Scheme to Find $k$ Disjoint Paths in Multi-Cost Networks

Ruchaneeya Leepila, Eiji Oki, and Naoto Kishi,
Department of Information and Communication Engineering,
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, JAPAN
E-mail: ruchanel@ice.uec.ac.jp, oki@ice.uec.ac.jp, kishi@ice.uec.ac.jp

Abstract—This paper proposes a scheme to find $k$ disjoint paths in multi-cost networks. This scheme, called the $k$-penalty scheme with initial arc cost matrix (KPI), penalizes the use of conflicting arcs found in previously set paths and increases the costs of these arcs in accordance with the initially given arc cost matrix. Simulations show that the KPI scheme is able to find $k$ disjoint paths faster than the conventional scheme that uses the incrementally updated auxiliary arc cost matrix to increase the cost of conflicting arcs in our examined networks. Moreover, the KPI scheme yields $k$ disjoint paths with lower total cost than the conventional scheme.

Index Terms—Network survivability, multi-cost network, $k$-disjoint paths, $k$-penalty algorithm.

I. INTRODUCTION

Disjoint path routing enhances the survivability of a network [1], [2], [3]. Disjoint paths, i.e., paths that do not share the same links or nodes, must be set between source and destination nodes to minimize the damage created by network failure. If $k$ disjoint paths are set between source and destination nodes, at least one path is protected against $k-1$ simultaneous failures. It means that a backup path will be used when a working path fails. To utilize efficient resources, the shared backup path protection is applied.

The problem of finding disjoint paths in a single-cost network has been widely studied [4], [5], [6], [7], [8]. In a single-cost network, the cost of each network arc is the same for all $k$ paths. Several algorithms have been introduced to find $k$ disjoint paths. The active path first approach (APF) presented in [4] finds the first shortest path by using a shortest path algorithm such as Dijkstra algorithm [5]. Then, it finds the next shortest path after removing the previously found paths. This procedure is repeated until the required number of disjoint paths is obtained. Suurballe’s algorithm [6], or its modification, Bhandari’s algorithm [7], finds disjoint paths and the total cost of all paths is minimized. The problem of finding such disjoint paths is called the Min-Sum problem.

In [9], J. Rak proposed the $k$-penalty algorithm to find $k$-disjoint paths in a multi-cost network. In a multi-cost network, each network arc can have a different cost for all $k$ paths. Its applications usually lie in the field of shared backup path protection [10], [11]. In this case, the cost of an arc for a backup path is often a fraction of that for a working path. The Min-Sum problem in a multi-cost network is NP-Complete (NPC) [4], [7], [12]. The $k$-penalty algorithm finds the shortest path as the first path. The arcs on the shortest path and those connected to the transit nodes on this path are considered as forbidden arcs for the next disjoint path to be found. Although the APF algorithm assigns the forbidden arcs infinitely high cost, the $k$-penalty algorithm gives them finite costs to avoid the trap problem [9]. That is, the next path must pay a penalty for using a forbidden arc. Forbidden arc cost is increased by the path cost of the previously found path. Arc costs are incrementally updated and kept in an auxiliary arc cost matrix. If any conflict, i.e., the current path is not disjoint with all previously found paths, occurs, all found paths are deleted. Before starting the process of finding $k$ disjoint paths again, the costs of conflicting arcs are incrementally increased by the cost of the last found path in the previous iteration. The path cost is computed using the auxiliary arc cost matrix. Thus, in this paper, we call this algorithm KPA, or $k$-penalty by using the auxiliary arc cost matrix to compute the path cost. This procedure, including the deletion of found paths, is iterated until $k$ disjoint paths are found, or the number of iterations reaches a number specified to avoid infinite loops.

We found that KPA sometimes fails even with a large number of iterations, even though disjoint paths actually exist. With every iteration, or conflict, the arc costs in the auxiliary arc cost matrix are increased. In order to avoid traversing arcs with large costs, the algorithm may find a path that overlaps already used paths, including forbidden arcs from the previous paths. This path overlapping causes the deletion of found paths and restarts the process. It takes time to find $k$ disjoint paths or sometimes they cannot be found at all. This problem must be solved to find $k$ disjoint paths in an efficient manner.

This paper proposes a $k$ disjoint path scheme based on the KPA scheme that eliminates the KPA problem. The proposed scheme uses the same penalty process but the cost increases are determined from the initially given arc cost matrix. Thus, this scheme is called KPI: $k$-penalty with the initial arc cost matrix. Numerical results show that the KPI scheme is able to find $k$ disjoint paths faster than the KPA scheme. Moreover, KPI yields disjoint paths with lower average total cost than the KPA scheme.

The remainder of this paper is organized as follows. Section II describes the KPA scheme. Section III presents the KPI.
scheme. Section IV compares the performance of the KPI scheme to that of the KPA scheme. Section V summarizes this paper.

II. KPA: k-PENALTY WITH AUXILIARY ARC COSTS MATRIX

This section presents the KPA scheme and its weakness.

A. Terminology

The terminology used in this paper is shown below.

- $d_r$: Demand to find a set of end-to-end $k$ disjoint paths between a pair of nodes $(s_r, t_r)$
- $s_r$: Source node of demand $d_r$
- $t_r$: Destination node of demand $d_r$
- $i_{max}$: Maximum allowable number of conflicts
- $p$: Index of path 1, ..., $k$
- $\eta_p$: $p$th path
- $a_h$: $h$th arc, where $h = 1, 2, \ldots$
- $\xi_h$: Cost of each arc $a_h$
- $\xi^P_h$: Cost of arc $a_h$ of the $p$th path
- $\xi^{aux}_h$: Auxiliary cost of arc $a_h$
- $\xi^{aux,p}_h$: Auxiliary cost of arc $a_h$ of the $p$th path
- $\Xi^{aux}$: Initial matrix of arc cost $\xi^P_h$
- $\Xi^{aux,p}$: Auxiliary matrix of arc cost $\xi^{aux,p}_h$
- $i_c$: Conflict counter

B. Description

The KPA scheme is shown in Fig. 1. Demand $d_r$ to find $k$ disjoint paths from source node $s_r$ to destination node $t_r$, the arc cost matrices for each disjoint path, and the maximum allowable number of conflicts, $i_{max}$, are initially given. The KPA scheme outputs the set of $k$ disjoint paths and the total costs of the $k$ disjoint paths. KPA uses the shortest-path-based algorithm. At Step 1, the conflict counter, $i_c$, is set to 1 and the initial cost matrix of the $p$th path, $\Xi^P$, is copied to the auxiliary cost matrix of the $p$th path, $\Xi^{aux,p}$, for all paths, $p = 1, \ldots, k$. $\Xi^P$ is kept to compute the total path cost using Eq. (1) after finding $k$ disjoint paths. At Step 2, set $j = 1$ to find the first path. In Step 3, $\Xi^{aux,p}$ is copied to $\Xi^{aux}$. Step 4 is skipped if $j = 1$. To find the next paths $\eta_j$ ($j \neq 1$), path $\eta_j$ has to pay a penalty for using one of the forbidden arcs, i.e., links traversed by previously found paths $\eta_1$ (link disjoint), or links corrected to transit nodes used by previously found paths (node disjoint). The cost of the forbidden arcs is increased by the costs of all $j - 1$ previously found paths at Step 4. At Step 5, $\eta_j$ is found as the shortest path on the network with auxiliary cost matrix $\Xi^{aux}$. At Step 6, if $\eta_j$ is disjoint with the $(j - 1)$ previously found paths, the index number of path, $j$, is increased by one and the process goes to Step 7 to check if the required number of $k$ disjoint paths has been obtained. The process terminates if the number of found disjoint paths has reached the required number, $k$. Otherwise, the process will find the next path by reentering.

Step 3. If $\eta_j$ is not disjoint (link or node) with the $(j - 1)$ previously found paths, a conflict is called and the costs $\xi^{aux,1}_h, \ldots, \xi^{aux,k}_h$ of each conflicting arc $a_h$, the link shared between the previously found $j - 1$ paths and path $\eta_j$, or the link connected to the node shared between previously found $j - 1$ paths and path $\eta_j$, is increased by the path cost $\xi^{aux}_h$ of $\eta_j$, which is computed from auxiliary costs matrix $\Xi^{aux}$ (Step 6a) as shown in Eq. (4). After increasing each conflicting arc $a_h$, all found paths are deleted and conflict counter, $i_c$, is increased by one. If $i_c$ is greater than the maximum allowable number of conflicts, $i_{max}$, the process is terminated. If $i_c$ is less than $i_{max}$, the process reenters Step 2.

C. Example of KPA Scheme

The KPA scheme is demonstrated with an example in Fig. 2. The example shows how to find the $k$ disjoint paths in a multi-cost network with $k = 3$ for the demand between node 1 to 7. Node disjoint paths are considered as disjoint paths in this example. Fig. 2 (a1), (a2) and (a3) are multi-cost network that has three sets of arc costs; one for the first path, $\xi_1$: the second path: $\xi_2^P$; and the third path: $\xi_3^P$. The scheme starts by setting the auxiliary cost matrix as $\Xi^{aux,p} = \Xi^P$ for $p = 1, 2, 3$ and $i_c = 1$. The scheme then considers at the first path $j = 1$ and sets the auxiliary cost matrix $\Xi^{aux} = \Xi^{aux,1}$. The first path $\eta_1 (1-4-7)$ is found as the shortest path, shown as Fig. 2 (b1). The costs, $\xi^{aux,1}_h$, for arcs incident to transit nodes of path $\eta_1$ of the set of arc costs $\xi^{aux,2}_h$ are increased by path cost $\xi^{aux,1}_h$ of path $\eta_1$, which is equal to 14 in the example, as shown in Fig. 2 (b2). Then, path $\eta_2 (1-3-5-7)$ is found. To find the third path, the costs of forbidden arcs of paths $\eta_1$ and $\eta_2$ on the network are increased by path cost $\xi^{aux,1}$ of path $\eta_1$ and path cost $\xi^{aux,2}$ of path $\eta_2$. However, $\eta_3 (1-4-7)$, which is not disjoint with $\eta_1$, is found, as shown in Fig. 2 (b3). Costs $\xi^{aux,p}_h$ of arcs incident to node 4 on $\eta_3$ for all paths, $p = 1, \ldots, k$, are increased by path cost $\xi^{aux}_h$, as shown in Eq. (4). The path cost $\xi^{aux}$ is defined by Eq. (5), which is equal to 36 in the example, as shown in Fig. 2 (c1). Next, all found paths are deleted and $i_c$ is increased by one. The KPA scheme starts finding $k$ disjoint paths from the beginning again, as shown in Fig. 2 (c1). However, the scheme takes time to find the required set of $k$ node-disjoint paths because it avoids the paths with high cost and this situation leads to overlap with used paths, as shown in Fig. 2 (d3), (e3), (f3), (g3) and (h3).

Finally, this scheme finds a set of $k = 3$ node-disjoint paths, which are $\eta_1 (1-3-6-7), \eta_2 (1-2-5-7)$, and $\eta_3 (1-4-7)$, at $i_c = 8$, as shown in Fig. 2 (i3).

III. KPI: k-PENALTY WITH INITIAL ARC COSTS MATRIX

The KPI scheme is an extension of the KPA scheme. The KPI scheme uses the same penalty process as the KPA scheme, only the policy of updating $\xi^{aux,p}_h$ is different from the KPA scheme (Step 6a). The KPI scheme increases the costs $\xi^{aux,1}_h, \ldots, \xi^{aux,k}_h$ of each conflicting arc $a_h$ of $\eta_j$ by path cost $\xi^P$ of $\eta_j$ using initial costs matrix $\Xi^P$. Step 6a of
INPUT: Demand \( d_e \), to find the set of \( k \)-disjoint paths between a pair of nodes \((s_e, t_e)\). The initial arc costs matrices \( \Xi^1, \Xi^2, \ldots, \Xi^k \) (one matrix for each path) of a demand. The maximum allowable number of conflicts, \( i_{\text{max}} \).

OUTPUT: The set of \( k \)-disjoint paths \( \eta_1, \eta_2, \ldots, \eta_k \) all between a given pair of demand source and destination nodes \((s_e, t_e)\). The total path cost of \( k \)-disjoint paths is

\[
\xi_{\text{total}} = \sum_{p=1}^{k} \sum_{a_h \text{ on } \eta_p} \xi^p_{a_h}.
\]

**PROCESS**

1. **Step 1** Set \( i_e = 1 \) and \( \Xi^{a_{\text{aux}}, p} = \Xi^p \) for \( p = 1, \ldots, k \).
2. **Step 2** Set \( j = 1 \).
3. **Step 3** Set \( \Xi^{a_{\text{aux}}} = \Xi^{a_{\text{aux}}, j} \).
4. **Step 4** Consider each path \( \eta_j \) from the set of previously found \( j-1 \) paths and for each arc \( a_h \) of \( \eta_j \), if \( a_h \) is a forbidden arc of the path \( \eta_j \), then increase the arc cost \( \xi^p_{a_h} \) by path cost \( \xi^{a_{\text{aux}}, i} \) of \( \eta_j \) on the network with costs matrix \( \Xi^{a_{\text{aux}}, i} \). That is

\[
\xi^{a_{\text{aux}}, p}_{a_h} = \xi^{a_{\text{aux}}, p}_{a_h} + \xi^{a_{\text{aux}}, i}_{a_h}.
\]

The path cost is defined by

\[
\xi^{a_{\text{aux}}, i}_{a_h} = \sum_{a_h \text{ on } \eta_j} \xi^{a_{\text{aux}}, i}_{a_h} \text{ for } i = 1, \ldots, j - 1
\]

5. **Step 5** Find the shortest path \( \eta_j \) on the network with costs matrix \( \Xi^{a_{\text{aux}}, j} \).

6. **Step 6** If \( \eta_j \) is disjoint with the previously found \( j-1 \) paths then set \( j = j + 1 \) and go to Step 7.

6a) Increase the costs \( \xi^{a_{\text{aux}}, 1}_{a_h}, \ldots, \xi^{a_{\text{aux}}, k}_{a_h} \) of each conflicting arc \( a_h \) of \( \eta_j \) by path cost \( \xi^j \) of \( \eta_j \) on the network with costs matrix \( \Xi^{a_{\text{aux}}} \). That is

\[
\xi^{a_{\text{aux}}, p}_{a_h} = \xi^{a_{\text{aux}}, p}_{a_h} + \xi^j, \quad \text{when } p = 1, \ldots, k
\]

where path cost \( \xi^j \) is defined by

\[
\xi^j = \sum_{a_h \text{ on path } \eta_j} \xi^j_{a_h}
\]

and then delete the found paths and set \( i_e = i_e + 1 \).

6b) If \( i_e > i_{\text{max}} \) then terminate and reject the demand, else go to Step 2.

7. **Step 7** If \( j > k \) then terminate and return the found set of paths, else go to Step 3.

---

![Fig. 1. KPA scheme for finding \( k \) disjoint paths given demand \((s_e, t_e)\).](image)

The KPI scheme is as follows.

6a) Increase the cost \( \xi^{a_{\text{aux}}, 1}_{a_h}, \ldots, \xi^{a_{\text{aux}}, k}_{a_h} \) of each conflicting arc \( a_h \) of \( \eta_j \) by path cost \( \xi^j \) of \( \eta_j \) on the network with cost matrix \( \Xi^j \). That is

\[
\xi^{a_{\text{aux}}, p}_{a_h} = \xi^{a_{\text{aux}}, p}_{a_h} + \xi^j, \quad \text{when } p = 1, \ldots, k
\]

where path cost \( \xi^j \) is defined by

\[
\xi^j = \sum_{a_h \text{ on path } \eta_j} \xi^j_{a_h}
\]

and then delete the found paths and set \( i_e = i_e + 1 \).

---

![Fig. 2. Example of KPI scheme for the multi-cost network; the demand \( d_e = (1, 7) \) and \( k = 3 \).](image)

**A. Example of KPI scheme**

We reuse the example in demonstrating the KPI scheme, see Fig. 3. The algorithm starts finding the first path \( \eta_1 \) (1-4-7) by using the shortest-path-based algorithm, Fig. 3 (b1). Before finding path \( \eta_2 \), the cost \( \xi^{a_{\text{aux}}, 1}_{a_h} \) of arcs incident to transit nodes of path \( \eta_1 \) are increased by the total cost \( \xi^{a_{\text{aux}}, 1}_{a_h} \) of path \( \eta_1 \), which is equal to 14 in the example, Fig. 3 (b2). After that path \( \eta_2 \) (1-3-5-7) is found. The cost, \( \xi^{a_{\text{aux}}, 2}_{a_h} \), of arcs incident to transit nodes of paths \( \eta_1 \) and \( \eta_2 \) are increased by path cost \( \xi^{a_{\text{aux}}, 1}_{a_h} \) of path \( \eta_1 \) and path cost \( \xi^{a_{\text{aux}}, 2}_{a_h} \) of path \( \eta_2 \), respectively. However, \( \eta_3 \) (1-4-7), which is not disjoint with \( \eta_1 \) and has a common transit node, node 4, is found as shown in Fig. 3 (b3). Cost \( \xi^{a_{\text{aux}}, p}_{a_h} \) of arcs incident to node 4 for all paths, \( p = 1, \ldots, k \), are increased by path cost \( \xi^p \) computed from the initial arc costs of \( \eta_3 \) defined by Eq. (7), which is equal to 8 in Fig. 3 (c1). Next, all the found paths are deleted and \( i_e \) is increased by one. The algorithm starts from the beginning, as shown in Fig. 3 (c1). Finally, the scheme finds a set of \( k = 3 \) node-disjoint
paths, which are \( \eta_1 \) (1-2-5-7), \( \eta_2 \) (1-4-7), and \( \eta_3 \) (1-3-6-7), as shown in Fig. 3 (d3). Since the KPI scheme is more careful in increasing the costs of conflicting arcs, it can find a set of \( k = 3 \) node-disjoint paths at the conflict counter \( i_c \) value of 3 in the same way as the KPA scheme. This example shows that the KPI scheme can find a set of \( k \) node-disjoint paths faster than the KPA scheme.

![Diagram of paths](image)

Fig. 3. Example of KPI scheme for the multi-cost network; the demand \( d_v = (1,7) \) and \( k = 3 \).

IV. PERFORMANCE EVALUATION

We compare the KPI scheme performance to that of the KPA scheme using computer simulations of the Italian network and the U.S. long-distance network, see Fig. 4 [9]. The required number of disjoint paths was set to \( k = 3 \). Therefore, additional links, shown as dashed lines in both networks, were needed to keep the degree of each node greater than or equal to three. The arc cost matrices of the multi-cost networks were set to three for \( k = 3 \). 100 arc cost matrices for each corresponding disjoint path were generated uniformly in a random manner in the range of \( 0 < \xi_h \leq 1 \), where \( \xi_h \) is the cost of arc \( a_h \). For both KPI and KPA schemes, we examined the average probability that \( k \) disjoint paths were successfully found within a specified maximum allowable number of conflicts, \( i_{max} \), over all source and destination node pairs for all generated cost matrices. The probability is defined as the success ratio of finding \( k \) disjoint paths.

Figures 5 (a) and (b) show that the KPI scheme finds \( k \) disjoint paths faster than the KPA scheme for both the Italian network and U.S. long-distance network. Regardless of \( i_{max} \), KPI has a higher success ratio than KPA. In addition, the KPI scheme yields success ratios of more than 99% with \( i_{max} = 10 \) for both networks, while the KPA scheme does not reach 99% when \( i_{max} \) becomes large. In the KPA scheme, the conflict path cost defined in Eq. (5) is set at too large a value, the conflicting arcs are always avoided. On the other hand, as the KPI scheme defines the conflict path cost according to Eq. (7), the conflicting arcs are appropriately utilized.

![Diagram of networks](image)

Fig. 4. Italian network and U.S. long-distance network

The total cost of \( k \) disjoint paths, which is defined in Eq. (1), is lower with the KPI scheme than with the KPA scheme. Figure 6 compares the normalized total costs of \( k \) disjoint paths, normalized by the total path cost of \( k \) disjoint paths by Bhandari’s scheme. We used Bhandari’s scheme, which is a scheme for finding \( k \) disjoint paths in single-cost network, to find \( k \) disjoint paths using only the arc cost matrix for the first path. After \( k \) disjoint paths are found by Bhandari’s scheme, the total path cost in a multi-cost network is calculated by using Eq. (1) with three arc costs matrices. The normalized costs for Bhandari’s, KPA, and KPI are taken as average values over all source and destination node pairs for all generated cost matrices. The results indicate that the KPI scheme yields lower total path cost of than KPA or Bhandari’s scheme for both the Italian and the U.S. long-distance multi-cost network, as shown in Fig. 6. Bhandari’s scheme yields the highest path cost among the three schemes, as it considers only the arc cost matrix for the first path to find \( k \) disjoint paths. Since, in the KPI scheme, the conflicting path cost is suitably estimated and the conflicting arcs are appropriately utilized, it returns the lowest total path costs.

V. CONCLUSION

This paper proposed KPI, \( k \)-penalty with initial arc costs matrix, to find \( k \) disjoint paths in a multi-cost network. The KPI scheme uses an initial arc cost matrix in estimating the conflicting path cost. The KPI scheme increases the costs of conflicting arcs in finding \( k \) disjoint paths. Numerical results indicate that the KPI scheme is able to find \( k \) disjoint paths faster and with lower total path cost than the KPA scheme. The KPI is an efficient scheme to find disjoint paths, which enhance the survivability of network. The deviation of the KPI scheme is not large, but the impact on overall performance is significant.
Fig. 5. Successful ratio of finding $k$ disjoint paths (%) within specified maximum allowable number of conflicts, $i_{\text{max}}$, on (a) Italian network and (b) U.S. long-distance network.

Fig. 6. Normalized summation of $k$ disjoint paths costs on Italian network and U.S. long-distance network by using Bhandari's, KPA and KPI scheme.

REFERENCES


[12] J. Rak, and W. Molisz, “A new approach to provide the differentiated levels of network survivability under a double node failure,” 11th Int. Conf. on Transparent Optical Networks (ICTON’09), pp. 1-4, Jul. 2009.