Математички Билтен
19 (XLV)
1995 (117–122)
Скопје, Македонија

ISSN 0351–336X

EMPIRICAL ANALYSIS OF THE ALGORITHM FOR FINDING ALL MINIMUM–HOP PATHS IN NETWORKS

Iskra K. Djonova–Popova

Abstract

A fairly simple, but quite useful extension of Dijkstra's shortest path algorithm for finding all minimum–hop paths in networks was presented in [1]. In order to pursue empirical study of the previous analytical results, both the rudimentary and the modified algorithm were implemented upon a simple of randomly generated networks. The results obtained confirm the relationship concerning the computational complexity of both algorithms, namely that they are of same order in their requirement of the computing resources.

1. Introduction

Analyzing an algorithm usually means predicting the resources that the algorithm requires. Most often the memory used and the computational time are of primary concern. In general, the time and the storage required by the algorithm grow with the size of the input that depends on the problem being studied. In graph algorithms the natural measure of the input size is the number of vertices and the number of links. Alternatively, instead of the number of links, the density of the graph could be used.

Following the notation in [1] let the network be modeled as a graph G, with a vertex set $V = \{1, 2, \ldots, n\}$ and a link set $E = \{1, 2, \ldots, m\}$. Its density is represented as a ratio of the number of links in G and the number of links in the complete graph with n vertices, $r = 2m/n(n-1)$. The graph contains neither parallel links, nor loops and the length associated with each link is 1. There is an arbitrary and distinguished node, s, in V called the source node. The Dijkstra's algorithm, [2], finds a single minimum–hop path from s to every other vertex in the graph and produces a minimum–hop path tree with the source node as a root. Its modified version, [1], constructs all minimum hop paths. The time complexity of both algorithms is of the same order.

In the paper, this is verified by computer programs tested upon randomly generated graphs with different input size. Moreover, as a result, the relationship between the processing time required and the complexity of the networks is established.

2. The empirical approach

The examination of the behavior of the algorithms required four separate procedures. The procedure RGCG (Randomly Generated Connected Graph) constructed randomly generated connected graph without parralel links and loops for a given input size (n vertices and m links). The basic data structure used for the graph representation was the adjacency list. SMHP (Single Minimum Hop Path) found a minimum–hop path from the source to all the other vertices, and AMHP (All Minimum–Hop Paths) located all minimum hop paths. A few additional data structures associated with each vertex in the graph were needed to store its distance and predecessors. The data structures for storing the predecessors and thus the memory requirements for AMHP were increased according to the number of multiple paths obtained. Finally, NCTR (Normalized Computational Times and Ratios) performed the computation of normalized CPU times for SMHP and AMHP and the ratios needed for comparasion of both algorithms.

The similar times obtained with different source vertices indicates that the choice of the source vertex has no significant impact on the results. Without loss of generality only the results for vertex 1 as a source are presented here. The machine indepedence of the running times is achieved by their normalization relative to the time for the smallest input size graph. The results are summarized in Table 1.

Table 1

n	m	r	T_S	T_A	T_A/T_S	N_A	N_S	N_A/N_S
10	9	0.2	1.0	1.0	1.000	9	9	1.000
10	13	0.3	1.5	1.5	1.000	9	10	1.111
10	18	0.4	1.0	1.5	1.500	9	14	1.556
10	23	0.5	1.5	1.5	1.000	9	14	1.556
10	27	0.6	1.5	2.0	1.333	9	17	1.889
20	19	0.1	5.5	0.5	1.000	19	19	1.000
20	38	0.2	7.0	7.5	1.071	19	28	1.474
20	57	0.3	9.0	10.0	1.111	19	36	1.895
20	76	0.4	10.5	11.5	1.095	19	46	2.421
20	95	0.5	12.0	12.5	1.042	19	45	2.368
20	114	0.6	13.0	14.0	1.077	19	57	3.000
30	44	0.1	18.5	19.0	1.027	29	38	1.310
30	87	0.2	29.0	30.0	1.034	29	62	2.138
30	131	0.3	37.5	39.5	1.053	29	73	2.517
30	174	0.4	46.0	47.0	1.022	29	85	2.931
30	218	0.5	52.5	55.0	1.048	29	114	3.931
30	261	0.6	58.5	61.5	1.051	29	141	4.862
40	78	0.1	49.0	50.5	1.031	39	58	1.487
40	156	0.2	83.0	84.5	1.018	39	83	2.128
40	234	0.3	113.0	115.5	1.022	39	110	2.821
40	312	0.4	139.0	142.5	1.025	39	154	3.949
40	390	0.5	164.5	169.0	1.027	39	206	5.282
40	468	0.6	183.0	188.0	1.027	39	234	6.000
50	123	0.1	111.5	112.5	1.009	49	83	1.694
50	245	0.2	193.5	196.5	1.016	49	119	2.429
50	368	0.3	279.0	279.0	1.000	49	169	3.449
50	490	0.4	339.5	345.5	1.018	49	218	4.449
50	613	0.5	396.0	405.0	1.023	49	311	6.347
50	735	0.6	442.0	450.5	1.019	49	359	7.327

The column labels are as follows:
n – number of vertices, m – number of links,
r – density of the graph G,
T_S – normalized computational time for procedure SMHP,
T_A – normalized computational time for procedure AMHP,
N_S – number of minimum – hop paths found in SMHP,
N_A – number of minimum – hop paths found with AMHP.

Diagrams of computational time for AMHP as a function of the number of vertices for graphs with density $r = 0.2$ and $r = 0.5$ are presented on Fig. 1 and Fig. 2 respectively.

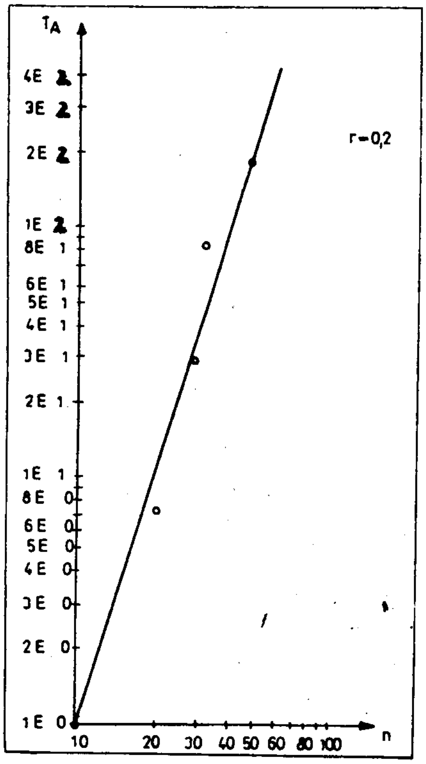

Fig. 1

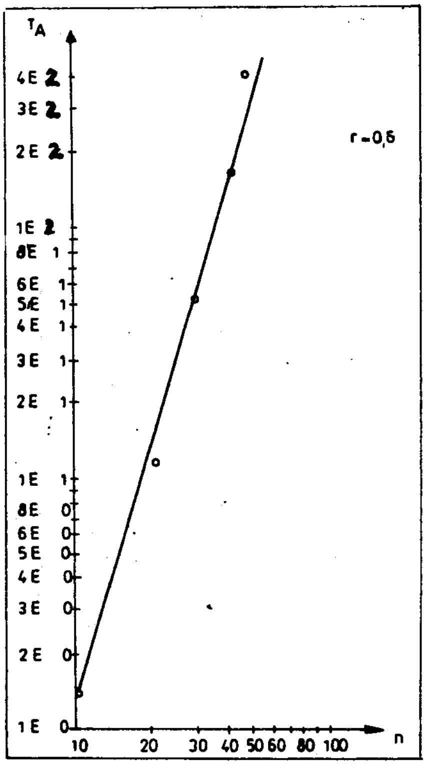

Fig. 2

3. Conclusions

From Table 1 and Fig. 1 and Fig. 2, it may be concluded that the CPU time of the modified version of Dijksta's algorithm to find all minimum-hop paths in a network grows equally with the one needed for Dijkstra's algorithm. On the other hand its memory requirements are larger only for the number of additional paths found. Hence, both algorihms have similar performances with respect to the time and the storage used.

Consequently, in any particular application that needs a better insight in paths from the source, AMHP could be used without concern that the time for processing the network will be overcome.

Acknowledgment: The author extends his appreciation to Z. Zografski for the helpful comments in his review of [1].

References

[1] Djonova–Popova K. I., Popov, B. O.: *Finding All Minimum–hop Paths in Networks*, Bulletin de la Société des mathematicien et des informaticien de Macedoiné, **16**, 99–104, 1992

[2] Dijkstra, E.: *A note on two problems in connection with graphs*, Numer. Math. vol. **1**, 269–271, 1959

ЕМПИРИСКА АНАЛИЗА НА АЛГОРИТМОТ ЗА ОДРЕДУВАЊЕ НА СИТЕ ПАТИШТА СО МИНИМАЛЕН БРОЈ НА ЛИНИИ ВО МРЕЖИ

Искра К. Џонова–Попова

Р е з и м е

Алгоритмот за одредување на сите патишта со минимален број на линии, [1], претставува едноставна, но корисна модификација на алгоритмот на Dijkstra за најкус пат. Практичната реализација на овој алгоритам, како и на алгоритмот на Dijkstra беше тестирана врз случајно генерирани мрежи со различна комплексност. Мерените времиња ги потврдуваат аналитички добиените резултати дека ефикасноста на двете постапки е приближно еднаква.

Centar za energetika i informatika,

MANU

Krste Misirkov 2

91000 Skopje

Makedonija