# Addressing Network Survivability Issues by Finding the K-best Paths through a Trellis Graph

Stavros D. Nikolopoulos*, Andreas Pitsillides* and David Tipper+

*Department of Computer Science, University of Cyprus, CY-1678 Nicosia, Cyprus.
e-mail: {stavros, cspitsil}@turing.cs.ucy.ac.cy
+Department of Information Science and Telecommunications,
University of Pittsburgh, Pittsburgh, PA 15260
email: tipper@tele.pitt.edu

***Abstract***

*Due to the increasing reliance of our society on the timely and reliable transfer of large quantities of information (such as voice, data, and video) across high speed communication networks, it is becoming important for a network to offer survivability, or at least graceful degradation, in the event of network failure. In this paper we aim to offer a solution in the selection of the K-best disjoint paths through a network by using graph theoretic techniques. The basic approach is to map an arbitrary network graph into a trellis graph which allows the application of computationally efficient methods to find disjoint paths. Use of the knowledge of the K-best disjoint paths for improving the survivability of ATM networks at the virtual path and virtual circuit level is discussed.*

*Keywords: Network management, Network survivability techniques, Graph theory, Trellis graph.*

## 1. Introduction

Due to the increasing reliance of society on the timely and reliable transfer of large quantities of information (such as voice, data, and video) across high speed communication networks, it is becoming important for networks to provide adequate service in the event of a failure. A network failure, such as the loss of a link or a node, can occur due to a variety of reasons causing service disruptions ranging in length from seconds to weeks. Typical events that cause failures are accidental cable cuts, hardware malfunctions, software errors, natural disasters (e.g., fire), and human error (e.g., incorrect maintenance). Since many of the causes of failures are beyond the control of the network providers, there has been increasing interest in the design of survivable networks. Survivability can be defined as network design and management procedures to minimize the impact of failures on the network. Survivability techniques can be classified into three categories: 1) prevention, 2) network design, and 3) traffic management and restoration. Prevention techniques focus primarily on improving component and system reliability. Some examples are the use of fault-tolerant hardware architectures in switch design, provision for backup power supplies, predeployment stress testing of software, use of frequency hopped spread spectrum techniques to prevent jamming in military radio networks and so on. Network design techniques try to mitigate the effects of system level failures such as link or node failures by placing sufficient diversity and capacity in the network topology. For example, the use of multi-homing nodes so that a single link failure cannot isolate a network node or an access network. Traffic management and restoration procedures seek to direct the network load such that a failure has minimum impact when it occurs and that connections affected by a failure are reconnected around the failure.

Survivability goals may be accomplished by designing network infrastructures that are robust to malfunctions of nodes and links, and implementing network control systems that are inherently fault-tolerant and self-healing. In recent years there has been significant work on survivability for circuit switched voice networks and the underlying physical transmission network.



0-8186-7780-5/97 $10.00 

In this paper we aim to use graph theory to select the K-best disjoint paths (or unmerged paths) between any Origin-Destination (OD) pair. The selection of the K-best disjoint paths can take into account many factors, such as selection of the shortest paths (hence minimizing delay), minimization of the bandwidth allocation (given the bandwidth demanded by customers), and maximization of network throughput. "Best" paths are those paths which are as diverse as possible (i.e. if the network topology allows there are K disjoint paths), and therefore will maximize our chances of survivability, or ensure at least a graceful degradation, (i.e. display fault tolerance) in the event of a network fault.

Here we concentrate on the selection of the K-best paths which can be used for load balancing or rerouting, using the shortest path as the minimization criterion. We provide an algorithm that transforms a given network into a trellis graph, and consequently, we use the algorithm of [6] for finding the K-best paths through the trellis. This algorithm is based on a transformation of the K-best path trellis problem into an equivalent *Minimum Cost Network Flow* (MCNF) problem.

The majority of published work concentrates on the k-shortest link disjoint path problem, e.g. [1, 2, 3, 4], rather than the K-best paths. A number of these algorithms are based on the iteration of Dijkstra's shortest path algorithm to find restoration paths for the failed links via surviving spare links on other spans of the network (referred to as the k-successively shortest link disjoint path algorithm in [3]). Note that once a restoration path is found, the spare links which make it up are removed from the network description and the algorithm is run again until it fails to find any additional paths. Examples include: [3] which addresses span restoration rather than path restoration; and [2] and [4] which are based on matrix and recursive matrix calculations respectively to improve computational complexity. Note that these methods are not strictly optimal in terms of finding the maximal number of paths in all possible networks, and may underestimate the number of paths whenever the k-successively shortest link disjoint path algorithm selects a path which blocks other potential paths (well illustrated in [3] using the generalised "trap" topology), or even worse they may overestimate the number of link disjoint paths (e.g. [2]). On the other hand, [5, 6, 7] concentrate on finding only a pair of disjoint paths between a given pair of nodes, by optimizing the physical length of paths: [5] finds the shortest pair of node-disjoint paths but cannot be applied at the span (physical) level (e.g. physical links sharing a common conduit); [6] finds a pair of disjoint paths between a given pair of nodes taking into consideration any span sharing by links, however the solution for networks with arbitrary connection patterns is not given; and [7] presents a heuristic approach (not necessarily optimal) to find in polynomial time a pair of paths which is as diverse as possible, taking into account any common spans.

In this paper, we formulate the network survivability problem in terms of a number of transformations leading to a trellis graph. A trellis graph is a structured graph offering several advantages in formulating many problems of diverse fields such as radar, sonar, radioasronomy, etc. [8, 9, 10, 11]. Using the trellis graph we seek to find the K-best disjoint paths from an origin node to a destination node. If there is a cutpoint (or articulation point), then there are no disjoint paths [12]. Therefore we seek the next best solution (not necessarily optimal), that is a solution which provides K-best paths which are as diverse as possible (i.e. with minimal overlap between the paths in terms of nodes and links). To this effect we run a similar algorithm, briefly described in section 3. Our algorithm offers several advantages, in comparison to existing: can find the K-best paths (rather than the K-shortest paths, or the pair of shortest paths; computes the K-best disjoint paths with respect to link cost (instead of just the hop count); if a particular OD pair contains m disjoint paths our algorithm can exactly compute them (i.e. does not under or over estimate).

The remainder of the paper is organized as follows. Section 2 describes the problem formulation, and section 3 constructs the algorithm. In section 4 we offer our conclusions together with some recommendations for future work.

## 2. Formulation of Survivability Issue

We introduce some graph theoretic concepts and we establish the notation and terminology we shall use throughout this paper, and apply these concepts in the network survivability problem (see also [13]).

### 2.1. Graph Modelling

A *directed graph* G = (V, E) is a structure consisting of a finite set of nodes $V = \{v_1, v_2, ..., v_n\}$ and a finite set of links $E = \{(v_i, v_j) \mid v_i, v_j \in V \text{ and } v_i \neq v_j\}$, where each link is an ordered pair.

We define a *trellis* as a directed graph G = (V, E) with nodes and directed links that satisfies the following conditions:

(i) The node set V is partitioned into L (mutually disjoint) subsets $V_1, V_2, ..., V_L$, such that $|V_i| = |V_j| = H$, $1 \le i, j \le L$.

(ii) Links connect nodes only of consecutive subsets $V_\ell$ and $V_{\ell+1}$, i.e., if $(v_i, v_j) \in E$, then $v_i \in V_\ell$ and $v_j \in V_{\ell+1}$, $1 \le t < L$.

The magnitude T we shall call *depth* of the trellis. A *K-trellis* is a trellis graph with two additional properties:

(i) It has two more nodes $s \in V_0$ and $t \in V_{L+1}$, such that $(s, v_i) \in E$, for every $v_i \in V_1$ and $(v_j, t) \in E$, for every $v_i \in V_L$, $1 \le i, j \le H$.

(ii) The node $v_i$ of the set $V_\ell$ is connected (where possible) with K = 2g+1 nodes $\{v_{i-g}, ..., v_i, ..., v_{i+g}\}$ of the set $V_{\ell+1}$, where $1 \le i \le H$, $1 \le t < L$ and $g = 1, 2, ..., (H-1)/2$.

The depth of a K-trellis graph will be equal to L+2. In Fig. 1 a K-trellis is presented with L = 4, H = 4 and K = 3. Throughout the paper, we shall refer to a K-trellis graph with K = H as *trellis graph.*

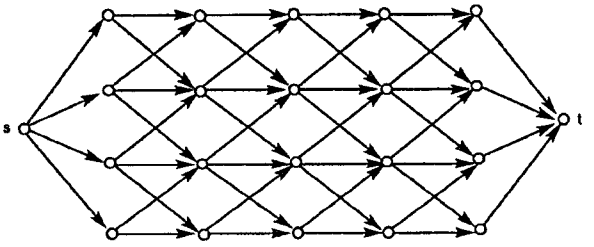



**Fig. 1. A K-trellis graph with L = 5, H = 4 and K = 3.**

A *walk* in a trellis is an alternating sequence of nodes and links, i.e., $P = [v_1, (v_1,v_2), v_2, ..., (v_{k-1},v_k), v_k]$. The *length L(P)* of a walk is the number of links in it. A *path* is a walk in which all nodes are distinct. For simplicity, we shall denote the path P by $P = \{v_1, v_2, ..., v_k\}$ and we shall refer to $v_1$ and $v_k$ as *first* and *last* nodes of P, respectively (i.e. the origin and destination nodes).

## 2.2. Link and Path Cost

In addition to the above defined link weights and metrics, in each link $(v_i, v_j) \in E$ of a trellis graph, $v_i \in V_\ell$ and $v_j \in V_{\ell+1}$, we associate a third number, which we call *link cost* and denote by $c_\ell(v_i, v_j)$ or $c_\ell(i, j)$ or $c(i, j)$, $1 \le t \le L$ and $1 \le i, j \le H$. The cost of each link is given as a function of the node and link metrics, denoted Q and D respectively, i.e.,

$$c(v_i, v_j) = f(Q(v_i), D(v_i, v_j)),$$
$$\forall\, v_i \in V_\ell \text{ and } \forall\, (v_i, v_j) \in E, \quad \ell = 1, 2, ..., L.$$

Let $P = \{v_1, v_2, ..., v_k\}$ be a path in a trellis graph. The *cost* c(P) of a path P through the trellis is defined as

$$c(P) = \sum_{(i,j) \in P} c(v_i, v_j)$$

where $c(v_i, v_j)$ is the cost of link $(i, j) \in E$. The *shortest path* from the node $v_i$ to node $v_j$ is a path $P = \{v_i, v_{i+1}, ..., v_j\}$ with minimum cost.

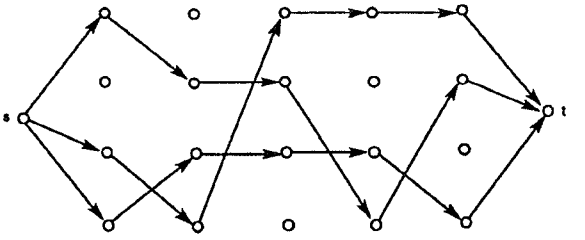



**Fig. 2. Illustration of three mutually disjoint paths on a K-trellis with L = 5, H = 4 and K = 4.**

Let G = (V, E) be a trellis graph with L H nodes, i.e., $|V| = LH$. Two paths $P_i = \{s, u_1, u_2, ..., u_L, t\}$ and $P_j = \{s, u'_1, u'_2, ..., u'_L, t\}$ are said to be *mutually exclusive* or *unmerged* if $u_\ell \neq u'_\ell$ for all $\ell = 1, 2, ..., L$; otherwise, they are said to be *merged.* Hereafter, we shall refer to unmerged paths as *disjoint* paths (see Fig. 2).

## 2.3. Finding the K-best Paths

With the above notation, the problem of finding the K best paths in a trellis graph is stated as follows:

**Problem**: Find K paths $P_1, P_2, ..., P_K$ through the trellis G = (V, E) which minimize the total cost

$$J = \sum_{i=1}^{K} c(P_i)$$

subject to the constraints that the paths $P_1, P_2, ..., P_K$ are mutually disjoint. Figure 2 illustrates this issue.

Constanon in [14] showed that the problem of finding the K-best paths through a trellis graph can be defined as a *Minimum Cost Network Flow* (MCNF)

problem. He also showed that the worst-case computation time for this problem is bounded by $n^3 \log n$, where n is the number of nodes in the trellis. His $O(n^3 \log n)$ time algorithms are much faster than those proposed in [15]. Note that the computational complexity of our algorithm is currently under investigation, and will be reported elsewhere.

It is important to note that, for $K \geq 2$ the best set of K paths is not found in general by finding the best path, then the next best path that is completely disjoint with the best path, etc (as in the k-succesively shortest link disjoint paths [3]). The reason for this is that these paths are not independent of one another because of the requirement that the paths be completely disjoint so that it might be better to use one or more of the branches that are part of the best single path for other paths [15].

## 3. Problem Transformation

We have formulated the path selection and rerouting problem in addressing network survivability as a graph theoretic, K-best path, problem with an efficient minimum cost network flow solution.

For any given network topology, our objective is to transform it to a trellis graph. Toward this end, we first define a node partition of an undirected graph according to distance between its nodes.

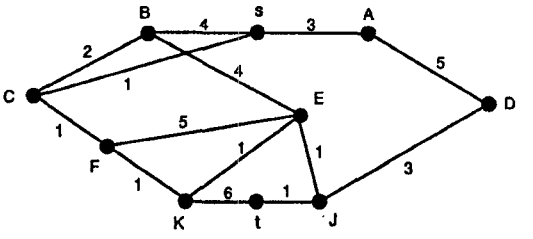
Given a graph G = (V, E) and a node $v \in V$, we define a partition $\mathcal{L}(G, v)$ of the node set V (we shall frequently use the term *partition of the graph* G), with respect to the node v as follows:

$$\mathcal{L}(G, v) = \{ AL(v, \ell) \mid v \in V, 0 \leq \ell \leq L_V, 1 \leq L_V < |V| \}$$

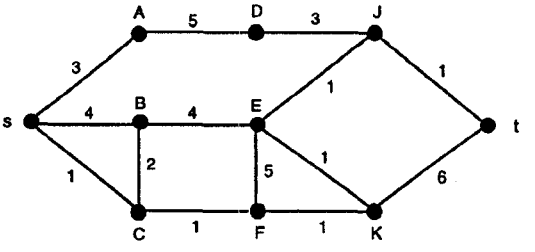
where $AL(v, \ell)$, $0 \leq \ell \leq L_V$, are the *adjacency-level sets*, or simply the *adjacency-levels*, and $L_V$ is the *length* of the partition $\mathcal{L}(G, v)$. The adjacency-level sets of the partition $\mathcal{L}(G, v)$, are defined as follows:

$$AL(v, \ell) = \{u \mid d(v, u) = \ell, \ 0 \leq \ell \leq L_V\}$$

where d(v, u) denotes the *distance* between nodes v and w in G. We point out that $d(v, u) \geq 0$, and $d(v, u) = 0$ when $v = u$, for every $v, u \in V$. Thus, the adjacency-level sets of the graph of fig.3.a, with respect to node s are shown in fig.3.b, i.e., AL(s, 0) = {s}, AL(s, 1) = {A, B, C}, AL(s, 2) = {D, E, F}, AL(s, 3) = {J, K} and AL(s, 4) = {t}. Some properties of these sets appear in [16].




(a)




(b)

**Fig. 3. (a) A weighted undirected graph G = (V, E) or a network (all weights are positive integer numbers); (b) Partition of the graph G, with respect to vertices s.**

Having defined the partition of a graph, or network, into adjacency-level sets, let us now describe the algorithm which transforms a given network topology to a trellis graph. For ease of exposition, we will use a particular example network (see fig. 3.a) to illustrate the transformation process and then generalize our approach.

Consider the network topology G = (V, E, c) given in fig. 3.a, where c is the *cost function* from the link set E to real numbers.

Step 1 The first step in the transformation process toward the trellis graph is to partition, with respect to a particular node, the node set of the network into adjacency-levels (see fig.3.b). By definition, this partition places the nodes of the network at "vertical" levels according to their distances. (Starting at the origin node of a given OD pair, we label the starting node as s, and the last as t.)

step 2 In step 2, we disconnect the network into two (sub)networks G' and G". Network G' contains all the nodes and links except the destination node t and the links incident to it (in graph-theoretic terms, it is the graph induced by the node set V-{t}). Network G" contains the nodes in set $\{t\} \cup N(t)$ and all the links with end-nodes t and x, where $x \in N(t)$. (We

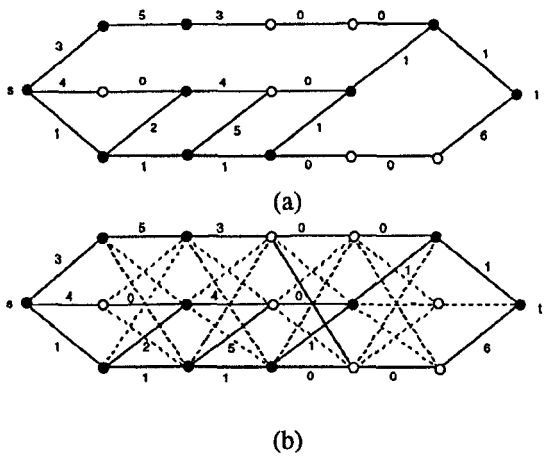
illustrate the details of this step through an example shown in the appendix.)

step 3 If now after step 1 and 2, we have any "vertical" links in G', i.e., two nodes connected by a link belonging to the same level (which our trellis model does not support), we apply two specific operations which eliminate "vertical" links. These operations are based on the addition of dummy nodes (indicated by empty circles; see fig. 4.a) and 0-cost links, in such a way that the path cost is preserved. The details of these operations are defined later.

step 4 In this step, we merge the resulting graph G' from step 3 with the graph G" by adding (if necessary) dummy nodes and 0-cost links. The resultant graph for this example is shown in fig. 4.a.

step 5 Now, if necessary, to complete the trellis graph, we introduce more dummy nodes, but with infinite link cost (shown as dotted lines; see fig. 4.b).

The resultant graph, which results in the complete trellis graph is shown in fig. 4.b. This algorithm (step 1 to 5) can be repeated for all OD pairs.



(a)

(b)

**Fig. 4. Illustration of transformation of the network of fig. 3.b into a trellis graph.**

Obviously, the crucial step of the transformation algorithm is step 3 where two specific operations are applied in the network. We next give the definition for the shortest path from node s to a node y, through a specific link, which is necessary for the definition of these operations.

*Definition*: Let x, y be two nodes of consecutive levels which are connected by a link. The *s-cost* of link (x, y) is defined to be the minimum cost of the path from s to y through node x, i.e., the cost of the path P = {s, ..., x, y}, and is denoted by $\phi(x, y)$.

**Operation P1**: Let G(V, E, c) be a network partitioned into adjacency-levels and let (x, y) ∈ E be a link, where nodes x, y are on the same level $\ell$, i.e., x, y ∈ AL(s, $\ell$). Let x be the node satisfying the following properties:

$$\min\{\phi(x, v) \mid v \in AL(s, \ell-1)\} > \min\{\phi(y, u) \mid u \in AL(s, \ell-1)\}$$

or

$$\min\{\phi(x, v) \mid v \in AL(s, \ell-1)\} = \min\{\phi(y, u) \mid u \in AL(s, \ell-1)\}$$

$$\sum_{v \in N(x) \cap AL(s,\ell-1)} \phi(x,v) \geq \sum_{u \in N(y) \cap AL(s,\ell-1)} \phi(y,u)$$

Then, replace node x with a dummy node x', move node x into level $\ell$ +1 and update the following parameters:

$w(x, x') = 0$

$\phi(x, x') = \min\{\phi(x', v) \mid v \in AL(s, \ell-1)\}$

$\phi(x, y) = \min\{\phi(x', v) \mid v \in AL(s, \ell-1)\} + c(x, y)$ □

Fig. 5 shows consecutive adjacency-levels of a partitioned network before and after the application of the Operation P1. Integer numbers indicate link cost, while integers in parenthesis indicate link s-cost. It is pointed out that before the application of the Operation P1 the link "vertical" (x, y) has only link cost, while after the operation it gains link s-cost.

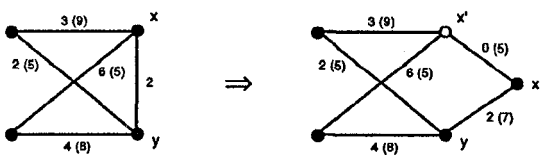



**Fig. 5. Illustration of the application of the Operation P1.**

**Operation P2**: Let G(V, E, c) be a network and let x be a node at level $\ell$ which, after operation P1, remains without neighborhoods in level $\ell$-1, i.e.,

$$N(x) \cap AL(s, \ell-1) = \varnothing.$$

Then, move node x into level $\ell+1$ and update the following parameter:

$$\phi(x, y) = \min\{\phi(y, v) \mid v \in AL(s, \ell-1)\} + c(x, y) \quad \square$$

Fig. 6 shows the resulting network topology when Operation P2 is applied to node x. Both links (z, x) and (y, x) have now link s-cost.

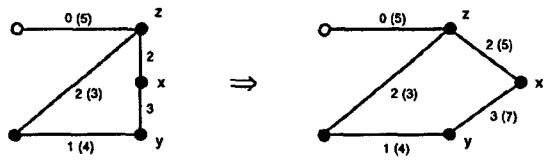



**Fig. 6. Illustration of the application of the Operation P2.**

Next we give a more formal listing of the previously described algorithm which transfers a given network topology to a trellis graph.

**Algorithm** Net_to_Trellis
**begin**

1. **Partition** the node set of the network G(V, E, c) into adjacent-level sets, with respect to origin node $s \in V$. That is, the nodes of the network are placed at levels according to their distances from s; the origin node s is placed at level 0;

2. **Disconnect** the network into two (sub)networks G' and G", where G' contains all the nodes and links except the destination node t and the links incident to it, while G" contains the nodes in set $\{t\} \cup N(t)$ and all the links of the form (x, t), where $x \in N(t)$.

3. **Apply** the operations P1 and P2 in the network G' in order to eliminate all the "vertical" links of the network G';

4. **Merge** the resulting network from step 3 with the network G" by adding (if necessary) dummy nodes and 0-cost links;

5. **Complete** the trellis structure by adding $\infty$-cost links;

**end.**

It is worth noting that as our main objective is to enhance network survivability, we seek K routes which are as diverse as possible. This can be achieved by selecting the K-best (disjoint) paths through the network. The term "K-best-paths" in our approach does not necessarily imply the "K-shortest-paths". This is natural, since we seek the K-best disjoint paths. Our example network (see fig. 3) illustrates this point. If we consider the shortest path, then there is no other disjoint path from s to t. Therefore, in order to find 2 disjoint paths it is obvious that the shortest path cannot be one of the 2 disjoint paths. This is easily illustrated in fig. 3 where the shortest path from s to t has cost 6, while the costs of the 2-best (disjoint) paths are 9 and 10 (contrast with the k-successively shortest link disjoint paths method [3], which selects the shortest path first–thus after removing from the network description the links selected in the shortest path, no links connecting to t remain, therefore the algorithm terminates with only one path found).

**Remark**: after step 1, we can easily check whether any disjoint paths for an OD pair exist. The process is fairly straight forward; it merely checks whether there is a cutpoint [11, 12]. If there are no disjoint paths, then we must find the K-best paths which are as diverse as the network topology allows (i.e., minimize any sharing of resources between the paths). Toward this end our initial thoughts follow: Break up the trellis graph at the cutpoint to form two trellis subgraphs. One from s till the cutpoint, and another one from the cutpoint till t. Find the K-best paths for each trellis subgraph, and then concatenate to obtain the K-best paths for the complete graph. The K-best paths now are not disjoint (since they share the node at the cutpoint), but at least from the path leading to the cutpoint, and from the cutpoint leading to the destination node, the K-best paths are disjoint (of course assuming that there are no further cutpoints).

## 4. Conclusions

In this paper, we use graph theoretic techniques to address network survivability issues by finding the K-best (disjoint) paths through a trellis graph (as compared with the majority of the literature which addresses the K-successively shortest link disjoint paths). Given any network topology we develop an algorithm which transforms the network into a trellis graph. Using the trellis graph we can find the K-best (node disjoint) paths for any Origin-Destination pair in a given network topology. Our algorithm offers many advantages (discussed in the introduction) in comparison to existing ones.

Knowledge of the K-best paths can be used in the design of survivable networks. Several questions regarding this work still exist. For example, how to best

use the knowledge of the K-best paths in a real network, where dynamic (re)routing, taking into account user (such as quality of service) and network (such as maximization of the throughput) requirements can offer substantial benefits.

## APPENDIX

In order to illustrate the workings of algorithm Net-to-Trellis, we present with a help of an example the processes of transforming a network G(V, E, c) into a trellis graph. The integer numbers indicate link cost, while integers in parenthesis indicate link s-cost. Here, vertices are named with capital letters.

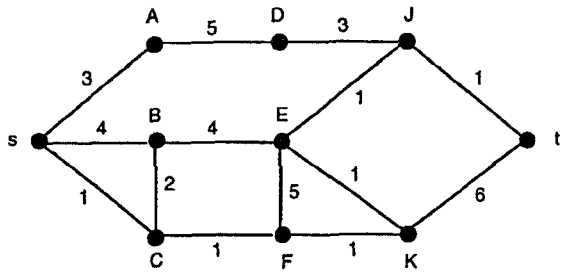



**Figure A1: A network G(V, E, c) used to illustrate the Graph-to-Trellis transformation (it is partitioned into adjacency-level sets; see section 3); Step 1 of the algorithm.**

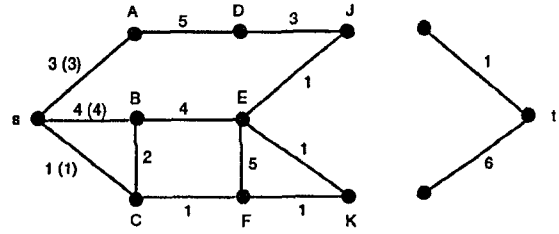



**Figure A2: The network G(V, E, c) is split into two (sub)networks G' and G"; Step 2 of the algorithm.**

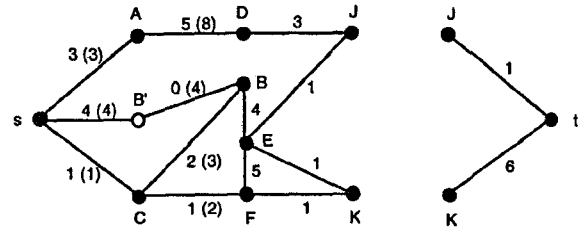



**Figure A3: The network after application of the Operation P1 on link (B, C); Step 3 of the algorithm.**

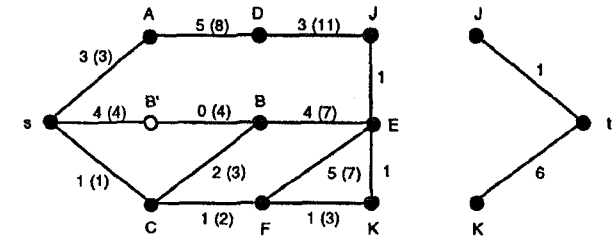



**Figure A4: The network after application of the Operation P2 on node E; Step 3 of the algorithm.**

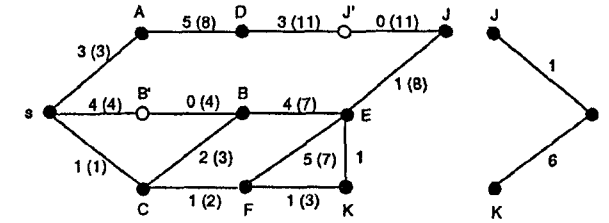



**Figure A5: The network after application of the Operation P1 on link (E, J); Step 3 of the algorithm.**

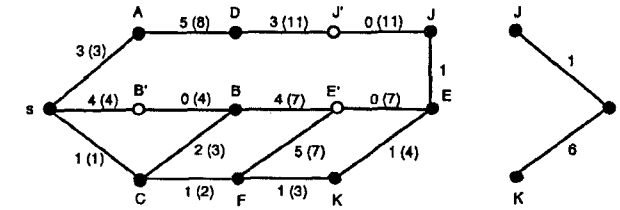



**Figure A6: The network after application of the Operation P1 on link (E, K); Step 3 of the algorithm.**

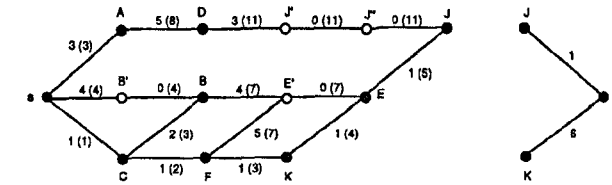



**Figure A7: The network after application of the Operation P1 on link (E, J); Step 3 of the algorithm.**

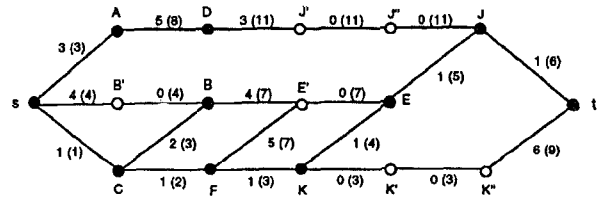



**Figure A8: The two networks G' and G" are connected into one; Step 4 of the algorithm.**

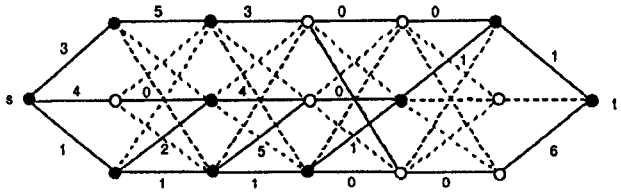



**Figure A9: The transformed network; Step 5 of the algorithm.**

# References


[1]. J.W. Suurballe, "Disjoint paths in a network", *Networks* 4, pp. 125-145, 1974.

[2] Y. Tanaka, F. Rui-xue, M. Akiyama, "Design method of highly reliable communication network by the use of the matrix calculation", IEICE Transactions Communications, vol.J70-B, no.5, pp. 551-556, 1987.

[3] D.A. Dunn, W.D. Grover, M.H. MacGregor, "Comparison og k-shortest paths and maximum flow routing for network facility restoration", IEEE Journal of Selected Areas in Communications, vol. 12, no. 1, pp. 88-99, January 1994.

[4] E. Oki, N. Yamanaka, "A recursive matrxi-calculation method for disjoint path search with hop link number constraints", IEICE Transactions Communications, vol. E78-B, no.5, pp. 769-774, May 1995

[5] J.W. Suurballe, R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths", *Networks* 14, pp. 325-336, 1984.

[6] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks", IEEE INFOCOM'94, Ontario, Canada, pp. 1498-1508, June 1994.

[7] S.Z. Shaikh, "Span-disjoint paths for physical diversity in networks", IEEE Symposium on Computers and Communications, Alexandria, Egypt, pp. 127-133, June 27-29, 1995.

[8] R.E. Bellman and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton (1962).

[9] W.G. Bliss and L.L. Scharf, "Algorithms and Architectures for Dynamic Programming on Markov Chains", *IEEE Trans.* ASSP **37**, pp. 900-912, 1989.

[10] S.D. Nikolopoulos, "Track Detection using Graph Theoretic Concepts", IN-669, SACLANT Undersea Research Centre, La Spezia, Italy, 1990.

[11] S.D. Nikolopoulos and G. Samaras, "Sob-optimal Approach to Track Detection for Real-time Systems", 21st Euromicro Conference on Design of Hardware and Software Systems, IEEE/CS, Como, Italy, pp. 98-107, Sept. 4-7, 1995.

[12] F. Harary, *Graph Theory*, Addison-Wesley, Reading (1969).

[13] S.D. Nikolopoulos and A. Pitsillides, "Towards Network Survivability by Finding the K-best Paths through a Trellis Graph", International Conference on Telecommunications (ICT'96), Istanbul, Turkiye, pp. 817-821, April 1996.

[14] D. Castanon, "Efficient Algorithms for Finding the K Best Paths Through a Trellis", *IEEE Trans.* AES **26**, pp. 405-410, 1990.

[15] J.K. Wolf, A.M. Viterbi and G.S. Dixon, "Finding the Best Set of K Paths Through a Trellis with Applications to Multitarget Tracking", *IEEE Trans.* AES **25**, pp. 287-296, 1989.

[16] S.D. Nikolopoulos, "Parallel Block-finding using Distance Matrices", *Journal of Parallel Algorithms and Applications* **9**, pp. 1-13, 1996.