Computer Science and Application 计算机科学与应用, 2018, 8(10), 1510-1517
Published Online October 2018 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2018.810164

Hans汉斯

Network Maximum Flow Analysis Base on Dinic's Algorithm

Gang Li, Jinbao Miao, Chun'an Hu

Information Engineering, Jiangxi University of Science and Technology, Ganzhou Jiangxi
Email: 1173298382@qq.com

Received: Oct. 2nd, 2018; accepted: Oct. 15th, 2018; published: Oct. 22nd, 2018

Abstract

This paper mainly analyzes the problem of calculating the maximum flow of the network by the Dinic algorithm. First we review the basic concepts of residual networks and hierarchical networks, then analyze the specific process of calculating the maximum flow of the network by the Dinic algorithm. By comparing with the Ford-Fulkerson algorithm and the Edmonds-Karp algorithm, the highlight of the Dinic algorithm is demonstrated. Through the comparison of correlations, the conclusion is that the Dinic algorithm works better than Ford-Fulkerson and is better than Edmonds-Karp.

Keywords

Dinic, Network Flow, Maximum Flow, Augmenting Chain

基于连续最短增广链的网络最大流分析

李　港，苗金宝，胡春安

江西理工大学，信息工程学院，江西　赣州
Email: 1173298382@qq.com

收稿日期：2018年10月2日；录用日期：2018年10月15日；发布日期：2018年10月22日

摘　要

本文主要是分析连续最短增广链算法计算网络最大流的问题。先综述残留网络和层次网络的基本概念，然后分析连续最短增广链算法计算网络最大流的具体过程，再通过与Ford-Fulkerson (福特-富尔克森算法)和Edmonds-Karp (埃德蒙兹-卡普算法)算法进行比较来体现出连续最短增广链算法的突出点。通过相关性的比较，结论是连续最短增广链算法运行效果明显比Ford-Fulkerson好，且优于Edmonds-Karp。

文章引用: 李港, 苗金宝, 胡春安. 基于连续最短增广链的网络最大流分析[J]. 计算机科学与应用, 2018, 8(10): 1510-1517. DOI: 10.12677/csa.2018.810164

关键词

连续最短增广链，网络流，最大流，增广链

Copyright © 2018 by authors and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

网络流中最大流问题是一个经典的问题，在最初的 Ford-Fulkerson 算法提出到现在已有 60 多年的历史了[1]，一直是个值得研究的问题。在此过程中，也提出了许多与之相关的算法。相比于初期的算法，现在对于网络最大流的算法得到了很大的改进，算法时间和空间复杂度都有所下降。常用的算法为 Ford-Fulkerson 算法[2]、Edmonds-Karp 算法[3]和 Dinic 算法[4]等。

Ford-Fulkerson 算法是利用深度优先搜索的思想来寻找增广链，而这样寻找会使得复杂度依赖于最大传输量。Edmonds-Karp 算法则在 Ford-Fulkerson 算法的基础上进行了修改，使得每次按最短路径寻找增广链，但每次找完一个最短增广链后需要重新寻找，利用率不高。而 Dinic 算法则是效率更高，使用更频繁。

为此，本文对连续最短增广链算法在网络最大流问题上做一个详细的分析。该算法虽然也是按最短路径来寻找增广链的，不过增加了一个层次网络。相比于每次重新寻找最短增广链来说，利用层次网络将避免了重新寻找最短的增广链所带来的多余的步骤。

2. 基本概念

2.1. 基本定义

定义 1.1：有向图为 D = (V, E)。V 为节点集合，E 为边集合，D 为由 E 和 V 构成的有向图。

定义 1.2：对于任意一条边 e = <u, v> {e ∈ E, v, u ∈ V}，将会有值 c(e)，表示该边的最大容量。

定义 1.3：f(e)为该边 e 现有的传输量，对于任意的边 e ∈ E，它将有 $0 \le f(e) \le c(e)$。

定义 1.4：容量网络中会有一个源点 s 和一个汇点 t。对于给定的容量网络，将存在源点 s 和汇点 t，分别表示网络图中的起点和终点。其它为中间顶点，源点 s 需要通过中间顶点传输到汇点。

定义 1.5：容量网络为 G = (V, E, C)。见图 1，对于任意边 e，将有(c(e), f(e))，其中 c(e)为最大容量，f(e)为实际传输量。

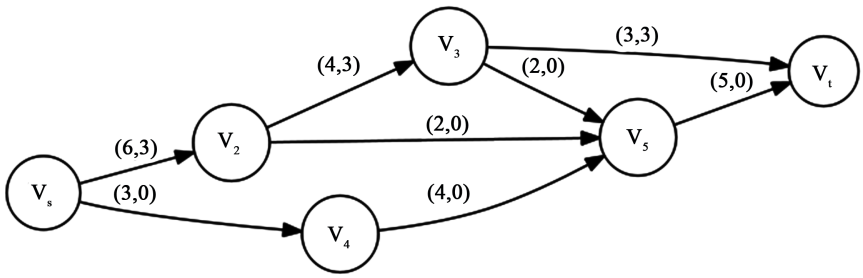

Figure 1. Capacity-network
图 1. 容量网络

2.2. 预备知识

学会连续最短增广链算法首先需要了解的是残留容量网络。设有容量网络为 G = (V, E, C)，那么残留容量网络为 G'(V', E', C') [5]，其中 G'的顶点集不变，即 V' = V，但对于 E'来说，就发生了改变。如果 G 中存在 e = <u, v> {e ∈ E, u, v ∈ V}，满足 $c(e) > f(e)$，那么 G'将有边 e' = <u, v> {e' ∈ E', u, v ∈ V'}，满足 $c(e') = c(e) - f(e)$。如果 G 存在 e = <u, v> {e ∈ E'，u, v ∈ V'}，满足 $f(e) > 0$，那么 G'将有 e' = <v, u> {e' ∈ E'，u, v ∈ V'}，满足 $c(e') = f(e)$。见图 2，边的值对应 $c(e')$。

了解残留网络还是不够的，还需要理解层次网络[6]。见图 3，在层次网络中，源点 s 的层次为 0，在残留网络中将存在一顶点 v ∈ V'，那么源点 s 到达 v 的最短路径长度就是 v 的层次。从源点 s 开始，以广度优先搜索的方法构建层次网络，使的能找到一条最短的增广链。如果无法构建层次网络，就说明网络最大流的求解已经得知。

3. 连续最短增广链算法

3.1. 算法思路

连续最短增广链算法是先建立层次网络，然后基于层次网络找到最短的增广链，以达到最优解，并且在层次网络上每使用一次增广链后还可以重复使用，直到最短的增广链不存在。如果找不到最短的增广链就利用剩下的残留容量网络继续构建层次网络，并重复这些步骤。当层次网络无法构建时，那么网络最大流问题也就结束了。

已知有一容量网络为 G = (V, E, C)，顶点数为 n，最大容量为 C，实际传输量为 f，层次数 l，源点为 s，汇点为 t。

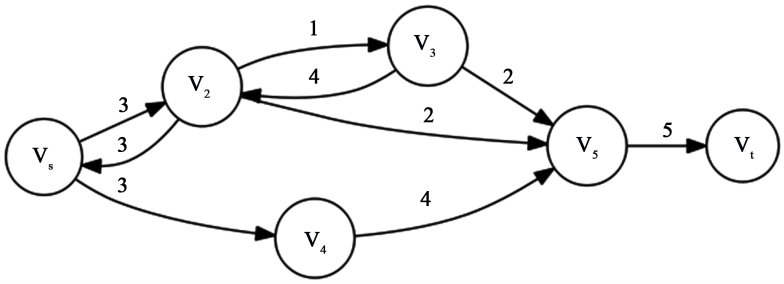

Figure 2. Residual network
图 2. 残留网络

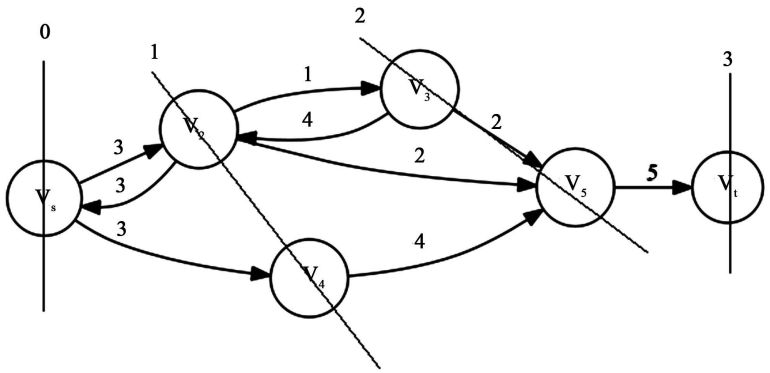

Figure 3. Hierarchical network
图 3. 层次网络

步骤①：初始化容量网络 G 和可行流 f，使的 f = {0}。

步骤②：构建层次网络，设集合 P 为已经设置好层次的顶点集合，Q 为已经从集合 P 中取出来使用的顶点集合，将源点 s 的层次设为 0，即 level(s) = 0，并将 s 加入集合 P 中，使的 P = {s}。接着从 P 中取出一顶点 v($v \in P$，$v \notin Q$)，对于任意边 e = <v, u> {$u \notin P$}且 c(e) > 0，有 level(u) = level(v) + 1。这样就能计算 v 的所有邻居节点的层次，并将 v 加入集合 Q，与 v 相连的 u 加入 P。重复以上步骤，使的无法从 P 中取出元素即层次网络构建完成。

步骤③：利用层次网络寻找最短的增广链，而层次网络可以重复使用，并且寻找最短的增广链需要以源点 s 为出发点，以汇点 t 为终点。一开始以层次为 0 的点开始，即源点 s，由于对于任意边 e = <s, u>且 c(e) > 0 有 level(u) = level(s) + 1，所以寻找 s 的下一顶点时就可以根据层次网络快速求得 u。同理，对于任意边 e = <u, v>且 c(e) > 0 有 level(v) = level(u) + 1，这样就可以得到 u 的下一顶点。由于步骤②层次网络构建完成，那么汇点 t 一定有层次，继续寻找下一顶点，直到汇点 t 就说明最短的增广链成功的找到。然后在增广链的基础上更新残留容量网络，接着继续寻找最短的增广链，直到找不到为止。

步骤④：若能构建层次网络，跳到步骤②，如若不能则网络最大流问题求解完毕。

3.2. 计算过程

下面将对一个具体的网络流进行分析，为了更好的理解连续最短增广链算法，模拟连续最短增广链算法计算网络最大流的过程。以图 4 为例进行分析。

图 4 中弧的数字表示最大容量，初始网络流值为 0。设 V_1 位源点，V_6 为汇点来计算网络图中的最大网络流。首先要构建层次网络，将源点 V_1 的层次设为 0，根据步骤②的思路来计算每个节点的层次数，得到的层次数如表 1 所示。

得到了层次网络后，就用它来寻找最短增广链。用层次网络能更快的查找出最短增广链，将得到两条最短增广链，分别是 $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_6$ 和 $V_1 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6$。接着更新网络图，以便得到最新的残留网络。更新后的残留网络如图 5 所示，此时最大网络流为 9。

图 5 中弧的两个数字分别为最大容量和实际传输量。在增广后明显可以看出 $V_2 \rightarrow V_3$ 这条边已经满了，无法再传输了，但根据增广链的思想，从 $V_3 \rightarrow V_2$ 还能传输 6。现在增广后的网络图上再重新构建层次网络，构建后的层次数如表 2 所示。

INF 表示无穷大，代表着无法从源点 V_1 达到此点。此时可以得出最短增广链只有一条 $V_1 \rightarrow V_2 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6$，继续在层次网络的基础上更新残留网络。得到的新的残留网络如图 6 所示。

由于此时不能再构建层次网络了，所以结束该算法，得到的网络最大流为 11。连续最短增广链算法主要是利用构建好的层次网络能更快的查找最短增广链。

Table 1. The hierarchy number of the original network planning
表 1. 原网络图中的层次数

节点	V_1	V_2	V_3	V_4	V_5	V_6
层次	0	1	2	1	2	3

Table 2. The hierarchy number of the original network planning after first update
表 2. 第一次更新的残留网络中的层次数

节点	V_1	V_2	V_3	V_4	V_5	V_6
层次	0	1	INF	2	3	4

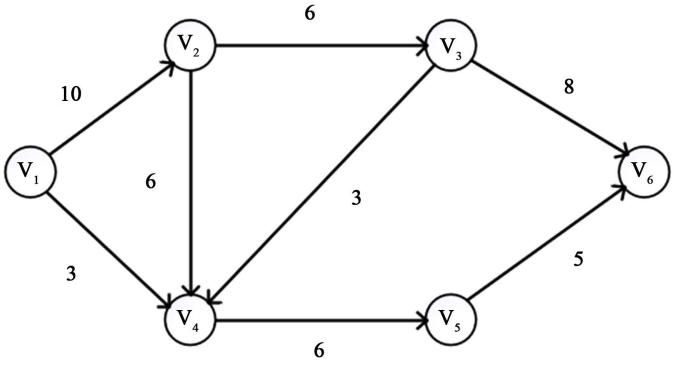

Figure 4. The original network planning
图 4. 原网络图

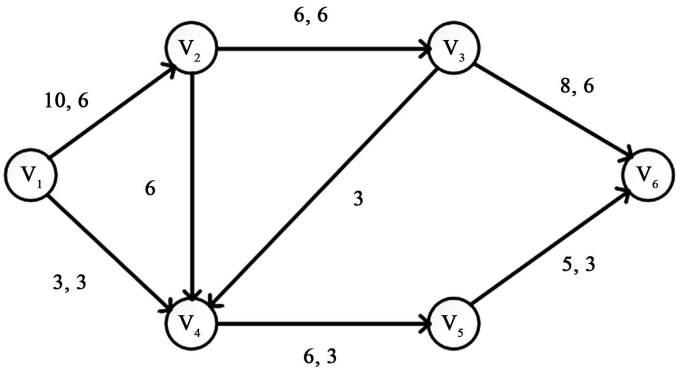

Figure 5. The residual network after first update
图 5. 第一次更新的残留网络

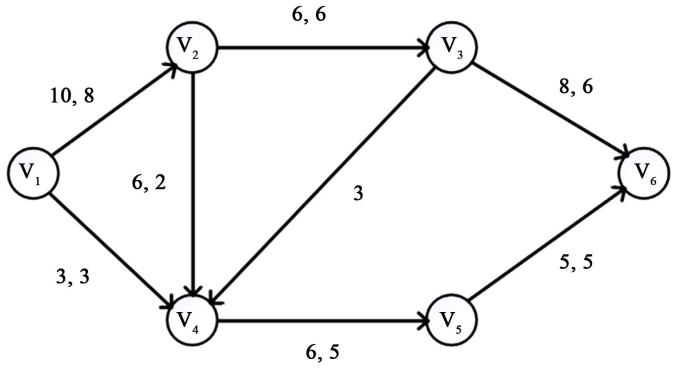

Figure 6. The residual network after second update
图 6. 第二次更新的残留网络

3.3. 算法复杂度分析

连续最短增广链算法需要重复运行步骤②和步骤③，构建层次网络以及查找最短增广链。已知容量网络 G 有 V 个顶点，E 条边。在连续最短增广链算法中最多构建 V 分层次网络。

首先分析步骤②构建层次网络，层次网络是从源点开始，将每个节点与源点的距离算出来以确定节点的层次数。而这过程是用 BFS 思想[7]来实现的，这就需要遍历每一条边。即构建一个层次网络的时间复杂度是 O(E)，由于需要构建 V 个层次网络，所以构建层次网络总的时间复杂度是 O(VE)。

步骤③是在层次网络的基础上寻找最短增广链。由于有 E 条边，那么最多将存在 E 条增广链。而对于每一条增广链，都是在层次网络的基础上寻找得来的。而层次数是与源点的距离，那么层次数最大是 V，即最短增广链的长度最大为 V。所以在寻找增广链上时间复杂度是 O(VE)。由于最多构建 V 个层次网络，而在每个层次网络上寻找最短增广链的时间复杂度是 O(VE)，那么寻找增广链总的时间复杂度是 $O(EV^2)$。

综上所述，连续最短增广链的时间复杂度是 $O(EV + EV^2)$，即 $O(EV^2)$。

4. 算法比较

1) Ford-Fulkerson 算法，该算法是每次都寻找一条从源点 s 到汇点 t 的路径[8]。然后更新残留网络，直到找不到为止。由于该算法是每找到一条路径后更新残留网络，而残留网络每一条边的最大传输量是根据上一个残留网络得来。已知容量网络的最大传输量是 F，那么该算法最坏情况下需要进行 F 次深度优先搜索，那么总的时间复杂度是 O(FE)，那么当 F 是很高时明显不是一种好方法。

2) Edmonds-Karp 算法，该算法是每次寻找一条最短的增广链来进行更新[9]。由于每次查找一条最短的增广链就更新一次，而最坏情况下会有 E 条增广链。每次更新残留网络的话需要 O(VE)的复杂度，那么总的时间复杂度是 $O(EV^2)$，在存在增广链的条件下，E 的最少数量是 V − 1，当然实际情况是更多的，即 E > V，对 E 进行平方明显不是最优的情况。

3) 连续最短增广链算法，该算法是分为 V 个阶段，每个阶段构建层次网络和查找最短增广链。先建立层次网络，在层次网络的基础上去找最短的增广链，而时间复杂度是 $O(EV^2)$。

一般情况下 V < E，那么 $O(EV^2)$的时间复杂度优于 $O(EV^2)$，而且不需要考虑最大容量上限的问题。不难看出，连续最短增广链算法运行效果明显比 Ford-Fulkerson 好，且优于 Edmonds Karp。

5. 案例比较

例 1：针对图 7 的图型模型，利用连续增广链算法求其最大流。

1) Ford-Fulkerson 算法：在最佳情况下需要增广 2 次，最坏情况下需要增广 4036 次。该算法就是简单用 DFS 思想去寻找一条增广链，最坏情况下会一直是 $V_s \to V_2 \to V_5 \to V_t$ 和 $V_s \to V_4 \to V_5 \to V_2 \to V_3 \to V_t$ 这两条。这就需要增广 4036 次。当然如果正好寻找增广链是寻找到的是 $V_s \to V_2 \to V_3 \to V_t$ 和 $V_s \to V_4 \to V_5 \to V_t$ 这两条，就只需要增广 2 次，这是最好的情况。

2) Edmonds-Karp 算法：在最佳情况下需要增广 2 次，最坏情况下需要增广 3 次。该算法相比 Ford-Fulkerson 算法，有一个优势，每次增广是按 BFS 思想寻找最短增广链的。所以在最坏情况下寻找到的增广链依次是 $V_s \to V_2 \to V_5 \to V_t$、$V_s \to V_4 \to V_5 \to V_t$ 和 $V_s \to V_2 \to V_3 \to V_t$ 这三条。在最佳情况下的寻找到的增广链是 $V_s \to V_2 \to V_3 \to V_t$ 和 $V_s \to V_4 \to V_5 \to V_t$ 这两条。按最短路径寻找增广链相比于 DFS 毫无目的的寻找快的多。

3) Dinic 算法：在最佳情况下需增广 2 次，最坏情况下需增广 3 次。该算法首先构建层次网络，在层次网络的基础上寻找最短增广链。最坏情况下寻找到的增广链分别 $V_s \to V_2 \to V_5 \to V_t$、$V_s \to V_4 \to V_5 \to V_t$ 和 $V_s \to V_2 \to V_3 \to V_t$ 这三条。在最佳情况下寻找到的增广链是 $V_s \to V_2 \to V_3 \to V_t$ 和 $V_s \to V_4 \to V_5 \to V_t$ 这两条。相比于 Edmonds-Karp 算法每次寻找增广链都用 BFS 查找一遍来说，在层次网络的基础上寻找最短增广链将快的多。

例 2：针对图 8 的图型模型，利用连续增广链算法求其最大流。

1) Ford-Fulkerson 算法：在最佳情况下，需要增广 2 次，最坏情况需要增广 4036 次。用 DFS 来寻找增广链最坏情况下，增广链将会一直是 $V_s \to V_6 \to V_5 \to V_t$ 和 $V_s \to V_2 \to V_4 \to V_5 \to V_6 \to V_t$

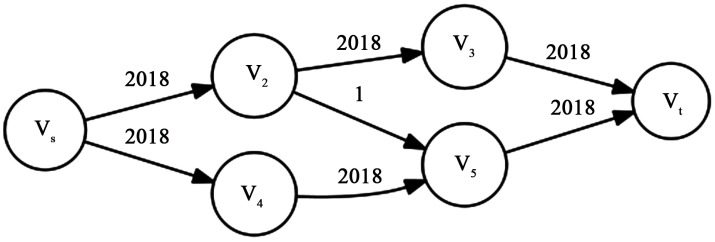

Figure 7. Example 1
图 7. 案例 1

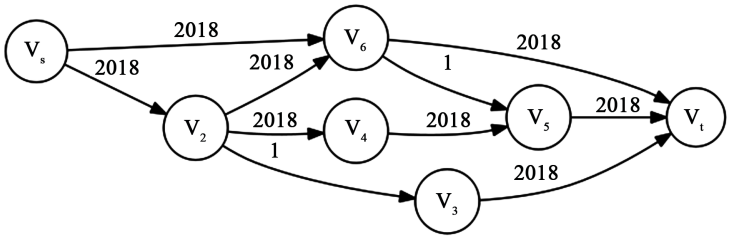

Figure 8. Example 2
图 8. 案例 2

这两条。这就需要增广 4036 次。也有可能正好寻找到 $V_s \rightarrow V_6 \rightarrow V_t$ 和 $V_s \rightarrow V_2 \rightarrow V_4 \rightarrow V_5 \rightarrow V_t$ 这两条，这是最佳情况下，需要增广 2 次。

2) Edmonds-Karp 算法：在最佳情况下需要增广 3 次，最坏情况下需要增广 4 次。按最短距离寻找增广链，最坏情况下将依次找到 Vs → V6 → Vt、Vs → V2 → V3 → Vt、Vs → V2 → V6 → V5 → Vt 和 Vs → V2 → V4 → V5 → Vt 这四条，最佳情况下将寻找到 Vs → V6 → Vt、Vs → V2 → V3 → Vt 和 Vs → V2 → V4 → V5 → Vt 这三条。

3) Dinic 算法：在最佳情况需要增广 3 次，最坏情况下需增广 4 次。最坏情况下的增广链分别是 Vs → V6 → Vt、Vs → V2 → V3 → Vt、Vs → V2 → V6 → V5 → Vt 和 Vs → V2 → V4 → V5 → Vt 这四条。最佳情况下的增广链分别是 Vs → V6 → Vt、Vs → V2 → V3 → Vt 和 Vs → V2 → V4 → V5 → Vt 这三条。

整体来说连续最短增广链算法运行速度效果明显快于 Ford-Fulkerson，而连续最短增广链算法是建立一个层次网络，使得查找最短的增广链可重复使用。又因为 Edmonds-Karp 算法每找到一个最短的增广链就更新下。所以可以得出结论：连续最短增广链算法效果明显比 Ford-Fulkerson 好，且优于 Edmonds-Karp 算法。

6. 结束语

网络最大流问题从提出来现在经历了这么多年，为了解决这个问题，提出了很多算法，算法复杂度也逐渐降低。为此，本文对主要常用的连续最短增广链算法，做了一个详细而全面的分析。由于连续最短增广链算法用到了一个层次网络，使得每次能快速地查找到最短的增广链并能重复使用。相比 Edmonds-Karp 算法有了明显时间上的优化，而且也不要像 Ford-Fulkerson 算法那样需要考虑容量网络中的最大传输量。

基金项目

江西理工大学创新基金资助项目(No.3103800226)。

参考文献

[1] 张宪超, 陈国良, 万颖瑜. 网络最大流问题研究进展[J]. 计算机研究与发展, 2003, 40(9): 1281-1292.

[2] Ford, L.R.J. and Fulkerson, D.R. (1954) Maximal Flow through a Network. *Canadian Journal of Mathematics*, **8**, 399-404.

[3] Edmonds, J. and Karp, R.M. (1972) Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM*, **19**, 248-264. https://doi.org/10.1145/321694.321699

[4] Dinitz, E.A. (1970) Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Mathematics. Doklady*, 1277-1280.

[5] 谢政. 网络算法与复杂性理论[M]. 国防科技大学出版社, 2003.

[6] 徐翠霞. 基于层次网络的最大流求解方法[J]. 潍坊学院学报, 2010, 10(4): 42-45.

[7] Peleg, D. and Peleg, D. (2016) Sparse Fault-Tolerant BFS Structures. ACM.

[8] Takahashi, T. (2016) The Simplest and Smallest Network on Which the Ford-Fulkerson Maximum Flow Procedure May Fail to Terminate. *Technical Report of IeiceCst*, **24**, 390-394. https://doi.org/10.2197/ipsjjip.24.390

[9] Lammich, P. and Sefidgar, S.R. (2016) Formalizing the Edmonds-Karp Algorithm. Interactive Theorem Proving. Springer International Publishing, 219-234.

Hans汉斯

知网检索的两种方式：

1. 打开知网页面 http://kns.cnki.net/kns/brief/result.aspx?dbPrefix=WWJD
   下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询
2. 打开知网首页 http://cnki.net/
   左侧“国际文献总库”进入，输入文章标题，即可查询

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org