doi: 10.12068/j.issn.1005-3026.2017.07.026

适用于大规模网络的全源最短路径重优化算法——RASP算法

江锦成[1], 吴立新[2,3], 张媛媛[4], 刘善军[2]

(1. 北京师范大学 减灾与应急管理研究院, 北京 100875; 2. 东北大学 资源与土木工程学院, 辽宁 沈阳 110819; 3. 中南大学 地球科学与信息物理学院, 湖南 长沙 410083; 4. 中国矿业大学(北京) 地球科学与测绘工程学院, 北京 100083)

摘 要: 为提升大规模网络全源最短路径的求解效率,基于重优化理论提出了一种快速的精确全源最短路径求解方法——RASP(reoptimization-based all-pairs shortest path)算法. 分析了异源最短路径树间的相关性和差异性;在已知单源最短路径树的基础上,基于重优化理论实现了异源最短路径树间的高效转换,进而得出高效求解全源最短路径的RASP算法;理论证明RASP算法的时间复杂度为$O(3n^2+2nm)$. 实验测试表明:无论是在稀疏还是稠密网络上,RASP算法都能有效地超越Floyd算法、n次Dijkstra算法及其改进算法.

关 键 词: 重优化;全源;最短路径;大规模网络;Floyd算法;Dijkstra算法

中图分类号: TP 301.6 **文献标志码**: A **文章编号**: 1005-3026(2017)07-1037-06

A Reoptimization-Based All-Pairs Shortest Path Algorithm for Large-Scale Network—RASP Algorithm

JIANG Jin-cheng[1], *WU Li-xin*[2,3], *ZHANG Yuan-yuan*[4], *LIU Shan-jun*[2]

(1. Academy of Disaster Reduction and Emergency Management, Beijing Normal University, Beijing 100875, China; 2. School of Resources & Civil Engineering, Northeastern University, Shenyang 110819, China; 3. School of Geoscience and Info-Physics, Central South University, Changsha 410083, China; 4. College of Geoscience and Surveying Engineering, China University of Mining & Technology, Beijing 100083, China. Corresponding author: WU Li-xin, E-mail: awulixin@263.net)

Abstract: In order to improve the computational performance of searching all-pairs shortest paths in a large-scale network, an exact all-pairs shortest path method—RASP algorithm is proposed based on the reoptimization theory. First, the correlation and difference between the shortest path trees with different sources are analyzed. Second, the efficient conversion from a known single-source shortest path tree to another one with different source is achieved based on the reoptimization-based theory. Furthermore, the reoptimization-based all-pairs shortest path (RASP) algorithm utilizes this conversion method to calculate efficiently all-pairs shortest paths. At last, the time complexity of RASP algorithm is proved to be $O(3n^2+2nm)$. The experimental test demonstrates that RASP algorithm gains the advantage over Floyd, n-Dijkstra algorithms and their improved algorithms in both sparse and dense networks.

Key words: reoptimization; all-pairs; shortest path; large-scale network; Floyd algorithm; Dijkstra algorithm

作为图论的经典问题,最短路径在诸多领域都有着广泛的应用,如:智能交通系统[1]、地理信息系统[2-4]、计算机网络与通讯[5]等. 为提升算法的计算效率或满足特定的应用需求,已有大量优秀算法[6-8]被提出. 按照解的精度,最短路径算法可分为精确和近似算法;按照求解目标的数量,最短路径问题又可分为单源和全(多)源最短路径. 本文的研究对象是精确全源最短路径问题,即旨

收稿日期: 2016-04-29

基金项目: 国家高技术研究发展计划项目(2011AA120302).

作者简介: 江锦成(1989-),男,江西九江人,北京师范大学博士研究生; 吴立新(1966-),男,江西宜春人,东北大学教授,博士生导师,教育部"长江学者奖励计划"特聘教授; 刘善军(1965-),男,河北涿鹿人,东北大学教授,博士生导师.

在求解所有点对之间的精确最短路径.

目前,求解全源最短路径问题的典型算法包括 Floyd 算法[9] 和多次 Dijkstra 算法[10]. 其中,Floyd 算法的时间复杂度为 $O(n^3)$;Dijkstra 算法是单源最短路径算法,其时间复杂度为 $O(n^2)$. 为获得全源最短路径,需执行 n 次,故总复杂度为 $O(n^3)$. 由此可见,全源最短路径算法的时间复杂度依然很高,难以满足现有大规模网络或实时计算的要求. 为进一步提升算法的计算效率,除设计低复杂度的高效算法外,还有学者[11] 采用并行计算技术来加速全源最短路径问题的求解,但并行计算不能减少总的计算量.

重优化方法的基本思想是充分利用异源最短路径树间的相关性,避免重复计算,从而达到高效求解全源最短路径的目的. 该方法是在已有最短路径树的基础上,针对网络的动态变化而对路径进行重新调整的过程. 目前,其已被用于解决动态最短路径问题,如单边权重变化、多边权重变化[12],以及源点变化[13] 等情景. 由于仅计算最短路径可能发生变化的节点,故重优化方法能有效减小搜索空间,提升求解速度. 其中,确定最小搜索空间以及分析最短路径的变化规律是重优化方法的两个重要内容. 本文首先分析源点发生变化时,异源最短路径树间的相关性与差异性,并依此设计高效的转换方法;随后将该方法扩展至全源最短路径问题的求解.

1 相关工作

将现实网络抽象成无向图 $G(V,E,W)$,其中 V 为节点集合,E 为边集合,W 为权重集合,$n=|V|$ 为节点数量,$m=|E|$ 为边数量. 对任意一条边 $(i,j)\in E, i\in V, j\in V$ 分别为边 (i,j) 的两个端点,$w(i,j)\in W$ 表示网络边 (i,j) 的权重. 单源最短路径问题是求解从起始点 s(或所有点)到终点 t 具有最小总权重的最短路径. 全源最短路径则是求解所有点对间的最短路径. 通常,赋予每个节点 i 一对标号 (d_i,p_i),其中 d_i 表示 i 到 t 的最短距离,p_i 表示最短路径上 i 的前驱节点,$d(i,j)$ 表示节点 i 和 j 间的距离.

Dijkstra 算法是求解单源最短路径最常用的方法,其主要步骤包括:1) 将 V 分解成 S 和 T,设置 $S=\{s\}$,$T=V-S$,$d_s=0$,$d_i=\infty$, $\forall i\in V/s$;2) 搜索 T 中距离 S 最近的节点 j,计算 d_j;3) 设置 $S=S\cup\{j\}$, $T=T-\{j\}$;4) 若 $T=\varnothing$ 或 $t\in S$,算法终止;否则转至步骤 2).

Floyd 算法是一种动态规划算法,适用于求解全源最短路径,主要步骤包括:1) 初始化任意点对 (i,j) 间的距离,$d(i,j)=w(i,j)$;若 $(i,j)\notin E$,则 $d(i,j)=\infty$;2) 对任意点对 (u,v),检查是否存在节点 x 使得 $d(u,x)+d(x,v)<d(u,v)$, 若存在则设置 $d(u,v)=d(u,x)+d(x,v)$;3) 重复步骤 2) 至所有点对间的距离都满足最优条件.

2 异源最短路径树

当 Dijkstra 算法计算出网络所有节点到目的节点 t 的最短路径后,可生成一棵最短路径树(见图 1a). 对于相邻的目的节点 t_1 和 t_2,其对应的两棵异源最短路径树 $T(t_1)$ 和 $T(t_2)$ 存在着很大的相关性. 本文以图 1 为例,分析两棵异源最短路径树间的相关性与差异性.

如图 1b 所示,$T(v_{10})$ 包含 3 棵子树:$T'(v_7)$,$T'(v_8)$ 和 $T'(v_9)$. 为方便表达,$T(v_7)$ 表示图 1b 中以 v_7 为根节点的子树,$T'(v_7)$ 则为图 1c 中以 v_7 为根节点的最短路径树. 对任意节点 i,d_i 表示在 $T(v_{10})$ 中 i 到 v_{10} 的最短距离,而 d'_i 则表示在 $T'(v_7)$ 中 i 到 v_7 的最短距离,$d(i,j)$ 表示节点 i 与 j 间的最短距离. 对比图 1b 和图 1c 可知:

1) 子树 $T'(v_7)$ 的结构保持不变,但其上所有点的最短距离全部减少 2.

证明 对于任意节点 $i\in T'(v_7)$,$d_i=d(i,v_{10})=d(i,v_7)+d(v_7,v_{10})$,而 $d'_i=d(i,v_7)=d_i-d(v_7,v_{10})$. 由于 $d(v_7,v_{10})$ 是全局最短距离,为固定常量,故 $T'(v_7)$ 上所有点的最短距离都减少 $d(v_7,v_{10})=2$,结构保持不变.

2) 对任意节点 $j\in T(v_{10})-T'(v_7)$,都存在 $|d'_j-d_j|\leqslant d(v_7,v_{10})$.

证明 在 $T(v_7)$ 中,必存在一条路径 $P(j,v7)=P(j,v_{10})+P(v_{10},v_7)$. 而在 $T'(v_7)$ 中,必有 $d'_j=d(j,v_7)\leqslant d(j,v_{10})+d(v_{10},v_7)$,故 $d'_j-d_j\leqslant d(v_{10},v_7)$. 因 $d(v_{10},v_7)=d(v_7,v_{10})$,所以 $|d'_j-d_j|\leqslant d(v_7,v_{10})$.

由上可知,当将 $T(v_{10})$ 转换至 $T'(v_7)$ 时,若更改 v_{10} 的前驱节点为 v_7,并将 $T(v_{10})-T'(v_7)$ 内任意节点 i 的距离增加 $d(v_7,v_{10})$,即 $d_i=d_i+d(v_7,v_{10})$,因 $d_i-d'_i\leqslant 2\times d(v_7,v_{10})$,因此 i 的最短距离变化量在后续计算中不超过 $2\times d(v_7,v_{10})$. 此外,若 j 的最短距离减少 Δ,$T(v_j)$ 中任意点的最短距离至少减少 Δ.

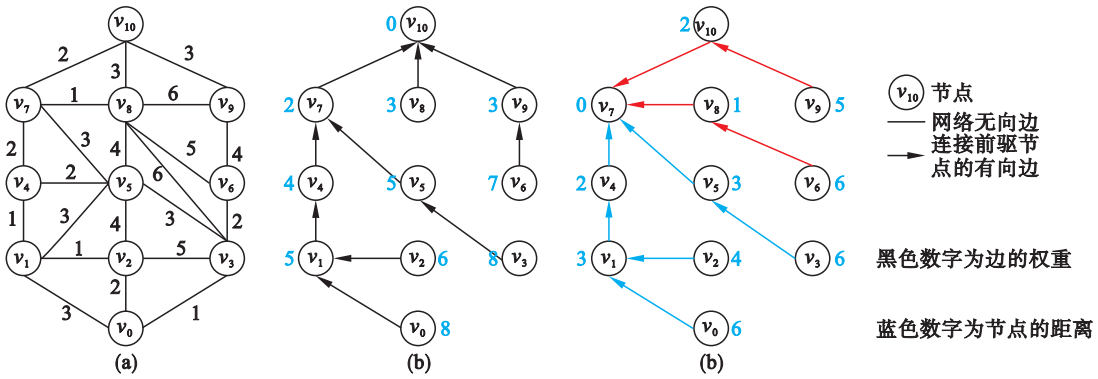

图 1　异源最短路径树实例

Fig. 1　An example of the shortest path tree with different sources

(a)—原始网络；(b)—最短路径树 $T(v_{10})$；(c)—最短路径树 $T(v_7)$.

图 2 描述了交通网络的一棵最短路径树及异源最短路径树间的差异. 图 2a 为原始底层交通网络，共包含 1 725 962 个节点，2 864 827 条边；在图 2b 的最短路径树中，每个节点（除目的节点 A 外）都有且仅有一条边连接前驱节点，故共有 1 725 960 条边；而在图 2c 中，两棵异源（即 $T(A)$ 和 $T(B)$）最短路径树间的差异很小，仅有 1 428 个节点的前驱节点发生变化. 因此，若仅计算前驱节点发生变化的节点，则可大幅度降低计算复杂度.

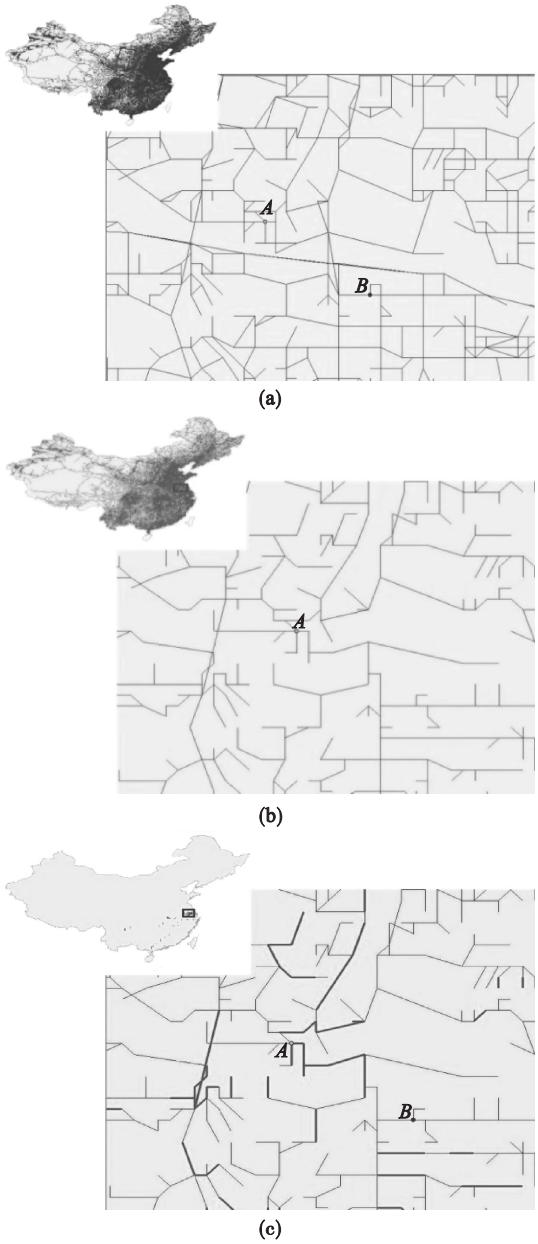

图 2　交通网络下的异源最短路径树

Fig. 2　Shortest path tree for real transportation network

(a)—原始底层网络；(b)—以 A 为目的节点的最短路径树；(c)—A 源和 B 源最短路径树差异.

3　重优化算法

3.1　异源最短路径树的转换

在实现异源最短路径树间的高效转换之前，需通过 Dijsktra 等算法求解一棵单源最短路径树 $T(t_1)$，再执行以下步骤将 $T(t_1)$ 转换成 $T(t_2)$.

1）修改路径 $P(t_2,t_1)$ 上所有节点的前驱节点，即若 $i\in P(t_2,t_1)$，$i\neq t_1$，令 $j=p_i$，则 $p_j=i$；

2）修改路径 $P(t_1,t_2)$ 上所有节点的距离，即若 $i\in P(t_1,t_2)$，令 $\Delta=2\times(d_{t2}-d_i)$，则 $d_i=\Delta+d_i$，对任意节点 $k\in T(i)$，设置 $d_k=\Delta+d_k$；

3）将节点集合 V 分解成 S，T 和 T_1，初始 $S=\{T(t_2)\}$，$T=\varnothing$，$T_1=V-T-S$；将路径 $P(t_1,t_2)$ 上的节点依次插入队列 Q' 队首；

4）取出 Q' 队首节点 q，令 $T=\{T(q)\}$，$T_1=V-T-S$；将 S 和 T 间所有代价差 $c_{ef}<0$ 的边插入到集合 Q 中，其中 $c_{ef}=w_{ef}+d_f-d_e$，$(e,f)\in E$，$e\in T$，$f\in S$.

5）从 Q 中选择代价差最小的边 (u,v)，并从 Q 中剔除 (u,v)，其中 $c_{ef}<0$，设置 $p_u=v$，$S=S\cup T(u)$，$T=T-T(u)$，对任意 $i\in T(u)$，$d_i=d_i+c_{uv}$；

6）将 $T(u)$ 与 T 间所有代价差 $c_{ef}<0$ 的边插入 Q 中，即 $(e,f)\in E$，$e\in T$，$f\in T(u)$；

7) 重复步骤 5)和步骤 6),直至 Q 为空;

8) $S=S+T$,若 Q'非空,则转至步骤 4),否则结束.

3.2 RASP 算法

RASP 算法利用上述转换方法,从已知最短路径树开始,依次求解网络所有节点的最短路径树,从而获得全源最短路径. 主要包括以下步骤:

1) 初始化所有节点为未访问节点,随机选择节点 t_1 作为目的节点,令 $t=t_1$,利用 Dijsktra 算法求解单源最短路径及 $T(t)$,标记 t 为已访问节点;

2) 使用搜索算法(如广度优先搜索算法)从根节点开始搜索下一个未访问节点 t',利用异源最短路径树转换方法将 $T(t)$ 转换成 $T(t')$,标记 t'为已访问节点,并令 $t=t'$;

3) 重复步骤 2)至所有节点都为已访问节点.

由第 2 节分析可知,t 与 t'间的最短距离直接影响异源最短路径树的转换效率,即距离越近,效率越高,因此 RASP 算法优先选择距离 t 较近的未访问节点 t';此外,集合 $T+T_1$ 包含的节点越少,则搜索空间越小,转换效率越高,故 RASP 算法优先选择具有较少子孙节点的 t'.

4 分 析

4.1 正确性

由 Dijsktra 算法计算得到的初始单源最短路径树是精确解,故本文只需验证异源最短路径树转换方法的正确性即可确保 RASP 算法的正确性. 证明如下:

1) 在 $T(t_1)$ 转换成 $T(t_2)$ 之前,在网络中添加虚拟节点 t_1 及虚拟边(t_2,t_0),其中 $w(t_2,t_0)=d(t_2,t_1)$. 当更新路径 $P(t_2,t_1)$ 和距离,即完成步骤 1)和步骤 2)后,任意节点都存在唯一路径至 t_0,且其当前记录距离即为该路径长度. 后续步骤的目的是调整所有节点的当前路径为到达 t_0 的最短路径.

2) 步骤 3)将节点集合 V 分解成 S,T 和 T_1 三部分. 其中 S 为已获得最短路径的节点集合,T 为待优化的节点集合,而 T_1 则是未优化的节点集合. 初始时,$S=\{T(t_2)\}$,第 2 节已证明 S 的结构不会发生变化,即 S 内所有节点的当前路径即为最短路径;在步骤 5) ~ 步骤 7)中,若 T 能将其内节点的路径优化成最短路径,则 $S=S+T$ 内的节点都已获得最优解;经过步骤 4) ~ 步骤 8)后,$S=V$. 以下证明集合 T 内的节点执行步骤 5) ~ 步骤 7)后获得最短路径.

3) 在当前生成树中,边(u,v)的代价差 $c_{uv}=|P(v,t_0)|+w(u,v)-|P(u,t_0)|$,$c_{uv}<0$ 表示路径 $P(u,v)+P(v,t_0)$ 比当前路径 $P(u,t_0)$ 更短,需调整以获得更短路径. 若网络中所有边的代价差都为非负值,则所有节点都已获得最短路径. 对于边(u,v),$u\in T$,在转换之前满足 $c_{uv}=d_v+w_{uv}-d_u\geqslant 0$. 在最短路径树转换过程时执行步骤 1)和步骤 2),则 $d_u=d_u+\Delta_1$,$d_v=d_v+\Delta_2$. 若 $v\in T$,则 $\Delta_1=\Delta_2$,$c_{uv}\geqslant 0$;若 $v\in T_1$,则 $\Delta_1<\Delta_2$,$c_{uv}\geqslant 0$;若 $v\in S$,则 $\Delta_1>\Delta_2$,c_{uv}可能小于零. 因此,若 $c_{uv}<0$,则必有 $v\in S$. 令 $\Delta_3=c_{uv}<0$ 为最小的代价差,则对于任意边(i,j)($u\in T(u)$,$j\in S$)的代价差 $c_{ij}=c_{ij}-\Delta_3>0$,故 $T(u)$ 中不存在代价差为负值的边,且其上所有点的最短距离都减少$|\Delta_3|$;在后续计算中,集合 $T-T(u)$ 中的节点距离减少量$|\Delta_4|<|\Delta_3|$,故也不存在 $c_{ij}<0$ 的边(i,j)($i\in T(u)$,$j\in T-T(u)$). 综上,执行步骤 5) ~ 步骤 7)后,$S=S+T$,$T=\varnothing$,S 内部不存在代价差为负值的边,即集合 T 经过步骤 5) ~ 步骤 7)后所有节点都获得最短路径.

4.2 时间复杂度

异源最短路径树转换方法主要经历以下过程:1)更改路径 $P(t_2,t_1)$ 上节点的前驱节点和距离(步骤 1)和步骤 2)),遍历整条路径,其时间复杂度为 $O(n_1)$,其中 n_1 为 $P(t_2,t_1)$ 的长度;2)步骤4) ~步骤 7)首先搜索 S 与 T 间所有的边,并计算其代价差,其时间复杂度为 $O(m_1)$,其中 m_1 为 S 与 T 相连边的数量;然后步骤 6)搜索 $T(v)$ 与 T 间所有的边,时间复杂度为 $O(m_2)$,其中 m_2 为 S 与 $T(u)$ 相连边的数量,因 $T(u)$ 的总量为 T,故 $O(\sum m_2)=O(m_1)$;3)在步骤 3) ~步骤 8)中,T 的目的是将集合 $V-S$ 转移到 S,即 $\sum T=V-S$. 因此,异源最短路径树转换方法的时间复杂度为 $O(n_1+2\times\sum m_1)\leqslant O(n+2m)$.

RASP 算法首先执行一次 Dijkstra 算法以获得初始最短路径树,其时间复杂度为 $O(n^2)$;然后共执行 n 次异源最短路径树转换方法得到全源最短路径,其中利用广度(或深度)优先搜索获得下一目的节点,因此,异源最短路径树转换的时间复杂度为 $O(n\times(n+(n+2m)))=O(2(n^2+nm))$. 综上所述,RASP 算法的时间复杂度为 $O(n^2+2(n^2+nm))=O(3n^2+2nm)$.

5 实验与分析

为测试 RASP 算法的正确性和时间效率,本

实验基于 Window 7 操作系统，采用 Visual C + + 编程，在 CPU 为 Intel(R) Xeon(R) E5 - 265 @ 2.00 GHz，内存为 48 GB 的图形工作站上，分别利用现实交通网络和人造网络对本算法进行测试，并与以下 4 种算法进行对比. ①n - Dijkstra - priority(NDP)算法：Dijkstra 算法[10]是经典的求解单源最短路径算法，根据采用的数据结构不同，可有多种实现方式，NDP 算法采用优先队列，并执行 n 次以获得全源最短路径；②n - Dijkstra - binary(NDB)算法[14]：不同于 NDP 算法，NDB 算法采用二叉树结构，并使用二分排序法对候选节点进行排序，每次迭代中选择标记最小的节点；③Floyd 算法：Floyd 算法[9]是经典的求解全源最短路径算法，在稠密网络中能取得较好的计算效率；④improved - Floyd(I - Floyd)算法：改进的 Floyd 算法在时间效率上具有一定的优势.

在测试实验中，本文 RASP 算法获得网络中任意点对间的最短距离，与 4 种对比算法计算所得的最短路径值相同，验证 RASP 算法的正确性. 在时间效率方面，测试结果如下.

5.1　交通网络

测试数据集为全国道路网，共含多个级别的道路，根据包含道路级别数的不同，将道路网分成 5 个大小不同的测试网络：①网络 1 包含 689 155 个节点，1 160 466 条边；②网络 2 包含 411 719 个节点，946 974 条边；③网络 3 包含 268 326 个节点，349 008 条边；④网络 4 包含 115 122 个节点，149 360 条边；⑤网络 5 包含 5 983 个节点，7 468 条边. 此处定义平均度为边与点的数量之比. 表 1 为最短路径算法的时间效率对比.

表 1　最短路径算法时间效率对比

Table 1　Time performance comparison of shortest path algorithms　　s

算法	网络 1	网络 2	网络 3	网络 4	网络 5
NDP	75 594	27 623	10 759	1 829	0.624
NDB	-	98 139	38 303	5 413	0.813
Floyd	×	×	×	×	312
I - Floyd	×	×	×	×	73
RASP	49 339	16 362	5 032	593	0.091 2

对比表 1(其中“×”表示超出内存，无法获得计算时间；“-”表示计算时间过长)中 5 种算法的测试结果，可得出以下结论：

1) RASP 算法的运行时间仅为 NDP 的 32% ~65%，为 NDB 算法的 12% 左右. 其原因在于 RASP 算法仅搜索最短路径可能发生变化的节点，即集合 $T + T_1$，故相比于 NDP 和 NDB 算法，大大减少了搜索空间，提升了计算速度. 可见 RASP 算法在大规模交通网络上，具有一定的时间优势.

2) 与 Floyd 和 I - Floyd 算法相比，RASP，NDP 和 NDB 算法在空间复杂度上具有绝对优势，因为 RASP，NDP 和 NDB 算法使用邻接链表(空间复杂度为 $O(n+m)$)，而 Floyd 和 I - Floyd 算法使用邻接矩阵(空间复杂度为 $O(n^2)$)，由于网络 1 ~ 网络 4 的数据量较大，Floyd 和 I - Floyd 算法无法申请到足够的连续内存，导致其无法运行. 在时间效率上，RASP，NDP 和 NDB 算法也优于 Floyd 和 I - Floyd 算法. 本例测试的交通网络是稀疏网络(平均度为 1.7)，Floyd 和 I - Floyd 算法效率低下.

综上所述，在交通网络上，RASP 算法无论是时间复杂度还是空间复杂度都优于 Floyd 和 I - Floyd 算法. 对比于 NDP 和 NDB 算法，RASP 算法的运行速度是其运行时间的 1.5 倍之余.

5.2　人工网络

为进一步测试网络特征对算法的影响，本实验采用由 the first DIMACS workshop[15] 提供的 netgen 网络生成器，构建节点数为 10 000，平均度分别为 8，16，32，64，128 和 256 的虚拟网络.

由图 3 测试结果可知：①Floyd 算法受网络平均度的影响很小，运行时间保持在 1 750 s 左右；RASP 算法的计算时间随着平均度的增加而呈现递增的趋势，但总体效率明显优于 Floyd 算法；②NDP 算法随着平均度的增加，其运行时间增长速度比 RASP 算法快，效率不如 RASP 算法；③NDB 和 I - Floyd 算法在本例测试中，不具备效率优势；④RASP 算法较其他所有对比算法始终保持最佳的时间效率.

由此可见，无论是稀疏网络还是稠密网络，RASP 算法的时间效率都具有明显的优势.

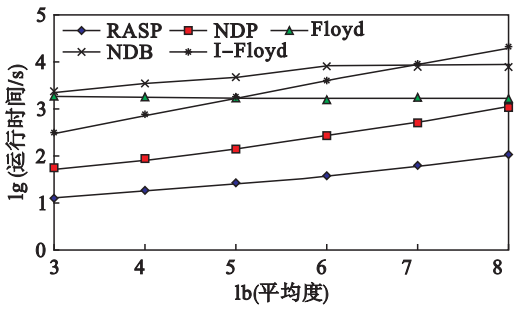

图 3　算法在不同平均度网络中的时间效率对比

Fig. 3　Time performance comparison in networks with varying degree

6 结 论

本文基于重优化理论实现了异源最短路径树间的高效转换,并在此基础上提出了一种适用于大规模网络的全源精确最短路径求解方法——RASP 算法. 理论证明该算法的时间复杂度为$O(3n^2+2nm)$. 实验结果表明:在稀疏的交通网络上,RASP 算法的运行时间约为 NDP 算法的 32% ~65% ,为 NDB 算法的 12% 左右,明显优于 Floyd 和 I - Floyd 算法;在平均度变化的人造网络上,RASP 算法的运行效率高于 NDP,NDB,Floyd 和 I - Floyd 算法. 因 $n<m<n^2$,RASP 算法不仅理论时间复杂度在稀疏网络上低,而且在实际测试中,RASP 算法无论是在稀疏还是稠密网络上,都能取得较好的时间效率,为快速求解大规模网络的全源最短路径提供了新方法.

参考文献:

[1] Parichart P, Dongjoo P, Laurence R R, et al. Dynamic and stochastic shortest path in transportation networks with two components of travel time uncertainty [J]. *Transportation Research Part C: Emerging Technologies*, 2003, 11(5): 331 - 354.

[2] 徐业昌,李树祥,朱建民,等. 基于地理信息系统的最短路径搜索算法[J]. 中国图象图形学报,1998,3(1):39 - 43.
(Xu Ye-chang, Li Shu-xiang, Zhu Jian-min, et al. A improved best-first search algorithm based on geographical information system [J]. *Journal of Image and Graphics*, 1998, 3(1): 39 - 43.)

[3] 严寒冰,刘迎春. 基于 GIS 的城市道路网最短路径算法探讨[J]. 计算机学报,2000,23(2):210 - 215.
(Yan Han-bing, Liu Ying-chun. A new algorithm for finding shortcut in a city's road net based on GIS technology [J]. *Chinese Journal of Computers*, 2000, 23(2): 210 - 215.)

[4] 吴京,景宁,陈宏盛. 最佳路径的层次编码及查询算法[J]. 计算机学报,2000,23(2):184 - 189.
(Wu Jing, Jing Ning, Chen Hong-sheng. Hierarchical encoding of optimal path and its retrieval [J]. *Chinese Journal of Computers*, 2000, 23(2): 184 - 189.)

[5] Ahmed Y H. A genetic algorithm for finding the k shortest paths in a network [J]. *Egyptian Informatics Journal*, 2010, 11(2): 75 - 79.

[6] Cherkassky B V, Goldberg A V, Radzikt T. Shortest paths algorithms: theory and experimental evaluation [J]. *Mathematical Programming*, 1996, 73: 129 - 174.

[7] 陆锋. 最短路径算法:分类体系与研究进展[J]. 测绘学报,2001,30(3):269 - 275.
(Lu Feng. Shortest path algorithms: taxonomy and advance in research[J]. *Acta Geodaetica et Cartographica Sinica*, 2001, 30(3): 269 - 275.)

[8] 李鸣鹏,邹兆年,高宏,等. 不确定图上期望最短距离的计算[J]. 计算机研究与发展,2012,49(10):2208 - 2220.
(Li Ming-peng, Zou Zhao-nian, Gao Hong, et al. Computing expected shortest distance in uncertain graphs[J]. *Journal of Computer Research and Development*, 2012, 49(10): 2208 - 2220.)

[9] Floyd R W. Algorithm 97: shortest path[J]. *Communications of ACM*, 1962, 5(6): 345.

[10] Dijkstra E W. A note on two problems in connection with graphs [J]. *Numerische Mathematik*, 1959, 1(1): 269 - 271.

[11] 刑星星,赵国兴,骆祖莹,等. 基于 GPU 的全源最短路径算法[J]. 计算机科学,2012,39(3):299 - 303.
(Xing Xing-xing, Zhao Guo-xing, Luo Zu-ying, et al. GPU-based algorithm of shortest path [J]. *Computer Science*, 2012, 39(3): 299 - 303.)

[12] Miller-Hooks E, Yang B. Updating paths in time-varying networks given arc weight changes [J]. *Transportation Science*, 2005, 39(4): 451 - 464.

[13] Pallottino S, Scutelli M. Shortest path algorithms in transportation models: classical and innovative aspects [C]// Marcotte P, Nguyen S. Equilibrium and Advanced Transportation Modeling. Boston: Kluwer Academic Publishers, 1998: 245 - 281.

[14] Xu M H, Liu Y Q, Huang Q L, et al. An improved Dijkstra's shortest path algorithm for sparse network [J]. *Applied Mathematics and Computation*, 2007, 185(1): 247 - 254.

[15] Johnson D S, McGeoch C C. Network flows and matching: first DIMACS implementation challenge [M]. Provindence: American Mathematical Society, 1993.