IEICE TRANS. COMMUN., VOL.E82–B, NO.4 APRIL 1999

PAPER

A K-Best Paths Algorithm for Highly Reliable Communication Networks*

Shi-Wei LEE† ***and*** **Cheng-Shong WU**†, ***Nonmembers***

SUMMARY In highly reliable communication network design, disjoint paths between pairs of nodes are often needed in the design phase. The problem of finding k paths which are as diverse as possible and have the lowest total cost is called a k-best paths problem. We propose an algorithm for finding the k-best paths connecting a pair of nodes in a graph G. Graph extension is used to transfer the k-best paths problem to a problem which deploits well-known maximum flow (MaxFlow) and minimum cost network flow (MCNF) algorithms. We prove the k-best paths solution of our algorithm to be an optimal one and the time complexity is the same as MCNF algorithm. Our computational experiences show that the proposed algorithm can solve k-best paths problem for a large network within reasonable computation time.

key words: *K-best paths, disjoint paths, reliable network design*

1. Introduction

Two kinds of path finding procedures are often needed in the design of reliable communication networks. The first one is to find k shortest paths between a pair of nodes. Those paths may be simple or allow loops. For the k shortest simple paths problem, Lawler proposed the best known algorithm in computation order $\mathcal{O}(k(m + n\log n))$ in undirected graphs [8], where n and m are the number of nodes and links of the input network. For the directed counterpart, Katoh et al. gave the best known bound in $\mathcal{O}(kn(m + n\log n))$[7]. Recently Eppstein developed an efficient algorithm for finding the k shortest paths allowing loops in $\mathcal{O}(m + n\log n + k)$ [5].

The other kind of path finding procedure is about disjoint paths. In [6], the authors proved that to find the maximum disjointed paths with length constraints (hop limitation) for hop count $h \geq 5$ is an NP-complete problem. Due to the intractable characteristics of the problem, several literature published on the k-shortest link-disjointed paths problem with unity link cost and hop limitation employed heuristic methods based on matrix multiplications [10], [13], [14].

The network reliability is not guaranteed by using the method in [13], [14] because the number of link-disjointed paths is over estimated. Oki et al. proposed a modified matrix multiplication method which used a recursive matrix-calculation (RM) method to get the link-disjointed paths [10]. The computation complexity for [10] is $\mathcal{O}(khM(n))$ for getting k link-disjointed paths with hop limitation h. Matrix multiplication needs $M(n)$ computation time which is $\Omega(n^2)$. The authors suggest to use a super computer equipped with vector processors to implement the RM method. In our opinion, the computation time can be reduced to $\mathcal{O}(km)$ by using Breadth First Search (BFS) in the path search steps of the RM method.

In this paper, we focus on finding k-best paths without hop limitation. The problem of finding k paths which are as diverse as possible and have the lowest total cost is called a k-best paths problem. Dunn et al. [4] proposed a successively shortest path algorithm which is not only a heuristic method but also may even overestimate the number of link-disjointed paths. Suurballe et al. [11] gave an algorithm for finding two disjoint paths. Castanon [3] gave an $\mathcal{O}(n^3\log n)$ algorithm for finding the k-best disjoint paths by using minimum cost network flow (MCNF) algorithm. However, the algorithm can only be applied in a trellis graph. Recently Nikolopoulos et al. [9] provided an algorithm to transfer an arbitrary graph to a trellis graph and then to obtain the k-best paths by Castanon's Algorithm. The computation time for the transformation is not given in their paper. However, in our observation, one has to transform a graph with n nodes to a corresponding trellis graph containing n^2 nodes in the worst case. Thus, the computation time to get the k-best paths in [9] is $\mathcal{O}(n^6\log n)$. In addition, the solution provided by [9] is not optimal since the algorithm is only a heuristic one.

In this paper, we propose a k-best paths (KBP) algorithm for arbitrary networks. In particular, graph extension is used to transfer the KBP problem to a problem which deploits well-known maximum flow (MaxFlow) and minimum cost network flow (MCNF) algorithms. For any input digraph, we prove that our KBP algorithm can output k-best paths optimally in $\mathcal{O}(c(n, m, k))$, where $c(n, m, k)$ is the time complexity for minimum cost network flow algorithm for a graph with n nodes, m links, and k units of flow.

If the desired number of paths, k, is larger than the maximum number of mutually disjointed paths the network can support, the k-best paths algorithm should

Manuscript received April 13, 1998.
Manuscript revised August 23, 1998.

†The authors are with the Department of Electrical Engineering, National Chung Cheng University, Chiayi 621, Taiwan, R.O.C.

*This work is supported in part by National Science Council, Taiwan, R.O.C., under Grant NSC 87-2213-E-194-044.

output the k paths with the lowest total cost and minimum number of common nodes. Although [9] tries to deal with this issue, only networks with cutpoints† are considered and the algorithm can not be applied to arbitrary networks.

The paper is organized as follows. In Sect. 2, related notation and graph transformation are described. Section 3 gives the KBP algorithm and its verification. Section 4 analyzes the time complexity for KBP algorithm. Some computational experiences are presented in Sect. 5. Finally, we give a brief conclusion in Sect. 6.

2. Background

Let $G(\mathcal{N}, \mathcal{L})$ be a directed network, where $\mathcal{N} = \{1, ..., n\}$ is a finite set whose elements are called nodes and $\mathcal{L} = \{1, ..., m\}$ is a finite set whose elements are called arcs or links. Each link l is defined by an ordered pair (i, j) of nodes, where i is the tail node and j is the head node. In what follows we assume that $i \neq j$, for any arc $(i, j) \in \mathcal{L}$.

Let $s \in \mathcal{N}$ and $t \in \mathcal{N}$ be two distinct nodes of G. A path p from s to t in G is a sequence of nodes and arcs of the form $p = \{s = n_1, (n_1, n_2), n_2, \ldots, n_{k-1}, (n_{k-1}, n_k), n_k = t\}$.

Let $c_{i,j}$ be a nonnegative cost value associated with link $(i, j) \in \mathcal{L}$ and let $c(p) = \sum_{(i,j)\in p} c_{i,j}$ be a function which associates a nonnegative cost to the path.

Let P_G be the k-best paths output from the KBP algorithm. $P_G = \{p_1, p_2, \ldots, p_k\}$. Total cost for the solution P_G is defined as $C(P_G) = \sum_{p\in P_G} c(p)$.

Note that, although in what follows the input graph G is a digraph, any undirected graph can also be applied to KBP algorithm by applying the transforming technique to become a directed one [1]. The computation complexity for an undirected graph still remains the same order as a directed one.

Throughout the paper, we assume the desired number k is greater than one. For k being equal to one, k-best paths problem is just the same as the shortest path problem. Following notation is needed in our algorithm.

Notation:
$\mathcal{N}, \mathcal{L}$: [node, link] set in digraph G
k: the number of desired best paths
n, m: number of [nodes, links] in graph G
s, t: [source, termination] node
G_1: graph obtained from G by letting link capacity to be unity
G_2: graph obtained from G by splitting every node of G_1 into two nodes and adding an artificial link between the splitted nodes
G_3: graph obtained from G_1 by adding additional artificial nodes and links
N_l: number of maximum link-disjointed paths in G
N_d: number of maximum mutually disjointed paths in G
M: a big number
MaxFlow(G, s, t): maximum flow algorithm on G between s and t
MCNF(G, k, s, t): minimum cost network flow for input graph G and k unit flow supplied between s and t
P_G: k-best paths output from KBP algorithm.
P_{G2}: k paths output from MCNF for graph G_2
P_{G3}: k paths output from MCNF for graph G_3

Before presenting the algorithm, let us first clarify the following terminology:

mutually disjointed paths: Paths that are both node-disjointed and link-disjointed for each other.
link-disjointed paths: Paths that are link-disjointed for each other.
K-best paths algorithm: Algorithm for finding k paths with lowest total link cost between a source-termination pair. These k paths are mutually disjointed if there exists such k mutually disjointed paths in graph G; otherwise the algorithm is to find k link-disjointed paths with the least number of nodes jointed.
in-degree: Number of links entering the interested node.
out-degree: Number of links leaving the interested node.

Several graphs denoted G_1, G_2, G_3 are needed in the KBP algorithm. Input Graph G is a digraph with non-negative link cost. Graph G_1 is obtained from G by setting the link capacity to be unity while reserving the same network topology. The reason for assigning unity link capacity in G_1 and other graphs used in the paper is to enforce the mutual exclusivity constraints such as link-disjointed or node-disjointed when we apply MaxFlow or MCNF algorithms to the graphs.

Graph G_2 is also obtained from graph G (Fig. 1). For every node in set $\mathcal{N} - \{s, t\}$, if min{in-degree, out-degree} ≥ 2, the node, say j, is splitted into two nodes and an artificial link is added between the two

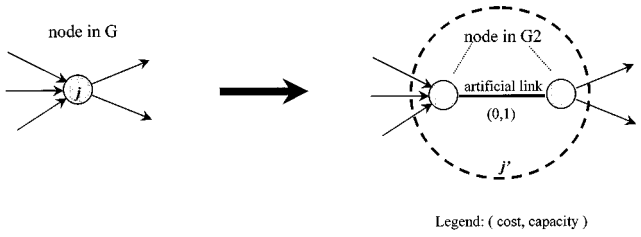

Fig. 1 Node splitting for graph G_2.

†A vertex is called a cutpoint if the network becomes disconnected when the vertex is removed. A network with cutpoints is a network which becomes disconnected if any one of the cutpoints is removed.

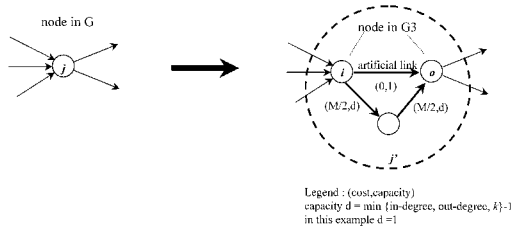

Fig. 2 Node splitting for graph G_3.

nodes to form a complex node j'. The link cost of artificial link is set to be zero. The capacity of every link including the artificial one is set to be unity.

Graph G_3 is derived from G_1. For each node in set $\mathcal{N} - \{s, t\}$, say node j, one of the two cases may happen. If $\min\{\text{in-degree}, \text{out-degree}\} = 1$, no additional process is needed. In the other case, if $\min\{\text{in-degree}, \text{out-degree}\} \geq 2$, the node is splitted into three sub nodes and three links are added to form a complex node j' as shown is Fig. 2 . One sub node, say i, is connected to all incoming links, while the other one, say o, is connected to all outgoing links. An artificial link with the cost of zero and unity capacity is added to connect sub node i and sub node o. The other two artificial links have link cost M/2 and capacity $d = \min\{\text{in-degree}, \text{out-degree}, k\} - 1$, where M is a big number greater than $\sum_{\{i,j\}\in\mathcal{L}} c_{i,j}$.

We define a mapping from paths obtained from graph G_2 to G, denoted $P_G \longleftarrow P_{G2}$, by replacing paths through node j' to node j. The same definition is also applied to mapping paths between P_{G3} and P_G.

3. The KBP Algorithm

The algorithm is stated in Fig. 3.

Since the capacity of each link in G_1 is set to be unity, each link can be selected at most once when applying the MaxFlow algorithm to graph G_1. Thus, step 2 gives us the maximum number of of the link-disjoited path in G_1.

Similarly, because only single unit of flow can pass through complex node j' in G_2, step 8 gives us N_d, the maximum nuber of the mutually disjointed paths. In the case of $k \leq N_d$, we can just run the MCNF algorithm on G_2 to find the k-best mutually disjointed paths on step 11. However, if $k > N_d$, the number of the mutually disjointed paths is less than k, i.e., there must be some node-jointed paths in the output of the algorithm. Assigning big M on the cost of the artificial links in the complex node in G_3 forces the MCNF algorithm not to choose a node more than once unless there are no other path to transfer the flow. Thus, steps 17–19 give us the k-best paths in which some of them are node jointed.

Algorithm KBP(G, k, s, t):

```
Begin
1. Get G1 from G;
2. Nl=MaxFlow(G1, s, t);
3. If k > Nl
4.    return('Cannot find k link-disjointed paths in G');
5. Else
6.    Begin
7.       Get G2 from G;
8.       Nd=MaxFlow(G2, s, t);
9.       If k ≤ Nd
10.         Begin
11.            Get PG2 by running MCNF(G2, k, s, t);
12.            PG ⟵ PG2; /*mapping PG2 to PG*/
13.            return(PG); /*K-best paths found, all of
                             them are mutually disjointed */
14.         End
15.      Else
16.         Begin
17.            Get G3 from G1;
18.            Get PG3 by running MCNF(G3, k, s, t);
19.            PG ⟵ PG3; /*mapping PG3 to PG*/
20.            return(PG); /*K-best paths found, some of
                             them are node jointed */
21.         End
22.   End
End.
```

Fig. 3 K-best paths algorithm.

3.1 Verification of KBP Algorithm

Definition 1: The number $N(P_G)$ for path set P_G is defined as $\sum_{v\in\mathcal{N}}[N_v(P_G) - 1]^+$, where $N_v(P_G)$ is total number of paths in P_G through node v and $[x]^+ = \max\{x, 0\}$.

Note that $N(P_G)$ is just the number of common nodes used by paths in P_G.

Before giving lemma and theorems, we first emphasize the following two equations are true between any path sets P_G and its corresponding P_{G2} and P_{G3}.

$$C(P_{G2}) = C(P_G)$$
$$C(P_{G3}) = N(P_G) \times M + C(P_G)$$

Since the big M is a big number greater than $\sum_{\{i,j\}\in\mathcal{L}} c_{i,j}$, the following inequality also holds.

$$N(P_G) \times M \leq C(P_{G3}) < (N(P_G) + 1) \times M$$

Lemma 1: If k mutually disjointed paths can be found in G, then the algorithm KBP outputs the optimal solution.

Proof: Let P_G be the output from KBP algorithm. Let lemma 1 be false. Assume that there exists another solution $\bar{P}_G$ such that $C(P_G) > C(\bar{P}_G)$. Since the cost of artificial edge in G_2 is zero, $C(P_G) = C(P_{G2})$. For graph G_2, $C(P_{G2}) \leq C(\bar{P}_{G2}) = C(\bar{P}_G)$ always holds because $C(P_{G2})$ is obtained from minimum cost flow algorithm. We have a contradiction. □

Lemma 2: If P_{G3} is an optimal solution got from MCNF for graph G_3, $N(P_G)$ is minimum.

Proof: Let lemma 2 be false. There exists $\bar{P}_G$ such that $N(\bar{P}_G) < N(P_G)$. Since P_G is optimal, $N(P_G) \times M \leq C(P_{G3}) \leq C(\bar{P}_{G3}) < (N(\bar{P}_G)+1) \times M$. Therefore, $N(P_G) < N(\bar{P}_G) + 1$, a contradiction. □

Lemma 3: P_{G3} is optimal to graph G_3 if and only if P_G has minimum $N(P_G)$ and minimum $C(P_G)$.

Proof: We first prove the only if part. Since P_{G3} is optimal to graph G_3, $N(P_G)$ must be minimum by lemma 2. Let $C(P_G)$ not be minimum. Then there exists optimal $\bar{P}_G$ to G with $C(\bar{P}_G) < C(P_G)$. From lemma 2, $N(\bar{P}_G) = N(P_G)$. Therefore, we have $C(\bar{P}_{G3}) = N(\bar{P}_G) \times M + C(\bar{P}_G) < N(P_G) \times M + C(P_G) = C(P_{G3})$ which is a contradiction. Hence $C(P_G)$ must be minimum.

For the if part, assume $\bar{P}_{G3}$ is optimal to graph G_3. $C(\bar{P}_{G3}) = N(\bar{P}_G) \times M + C(\bar{P}_G) \geq N(P_G) \times M + C(P_G) = C(P_{G3})$. Therefore, P_{G3} is also optimal. □

Theorem 1: If input graph G has k-best paths, algorithm KBP outputs one optimal solution.

Proof: Immediate from Lemma 1 and Lemma 3. □

4. Computational Complexity

Theorem 2: KBP algorithm generates the k-best paths at a computational effort of $\mathcal{O}(c(n, m, k))$.

Proof: The computational effort in each step of KBP is analyzed as follows.

Step 1. Graph G_1 can be obtained in $\mathcal{O}(m)$.

Step 2,8. Both of them can be performed in $\mathcal{O}(\min\{n^{2/3}m, m^{3/2}\})$ time because both graph G_1 and G_2 have $\mathcal{O}(n)$ number of nodes and $\mathcal{O}(m)$ number of links which have all unity link capacity.

Step 7,17. Graph G_2, G_3 can be obtained in $\mathcal{O}(n)$.

Step 11,18. Since graph G_2 and G_3 have $\mathcal{O}(n)$ number of nodes and $\mathcal{O}(m)$ number of links, the computational complexity of MCNF algorithm is then denoted by $\mathcal{O}(c(n, m, k))$. The best known algorithm for MCNF is $\mathcal{O}(\min\{A1, A2, A3\})$, where $A1 = nm \log(n^2/m) \log nM$, $A2 = nm(\log\log k)\log(nM)$ and $A3 = (m \log n)(m + n \log n)\}$ [1].

The analysis shows the computational effort dominated by minimum cost network flow algorithm, i.e. $\mathcal{O}(c(n, m, k))$. □

5. Experimental Results

We apply our algorithm to graphs generated by GRIDGEN, a program generating random graph developed by [15]. The codes of maximum flow algorithm, ϵ-Relax-MF and the minimum cost network flow algorithm, RELAX-IV, can be found in [15] and [16].

Table 1 Experimental results.

n	m	k	mutually disjointed	CPU time (sec)
100	1000	8	Y	6.714
100	1000	9	N	6.781
100	2000	20	Y	10.448
100	2000	23	N	10.465
100	3000	27	Y	11.049
100	3000	33	N	11.196
200	4000	21	Y	11.849
200	4000	22	N	11.882
200	5000	23	Y	12.516
200	5000	24	N	12.549
200	6000	28	Y	13.199
200	6000	31	N	13.345
400	12000	29	Y	14.478
400	12000	30	N	14.568
400	14000	31	Y	15.681
400	14000	33	N	15.864
400	16000	36	Y	16.999
400	16000	38	N	17.232

The program is written by FORTRAN and runs on IBM RS6000 machine. Table 1 summaries the computational results. We show the CPU time (in second) and whether the output k-best paths are mutually disjointed or not for different number of nodes (n), links (m), and desired number of disjoint paths (k). The experimental results show the KBP algorithm can solve k-best paths problem for a large network within reasonable computation time.

6. Conclusions

In this paper, a KBP algorithm which solves k-best paths problem for highly reliable communication network is proposed. KBP algorithm generates the k-best paths at a computational effort of $\mathcal{O}(c(n, m, k))$. The solution output by KBP is a real optimal solution for k disjoint paths and it is very useful for planning highly reliable communication networks.

Acknowledgement

The authors would greatly appreciate the anonymous reviewers' comments for improving the quality of this paper.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, "Network Flows," Prentice-Hall, 1993.

[2] Y.P. Aneja and K.P.K. Nair, "The constrained shortest path problem," Naval Research Logistics, Quarterly 25 pp.549–555, 1978.

[3] D. Castanon, "Efficient algorithms for finding the K best paths through a trellis," IEEE Trans. AES, vol.26, pp.405–410, 1990.

[4] D.A. Dunn, W.D. Grover, and M.H. MacGregor, "Comparison of K-shortest paths and maximum flow routing for network facility restoration," IEEE J. Sel. Areas Commun.,

vol.12, no.1, pp.88–99, 1994.
[5] D. Eppstein, "Finding the K shortest paths," SIAM J. Computing, vol.28, no.2, pp.652–673, 1998.
[6] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," Networks, vol.12, pp.277–286, 1982.
[7] N. Katoh, T. Ibaraki, and H. Mine, "An efficient algorithm for K shortest simple paths," Networks, vol.12, pp.411–427, 1982.
[8] E.L. Lawer, "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem," Management Science vol.18, pp.401–405, 1972.
[9] S.D. Nikolopoulos, A. Pitsillides, and D. Tipper, "Addressing network survivability issues by finding the K-best paths through a trellis graph," Proc. IEEE INFOCOM, Kobe, Japan, 1997.
[10] E. Oki and N. Yamanaka, "A recursive matrix-calculation method for disjoint path search with hop link number constraints," IEICE Trans. Commun., vol.E78-B, no.5, pp.769–774, 1995.
[11] J.W. Suurballe and R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," Networks, vol.14, pp.325–336, 1984.
[12] J.W. Suurballe, "Disjoint paths in a network," Networks, vol.4, pp.125–145, 1974.
[13] Y. Tanaka, M. Akiyama, and B. Wallstrom, "A design method for cost minimization of highly reliable communication networks," IEICE Trans., vol.J71-B, no.4, pp.483–489, April 1988.
[14] Y. Tanaka, F. Rui-xue, and M. Akiyama, "Design method of highly reliable communication network by the use of the martrix calculation," IEICE Trans., vol.J70-B, no.5, pp.551–556, May 1987.
[15] D.P. Bertsekas, "Linear Network Optimization: Algorithms and Codes," The MIT Press, 1992.
[16] Jiefeng Xu, http://ucsu.colorado.edu/ xu/software.html.

Shi-Wei Lee was born in Taipei, Taiwan, R.O.C., in 1970. He received his B.S. degree in electrical engineering from Tatung Institute of Technology, Taipei, Taiwan, R.O.C., in 1992. He is now working towards his Ph.D. degree in electrical engineering at National Chung Cheng University. His research interests include high-speed networks, network planning and computer network optimization.

Cheng-Shong Wu was born in Taoyuan, Taiwan, R.O.C., in 1961. He received the B.S. and M.S. degrees in electrical engineering in 1983 and 1985 from National Taiwan University, Taiwan, and Ph.D. degree in electrical engineering from University of Southern California, USA, in 1990. During 1991, he was a Visiting Assistant Professor in the Center for Advanced Computer Studies at the University of Southwestern Louisiana. He joined the Department of Electrical Engineering at National Chung Cheng University, Taiwan, R.O.C., in 1992. Currently he is an Associate Professor in the department. His research interests include high-speed networks, wireless networks, queueing theory, and mathematical programming.