SPFA算法的分析及改进

夏正冬 卜天明 张居阳
(华东师范大学上海市可信重点实验室 上海 200062)

摘 要 SPFA(Shortest Path Faster Algorithm)算法是一种对任意有向图求单源最短路径的算法。该算法实现简单,实际运行效果较好,在国内有着比较大的影响力。但遗憾的是,该算法一直缺少正确的理论分析。对该算法进行了分析,指出该算法在不存在源点可达负圈的有向图中,最坏情况运行时间为$\Theta(|V||E|)$;在存在源点可达负圈的有向图中,算法将无限运行下去。对此,给出了改进的SPFA算法,对于任意的有向图,该算法能够在$O(|V||E|)$内运行完毕。最后,从实际运行角度将SPFA算法与其它思想上同源的最短路径算法进行了一系列比较。

关键词 组合算法,单源最短路径,SPFA算法,Bellman-Ford算法
中图法分类号 TP312 **文献标识码** A

Analysis and Improvement of SPFA Algorithm

XIA Zheng-dong BU Tian-ming ZHANG Ju-yang
(Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China)

Abstract SPFA (Shortest Path Faster Algorithm) algorithm is a kind of single-source shortest path algorithm for digraphs. Because its easy implementation and good performance, SPFA has a large influence in the Domestic. But unfortunately, it lacks of correct theoretical analysis. This paper analysed this algorithm both in theoretical and experimental. The paper pointed out that the worst-case running time of SPFA algorithm is $\Theta(|V||E|)$ in the case of digraphs with no negative-weight cycle reachable from the source. In the case of digraphs with negative-weight cycle reachable from the source, the algorithm will run indefinitely. This paper then revisesd the algorithm so that it is $O(|V||E|)$ for any digraphs. Finally, this paper compared SPFA in experiment with other shortest path algorithms which are based on the same idea.

Keywords Combinational algorithm, Single-source shortest paths, SPFA algorithm, Bellman-Ford algorithm

1 概述

最短路径问题是一个经典的组合算法问题,有着重要的实用价值和理论意义,现有的各种导航系统背后的核心算法就是最短路径算法。到目前为止,针对各种各样具体场景下的最短路径算法有几十种。文献[2]中介绍了一些经典的最短路径算法,例如Dijkstra算法、Bellman-Ford算法、Pape-Levit算法、Pallottino算法、Threshold算法等,并从实验的角度进行了分析比较。

1994年,西南交通大学的段凡丁提出了SPFA算法(Shortest Path Faster Algorithm)[1]。该算法本质上是Bellman-Ford算法的一个"队列"实现,其主要思想是用FIFO队列保存被松弛的结点,并且不断迭代以求得最短路径。

SPFA的伪代码如下:

算法1 SPFA

```
输入:有向图G,源点s
输出:数组d[]
1.    FOR EACH v∈Vertex
2.        d[v]=+∞;
3.        flag[v]=false;
4.    d[s]=0;
5.    Q={s};
6.    flag[s]=true;
7.    WHILE(Q≠Φ)
8.        u=Q.DEQUEUE();
9.        flag[u]=false;
10.       FOR EACH v∈Adj[u]
11.           IF d[v]>d[u]+ω(u,v)
12.               d[v]=d[u]+ω(u,v);
13.               IF flag[v]==false
14.                   Q.ENQUEUE(v);
15.                   flag[v]=true;
16. return d;
```

它的具体执行过程如下:第1-3行进行初始化,其中$flag$是一个长度为$|V|$的数组,用来标记节点是否在队列中;如果$flag[v]$=false,表示结点v不在队列中。初始化时将每个节点的$flag$值都设置为false。d是一个长度为$|V|$的数组,表示节点的最短路径估计,等到算法执行结束时,此数

到稿日期:2013-07-24 返修日期:2013-11-03 本文受国家自然基金青年基金(61003068)和华东师范大学科研创新基金资助。
夏正冬(1991-),男,硕士生,主要研究方向为组合算法的设计与分析,E-mail:xiazdong@126.com;卜天明(1977-),男,博士,副教授,主要研究方向为组合算法的设计与分析、算法博弈论;张居阳(1978-),男,博士,副教授,主要研究方向为智能调度与约束规划。

组中存储了每个节点的最短路径值。第4－6行是将源点 s 加入队列，$flag[s]$=true，并初始化 $d[s]$=0。第7－15行的循环中，每次循环都从队列中取出一个节点，并松弛与它相邻的每个节点 v，如果 v 能够被松弛且原本不在队列中，则将 v 放入队尾。当队列为空时，算法跳出循环并终止。

SPFA 自从1994年被提出以来，引起了广泛的讨论。在网络上，类似于“SPFA 和 Dijkstra 哪个快?”等问题被人们纷纷提出，而回答也各不相同。谷歌搜索“SPFA algorithm”关键字可以得到15900条结果，百度搜索“SPFA 算法”则可以得到49500条结果。维基百科[1]和百度百科[2]也对 SPFA 做了详细介绍，甚至在《国家集训队2009年论文集》[3]中也有关于 SPFA 的详细介绍。

此外，现在还提出了一些基于 SPFA 的改进算法。在文献[1]中就介绍了两个这样的算法：SLF(Small Label First)和 LLL(Large Label Last)。这两个改进算法的思想都是基于先松弛离源点“较近”的节点(所谓“较近”指的是 d 值较小)：

(1)SLF：将 FIFO 队列改为双端队列。设要加入的节点是 v，当前的队首元素为 u，若 $d[v]<d[u]$，则将 v 插入队首，否则插入队尾。

(2)LLL：仍然是 FIFO 队列。设队首元素为 u，队列中所有 d 值的平均值为 x，若 $d[u]>x$，则将 u 出队后不松弛 u 的相邻节点，并将 u 插入到队尾；若 $d[u]\leqslant x$，则 u 出队后对 u 相邻节点进行松弛操作。

可见 SPFA 无论是在民间，还是在学术界，都有着较大的影响力。

本文对 SPFA 算法从理论上做了仔细分析，并指出了文献[1]中关于 SPFA 的一些错误观点：

(1)原文中指出 SPFA 的运行时间始终为 $O(|E|)$，而实际上 SPFA 的最坏情况运行时间为 $\Theta(|V||E|)$。

(2)原文中指出 SPFA 适用于任意有向图，而实际上 SPFA 并不适用于任意有向图。本文进而对 SPFA 进行了修正，使之能适用于任意有向图。

本文还对 SPFA 算法在有源点可达负圈和无源点可达负圈的任意图、稀疏图、稠密图等各种情况下的实际运行性能进行了详细的实验分析，并将它与 Bellman-Ford 算法进行了比较。结果表明，在不存在源点可达负圈的有向图中，SPFA 比 Bellman-Ford 的性能更好，而在存在源点可达负圈的有向图中，SPFA 不如 Bellman-Ford 算法。而经过 SLF 优化或 LLL 优化的 SPFA 算法并不能在任意的图中都对 SPFA 算法进行优化，SLF 与 LLL 优化的 SPFA 算法的最坏情况运行时间为指数级，即最坏情况复杂度远远高于普通的 SPFA 算法的最坏情况复杂度，因此这些优化方法是很不稳定的。

本文第2节讨论了在不存在源点可达负圈的有向图中，SPFA 算法的正确性及效率，得出 SPFA 的最坏情况运行时间为 $\Theta(|V||E|)$，最佳情况运行时间为 $\Theta(|E|)$；第3节改进了 SPFA 算法，使之能够在任意的有向图中正确运行，并且能够判断图中是否存在源点可达的负圈；接着讨论了该算法的正确性及效率；第4节分别对改进的 SPFA 算法、经过 SLF 优化的 SPFA 算法、经过 LLL 优化的 SPFA 算法与 Bellman-Ford 算法在各种情形下的实际运行性能进行了实验比较。

下面将简单介绍本文会用到的一些符号及术语，这些符号和术语都来自文献[4]：

(1)$\omega(u,v)$表示 u 到 v 这条边的权重。

(2)松弛：每个节点 v 都有一个 $d[v]$值，用来描述从源点 s 到 v 的最短路径上权值的上界。一次松弛操作可以减小 $d[v]$，并更新 v 的前趋域 $\pi[v]$。松弛操作的伪代码如下：

算法2 RELAX

输入：节点 u，节点 v

输出：无

```
1. IF(d[v]>d[u]+ω(u,v))
2. d[v]=d[u]+ω(u,v);
3. π[v]=u;
```

(3)$\delta(s,v)$表示从源点 s 到节点 v 的最短路径值。

2 图中不含源点可达负圈的情况

这里我们假设图中不存在源点可达的负圈。下面将证明在这个前提下 SPFA 的算法正确性，并分析最佳情况运行时间和最坏情况运行时间。

2.1 SPFA 的正确性

定理1 如果图中不存在源点可达的负圈，则 SPFA 能够正确地计算出最短路径。

证明：(反证法)假设 SPFA 执行之后，存在某个点 x 的最短路径计算错误，则从源点 s 到 x 的最短路径(设为 P)中找到第一个计算最短路径错误的点，不妨设它为 v，即 $d[v]>\delta(s,v)$。设 P 中 v 的前驱是 u。因为 $d[u]=\delta(s,u)$，因此 $\delta(s,v)=d[u]+\omega(u,v)<d[v]$。又因为 u、v 都是源点可达的，因此 $d[u]<\infty$。$d[u]$被松弛为 $\delta(s,u)$时一定会进入队列，当 u 离开队列时，由于 $d[v]>d[u]+\omega(u,v)$，因此 u 一定会松弛 v，使得 $d[v]=\delta(s,v)$，这与假设 $d[v]>\delta(s,v)$矛盾，因此 SPFA 执行后必能得出最短路径。

2.2 SPFA 的效率

定理2 SPFA 的最佳运行时间为 $\Theta(|E|)$。

证明：因为求单源最短路径至少要遍历全部的边一次，所以一个单源最短路径算法至少是 $\Omega(|E|)$。接着我们要证明给定结点个数 n，边个数 m，假设图是弱连通的，需要构造出一个图，使得对这张图运行 SPFA 算法的运行时间为 $\Theta(|E|)$。构造图的形式化描述如下：

(1)对于 $\forall v_i, v_{i+1}$，其中 $1\leqslant i\leqslant n-1$，$(v_i,v_{i+1})\in E$，且 $\omega(v_i,v_{i+1})=1$，即构成了线性链，此时添加了 $n-1$ 条边。

(2)对于剩下的 $m-n+1$ 条边，添加任意非自环边，且边的权值足够大即可，这里的权重为 $+\infty$。

因此构造的图如图1所示。

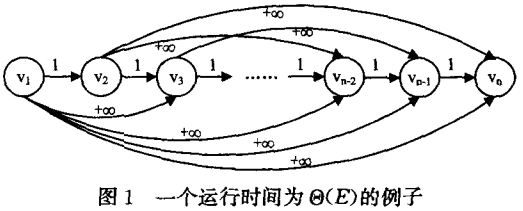

图1 一个运行时间为 $\Theta(E)$的例子

1) Shortest Path Faster Algorithm, http://en.wikipedia.org/wiki/Shortest_Path_Faster_Algorithm 2013,2,3

2) SPFA, http://baike.baidu.com/view/682464.htm 2013,2,1

对于图1,每个节点都只会进入队列一次,并且根据聚合分析,共访问了$|E|$条边,因此SPFA的运行时间为$\Theta(|E|)$。

引理1 对于不存在源点可达负圈的图,若s到v_k的最短路径上的边的个数为k,则v_k至多进入队列k次。

证明:设s到v_k的最短路径为$s\to v_1\to v_2\to v_3\to\cdots\to v_{k-1}\to v_k$,根据算法的执行过程可知,$s$首先会进入队列,当$s$出队列时一定会松弛$v_1$,且$d[v_1]=\delta(s,v_1)$;而且在$s$出队到$v_1$入队的过程中,$v_k$最多进入队列1次。$v_1$出队列时一定会松弛$v_2$,且$d[v_2]=\delta(s,v_2)$;而且在$v_1$出队到$v_2$入队的过程中,$v_k$最多进入队列1次。以此类推,$v_{k-1}$出队时,一定会松弛$v_k$,且$d[v_k]=\delta(s,v_k)$。因此在$v_{k-1}$出队后,$v_k$最多再进入队列1次,所以$v_k$最多进入队列$k$次。

引理2 对于不存在源点可达负圈的图,任意节点v至多进入队列$|V|-1$次。

证明:因为s到任何一个节点的最短路径的边的个数至多为$|V|-1$,又根据引理1,得出任何一个节点至多进入队列$|V|-1$次。

定理3 SPFA的最坏运行时间为$O(|V||E|)$。

证明:因为所有节点出队1次所访问的边的总和为$|E|$,而每个节点至多出队$|V|-1$次(引理2),因此对于不存在源点可达负圈的有向图,SPFA的运行时间为$O(|V||E|)$。

定理4 SPFA的最坏运行时间为$\Omega(|V||E|)$。

证明:给定一个结点个数n,边个数m,假设图是弱连通的,我们需要构造出一个图,使得对这张图运行SPFA的运行时间为$\Omega(|V||E|)$。

(1)如果$n-1\leqslant m\leqslant 2n-3$

构造图的形式化描述如下:

1)对于$\forall v_i,v_{i+1}$,其中$1\leqslant i\leqslant n-1$,$(v_i,v_{i+1})\in E$,且$\omega(v_i,v_{i+1})=1$,即构成了线性链,此时添加了$n-1$条边。

2)对于剩下的$m-n+1$条边,以v_1为起点,分别向v_3,v_4,$\cdots$,v_{m-n+3}添边,其中$\omega(v_1,v_i)=2i+1$。

因此构造的图如图2所示。

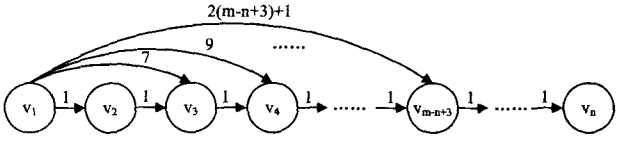

图2 $n-1\leqslant m\leqslant 2n-3$的构造

因为$n-1\leqslant m\leqslant 2n-3$,因此$m=\Theta(n)$。我们要计算所有点的入队次数和出度:

v_1出度为$m-n+2$,入队1次。

v_2出度为1,入队为1次。

v_3出度为1,入队为2次(分别为v_1先松弛v_3,v_2再松弛v_3)。

v_4出度为1,入队为3次。

……

v_{m-n+3}出度为1,入队为$m-n+2$次。

v_{m-n+4},v_{m-n+5},$\cdots$,v_{n-1}(共$2n-m-4$个节点)的出度为1,入队为$m-n+2$次。

v_n出度为0,入队为$m-n+2$次。

所以总共的访问的边数为:

$(m-n+2)+1*(1+2+3+\cdots+(m-n+2))+(2n-m-4)*(m-n+2)+(m-n+2)*0$

$=\Omega(m^2)=\Omega(mn)=\Omega(|V||E|)$

(2)如果$m>2n-3$

1)对于$\forall v_i,v_{i+1}$,其中$1\leqslant i\leqslant n-1$,$(v_i,v_{i+1})\in E$,且$\omega(v_i,v_{i+1})=1$,即构成了线性链,此时添加了$n-1$条边。

2)以v_1为起点,分别向$v_3,v_4,\cdots,v_n$添边,$\omega(v_1,v_i)=2i+1$,此处用去了$n-2$条边。

3)分别以$v_n,v_{n-1},\cdots,v_2$为起点添加权重为$+\infty$的任意非自环边,并且不断循环添加。

因此构造的图如图3所示。

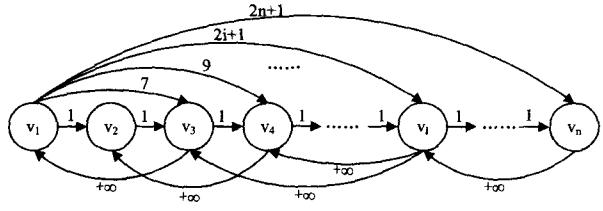

图3 $m>2n-3$的构造

我们要计算所有点的入队次数和出度:

v_1出度为$n-1$,入队1次。

v_2到v_n出度至少为$(m-2n+3)/(n-1)$。

v_2到v_n的入队次数分别为$1,2,3,\cdots,n-1$。

所以总共访问边数至少为:

$(n-1)*1+((m-2n+3)/(n-1))(1+2+\cdots+(n-1))$

$=(n-1)+((m-2n+3)/(n-1))*(n-1)n/2$

$=(n-1)+(m-2n+3)n/2=\Omega(mn)=\Omega(|V||E|)$

根据定理3和定理4,可以得出SPFA的最坏情况运行时间为$\Theta(|V||E|)$。

3 图中含源点可达负圈的情况

文献[1]中并没有提及有向图中是否含有源点可达负圈的情况,而事实上原来的SPFA并不适用于存在源点可达负圈的图。

定理5 如果图中存在源点可达的负圈,则SPFA算法会一直运行下去。

证明:如果图中存在源点可达的负圈,设负圈为$v_1\to v_2\to v_3\to\cdots\to v_n\to v_1$。由于负圈是源点可达的,因此$s$一定会松弛$v_1$,然后$v_1$松弛$v_2$,$v_2$松弛$v_3$,$\cdots$,$v_n$松弛$v_1$,并且会无限循环下去,因此SPFA不会结束。

比如图4,如果对这张图运行SPFA,则会无限循环下去。

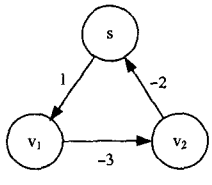

图4 一个SPFA会一直运行下去的例子

3.1 修正的SPFA

下面给出适用于任何有向图的SPFA算法的一个简单修正,它面对任何有向图,都不会无限运行下去。该算法利用了

引理 2，即如果存在某个节点进入队列的次数大于$|V|-1$，则说明该图中存在负圈。算法伪代码如下：

算法 3 SPFA_revised(G,s)

```
输入：有向图 G 和源点 s
输出：true 或 false
1.    FOR EACH v∈Vertex
2.        d[v]=+∞;
3.        flag[v]=false;
4.        count[v]=0;
5.    d[s]=0;
6.    Q={s};
7.    count[s]++;
8.    flag[s]=true;
9.    WHILE(Q≠Φ)
10.       u=Q.DEQUEUE();
11.     flag[u]=false;
12.       FOR EACH v∈Adj[u]
13.           IF d[v]>d[u]+ω(u,v)
14.               d[v]=d[u]+ω(u,v)
15.               IF flag[v]==false
16.                   Q.ENQUEUE(v);
17.             count[v]++;
18.               IF count[v]≥|V|
19.                   return false;
20.               flag[v]=true;
21.   return true;
```

3.2 修正的 SPFA 算法的正确性和时间复杂性

定理 6 当某个顶点 v 进入队列次数超过$|V|-1$次时，则说明有源点可达负圈。

证明：根据引理 2，可得如果某个节点进入队列超过$|V|-1$次，则说明有源点可达的负圈。

容易看到，在修正的 SPFA 算法中，每个节点入队的次数不会超过$|V|$次，因此修正的 SPFA 算法的最坏情况运行时间仍为$\Theta(|V||E|)$，最佳情况运行时间也还是$\Theta(|E|)$。

4 实验

本次实验分为 3 个大场景：任意图、稀疏图、稠密图，并且每个大场景下分两个小场景：存在源点可达负圈的图、不存在源点可达负圈的图。

由于 SPFA 算法本质上是 Bellman-Ford 算法的一个“队列”的实现，因此在实验中，我们不仅实现了修正的 SPFA 算法，而且还实现了 Bellman-Ford 算法。我们将这两个算法放在同一场景下进行比较，以示 SPFA 算法对 Bellman-Ford 算法的改进程度。由于原始的 Bellman-Ford 算法对所有的边总要松弛$|V|-1$次，这将使得比较没有任何悬念；而且在实际中一般也不会使用不加任何优化的 Bellman-Ford 算法。因此，在这里我们将修正的 SPFA 算法、经过 SLF 优化的 SPFA 算法、经过 LLL 优化的 SPFA 算法和经过“简单的优化”的 Bellman-Ford 算法进行比较。这里所谓的“简单的优化”，即指当经过一次对所有边的松弛操作之后，若没有任何节点的 $d[]$ 发生改变，就终止运行[4]。由于 SLF 和 LLL 优化的 SPFA 算法在源点可达负圈的有向图中也会无限运行下去，而即使有向图中不存在源点可达负圈，这两个算法在某种特定的松弛顺序下会使得某个节点进入队列次数为指数次[5]，大大超过了$|V|-1$次，因此并不能像修正 SPFA 那样通过判断是否存在某个节点进入队列超过$|V|-1$次的方法判断有向图是否存在源点可达负圈。因此在存在源点可达负圈的有向图中只是将修正的 SPFA 算法与经过简单优化的 Bellman-Ford 算法进行比较，而没有包括 SLF 和 LLL 优化的 SPFA 算法。

整个实验的环境是 Pentium(R) Dual-Core CPU E5800@3.2GHz，内存 4GB，操作系统为 Windows 7。

4.1 随机生成图的实现

4.1.1 随机生成不存在源点可达负圈的图

由于判断一个图中是否存在负圈的时间复杂度较高，因此为了保证所生成的随机图一定不含负圈，我们在这里令边的权重取值为$[0,2m]$，其中 m 为边的个数（注意我们在这里将边的权重设为正仅仅是为了保证不会出现负圈，修正的 SPFA 算法和 Bellman-Ford 算法本身都可以在任意图上运行）。此外我们还要求生成的随机图中没有自环和重边。伪代码如下：

算法 4 generate_random_no_negative_cycle

```
输入：点的个数 n，边的个数 m
输出：有向图 g
1. Graph g=new Graph(n);
2. FOR(i=1 to m)
3.    edge=generate_random_edge();
4.    g.addEdge(edge);
5. return g;
```

4.1.2 随机生成存在源点可达负圈的图

这里生成源点可达负圈的图的方法是先随机生成图，然后检测是否存在负圈，如果不存在，则重新生成。这里生成的随机图没有自环、重边，边的权重范围为$[-2m,2m]$，其中 m 为边的个数。伪代码如下：

算法 5 generate_random_negative_cycle

```
输入：点的个数 n，边的个数 m
输出：有向图 g
1. WHILE(true)
2.    Graph g=new Graph(n);
3.    FOR(i=1 to m)
4.    edge=generate_random_edge();
5.      g.addEdge(edge);
6.    IF g has negative cycle
7.      return g;
```

4.2 任意图

我们固定节点个数$|V|=800$，而将$|E|$按照$2|V|$，$3|V|$，$4|V|$，…，$150|V|$增长，对于每个$(|V|,|E|)$对，用 4.1 节中的方法随机生成一张对应的图，共生成了 149 张图。为了让实验结果更准确，对于无源点可达负圈的图运行 SPFA 算法、经过 SLF 优化的 SPFA 算法、经过 LLL 优化的 SPFA 算法和 Bellman-Ford 算法各 100 次，再求算术平均值作为最后的结果。折线图以$|E|$为横坐标，时间（秒）为纵坐标。

实验结果如图 5、图 6 所示。

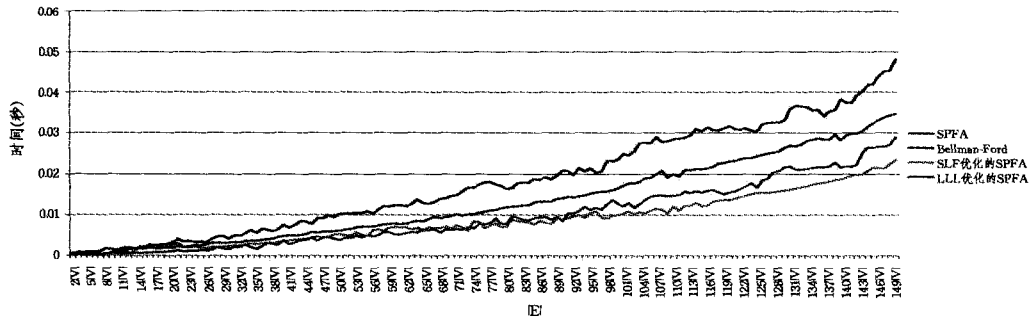

图 5　无源点可达负圈

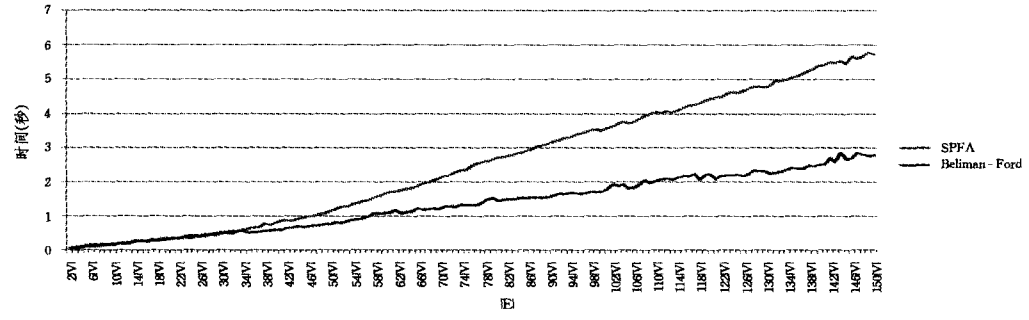

图 6　有源点可达负圈

4.3　稀疏图

这里假定$|E|=4|V|$,$|V|$按照$30\times1,30\times2,30\times3,\cdots,30\times100$增长,对于每个($|V|$,$|E|$)对,用 4.1 节中的方法随机生成一张对应的图,因此共生成了 100 张图。为了让实验结果更准确,对于无源点可达负圈的图运行 SPFA 算法、经过 SLF 优化的 SPFA 算法、经过 LLL 优化的 SPFA 算法和 Bellman-Ford 算法各 100 次,再求算术平均值作为最后的结果。对于有源点可达负圈的图运行 SPFA 算法和 Bellman-Ford 算法各 100 次,再求算术平均值作为最后的结果。折线图是以节点个数$|V|$为横坐标,时间(秒)为纵坐标。

实验结果如图 7、图 8 所示。

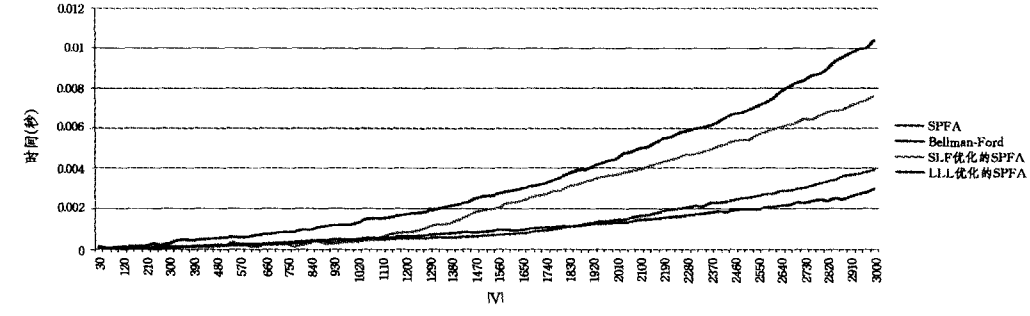

图 7　无源点可达负圈

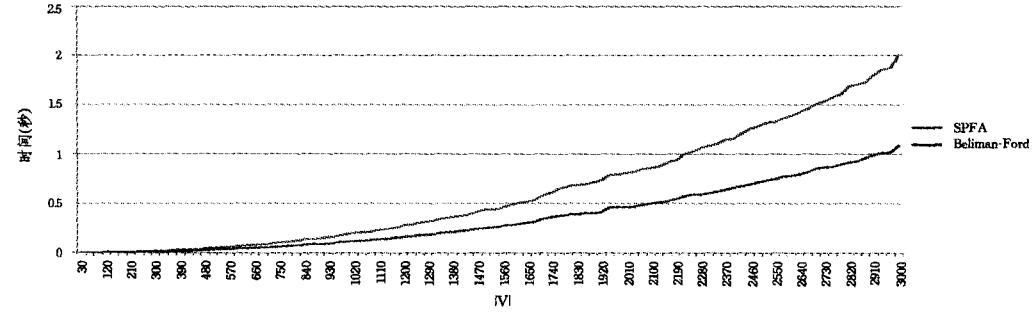

图 8　有源点可达负圈

4.4　稠密图

这里假定$|E|=|V|^2/4$,$|V|$按照$10\times1,10\times2,10\times3,\cdots,10\times100$增长,对于每个($|V|$,$|E|$)对,用 4.1 节中的方法随机生成一张对应的图,因此共生成了 100 张图。为了让实验结果更准确,对于无源点可达负圈的图运行 SPFA 算法、经过 SLF 优化的 SPFA 算法、经过 LLL 优化的 SPFA 算法和 Bellman-Ford 算法各 100 次,再求算术平均值作为最后的结果。对于有源点可达负圈的图运行 SPFA 算法和 Bellman-Ford 算法各 100 次,再求算术平均值作为最后的结果。折线

(下转第 213 页)

[9] Castro P A D, Zuben F J V. Multi-objective Feature Selection Using a Bayesian Artificial Immune System[J]. International Journal of Intelligent Computing and Cybernetics, 2010, 3(2): 235-256

[10] Ciccazzo A, Conca P, Nicosia G, et al. An Advanced Clonal Selection Algorithm with Ad Hoc Network-Based Hypermutation Operators for Synthesis of Topology and Sizing of Analog Electrical Circuits[C]// Artificial Immune Systems-7th International Conference, ICARIS 2008, Proceedings, 2008. Tiergartenstrasse 17, Heidelberg, D-69121, Germany: Springer Verlag, 2008: 60-70

[11] Wilson W O, Garrett S M. Modelling Immune Memory for Prediction and Computation[C]// 3rd International Conference in Artificial Immune Systems, ICARIS2004, 2004. Catania, Sicily, Italy: Springer, 2004: 386-399

[12] 陶媛, 吴耿锋, 胡珉. 一种基于进化与免疫的动态多目标人工免疫系统模型[J]. 计算机科学, 2010, 37(1): 217-22

[13] De Castro L N, Von Zuben F J. Clonal selection algorithm with engineering applications[C]// GECCO 2002-Workshop Proceedings, 2000. Las Vegas: Proc. of GECCO'00, 2000: 36-37

[14] Aickelin U, Greensmith J. Sensing danger: Innate immunology for intrusion detection[J]. Information Security Technical Report, 2007, 12(4): 218-227

(上接第 184 页)

图以|V|为横坐标，时间(秒)为纵坐标。实验结果如图 9、图 10 所示。

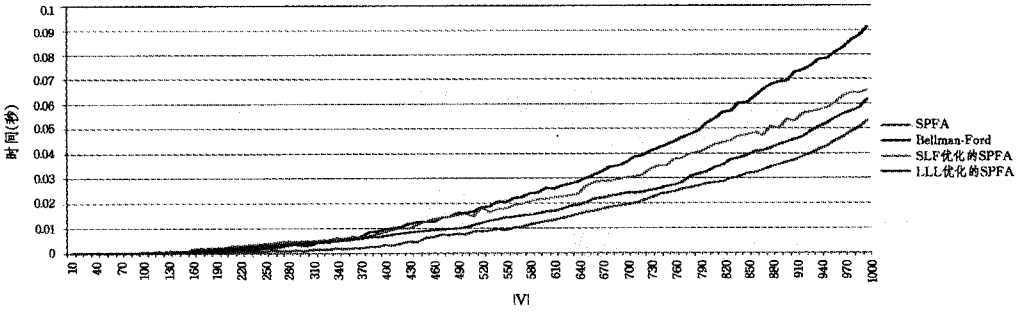

图 9 无源点可达负圈

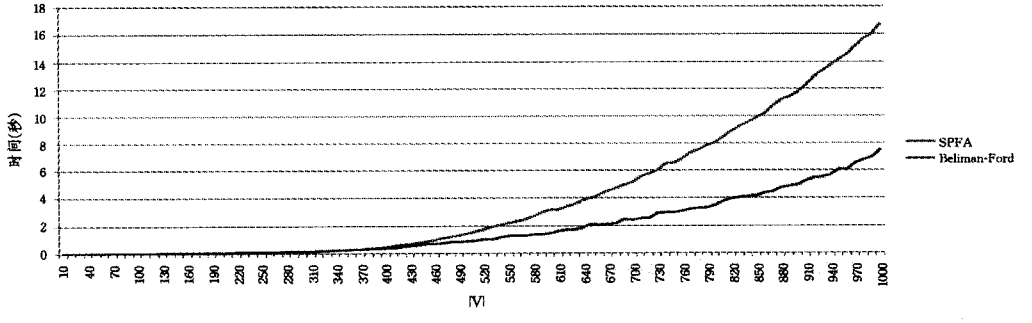

图 10 有源点可达负圈

结束语 从实验结果中我们可以明显看出：无论是稀疏图，还是稠密图，当图中不包含负圈的时候，SPFA 算法的实际运行时间都要好于经过简单优化的 Bellman-Ford 算法；当图中存在负圈的时候，则 SPFA 算法的实际运行时间要差于经过简单优化的 Bellman-Ford 算法。经过 SLF 或 LLL 优化的 SPFA 算法并不能始终对 SPFA 算法进行优化，甚至最坏情况复杂度为指数级，远远高于普通的 SPFA 算法，因此在一般情况下并不会使用这些优化。在实际情况中，我们一般只能事先知道我们所处理的问题是否会有负边出现的情况，而无法知道若出现负边，是否会有可能出现负圈。如果我们事先知道了所处理的情况不会出现负边，虽然 SPFA 算法这时会优于 Bellman-Ford 算法，但可以采用性能更优的 Dijkstra 算法；若我们事先只知道有可能会出现负边，但不能保证是否一定不会出现负圈，那么这时 SFPA 算法并不会比 Bellman-Ford 算法更有优势。

参 考 文 献

[1] 段凡丁. 关于最短路径的 SPFA 快速算法[J]. 西南交通大学学报, 1994, 29(2): 207-212

[2] Cherkassky B V, Goldberg A V, Radzik T. Shortest paths algorithms: Theory and experimental evalutation [J]. Mathematical Programming, 1996, 73(2): 129-174

[3] 姜碧野. SPFA 的优化与应用[C]// 国家集训队 2009 年论文集. 2009

[4] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms(Third Edition) [M]. MIT Press, Cambridge, MA, 2009: 651-655

[5] Chen Z L, Powell W B. A note on bertsekas' small-label-first strategy [J]. Networks, 1997, 29: 111-116