3) LP 和 RP 算法的识别近似率接近相同，GP 和 SP 算法识别近似率也近似相同。由于 LP 和 RP 算法都是属于随机性算法，所以从表 5 中看出，当边密度较低时，这 2 种算法的平均近似率都在 80%左右；而在边密度较高情况下，近似率都在 93%左右，这是由随机算法的随机特性决定的。而 GP 和 SP 算法在边密度较低情况下，近似率达到了 95.5%以上，而边密度较高情况下，近似率可以达到 98.6%以上，这是由于 GP 和 SP 算法都是有目的的选择性算法，且 SP 算法主要是依据 GP 算法的结果进行选择的结果。

4) 从表 6 中不难看出，伴随着 k 值的增加，这 4 种算法的平均求解时间都有所增加。例如 GP 算法在 k=5、k=6 和 k=7 的情况下，运行时间从 17.4 ms 逐渐增大到 25.8 ms。此外，边密度增加也会导致算法的求解时间的增加，这是由于边密度增加，自然会引发图中顶点的最大入度和出度的增加，而 4 种算法的时间复杂度描述中都存在图的入度和出度。由于 LP 和 RP 算法都属于近似随机算法，因此平均运行时间大致相同，都为 16 ms 左右，而 GP 算法由于有一定数量的运算，因此时间略长。SP 算法由于需要从 LP、RP 和 GP 算法的解中进行选择，因此控制策略最为复杂，平均运行时间比 LP、RP 和 GP 算法运行时间总和还略长。综上，表 6 的平均运行时间与 4 种算法时间复杂度分析一致，验证了时间复杂度分析的正确性。

5) 从算法的精确度角度可以看出 GP 算法的精确度最好，通常情况下其都能取得最好的近似率。此外，SP 算法实验结果说明简单地将结果进行拼装，之后从众多结果中选择对图影响最小的路径，这一想法难于进一步提高算法的近似率，甚至在某些情况下有可能导致近似率的降低。而从运行时间角度看，SP 算法的运行时间也大大长于其他 3 种算法。而 GP 算法的运行时间，虽然较随机算法的运行时间略长，但是依然很快，平均 30 ms 左右就可以对 200 个顶点的图问题进行处理，这说明该算法具有很强的实用性。因此 GP 算法无论是在解的质量方面，还是运行速度方面均为非常理想的算法。

6　结束语

本文对求解有向无环图中具有长度约束的最大不相交路径问题进行了研究，利用网树结构快速计算了有向无环图中每个顶点的总路径数，并在此基础上，利用网树数据结构构造了贪婪算法 GP。为了测试 GP 算法的近似性，建立了用于生成测试用例的生成算法 DPC。之后大量实验结果表明 GP 算法具有良好的求解近似性，并具有较低的时间复杂度，充分地验证了 GP 算法的可行性和有效性。

参考文献：

[1] KUMAR A, VARMA S. Geographic node-disjoint path routing for wireless sensor networks[J]. Sensors Journal, IEEE, 2010, 10(6): 1138-1139.

[2] 方效林, 石胜飞, 李建中. 无线传感器网络一种不相交路径路由算法[J]. 计算机研究与发展, 2009, 46(12): 2053-2061.
FANG X L, SHI S F, LI J Z. A disjoint multi-path routing algorithm in wireless sensor network[J]. Journal of Computer Research and Development , 2009, 46(12): 2053-2061.

[3] HASHIGUCHI T, TAJIMA K, TAKITA Y, NAITO T. Node-disjoint paths search in WDM networks with asymmetric nodes[A]. 2011 15th IEEE International Conference on Optical Network Design and Modeling (ONDM) [C]. 2011. 1-6.

[4] 包学才, 戴伏生, 韩卫占. 基于拓扑的不相交路径抗毁性评估方法[J]. 系统工程与电子技术, 2012, 34(1):168-174.
BAO X C, DAI F S, HAN W Z. Evaluation method of network invulnerability based on disjoint paths in topology[J]. Systems Engineering and Electronics, 2012, 34(1):168-174.

[5] GORBENKO A, POPOV V. The problem of finding two edge-disjoint hamiltonian cycles[J]. Applied Mathematical Sciences, 2012, 132(6): 6563-6566.

[6] CYGAN M, MARX D, PILIPCZUK M, PILIPCZUK M. The planar directed *k*-vertex-disjoint paths problem is fixed-parameter tractable[A]. Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium[C]. IEEE, 2013. 197-206.

[7] ITAI A, PERL Y, SHILOACH Y. The complexity of finding maximum disjoint paths with length constraints [J]. Networks, 1982, 12(3): 277-286.

[8] SEGUIN-CHARBONNEAU L, SHEPHERD F B. Maximum edge-disjoint paths in planar graphs with congestion 2[A]. Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium[C]. IEEE, 2011. 200-209.

[9] GUO L, SHEN H. On the complexity of the edge-disjoint min-min problem in planar digraphs[J]. Theoretical Computer Science, 2012, 432: 58-63.

[10] CHEN X B. Unpaired many-to-many vertex-disjoint path covers of a class of bipartite graphs[J]. Information Processing Letters, 2010, 110(6): 203-205.

[11] YU C C, LIN C H, WANG B F. Improved algorithms for finding length-bounded two vertex-disjoint paths in a planar graph and min-max *k* vertex-disjoint paths in a directed acyclic graph [J]. Journal of Computer and System Sciences, 2010, 76 (8): 697-708.

[12] 冯涛, 郭显, 马建峰等. 可证明安全的节点不相交多路径源路由协议[J]. 软件学报, 2010, 21(7): 1717-1731.
FENG T, GUO X, MA J F, *et al*. Provably secure approach for multiple node-disjoint paths source routing protocol[J]. Journal of Software, 2010, 21(7): 1717-1731.

[13] WU B Y. A note on approximating the min-max vertex disjoint paths on directed acyclic graphs [J]. Journal of Computer and System Sciences, 2011, 77 (6): 1054-1057.

[14] WU Y, WU X, MIN F, LI Y. A nettree for pattern matching with flexible wildcard constraints [A]. Proceedings of the 2010 IEEE International Conference on Information Reuse and Integration[C]. Las Vegas, USA, 2010. 109-114.

[15] 李艳, 孙乐, 朱怀忠等. 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[J]. 计算机学报, 2012, 35(10):2194-2203.
LI Y, SUN L, ZHU H, *et al*. A nettree for simple paths with length constraint and the longest path in directed acyclic graphs[J]. Chinese Journal of Computers, 2012, 35(10):2194-2203.

作者简介：

李艳（1975-），女，河北香河人，博士，河北工业大学副教授，主要研究方向为供应链管理与数据挖掘。

武优西（1974-），男，河北故城人，博士，河北工业大学教授，主要研究方向为数据挖掘与智能计算。

黄春萍（1976-），女，河北抚宁人，博士，河北工业大学副教授，主要研究方向为创业管理及其复杂性。

张志颖（1976-），女，河北遵化人，博士，河北工业大学副教授，主要研究方向为信息系统与组织合作。

曾珍香（1965-），女，湖南益阳人，河北工业大学教授、博士生导师，主要研究方向为企业信息资源规划。