International Journal of Performability Engineering Vol. 11, No. 3, May 2015, pp.265-274.
© RAMS Consultants
Printed in India

Bi-Objective Topology Design of Communication Networks Using Dynamic Programming

BASIMA ELSHQEIRAT[1], SIETENG SOH*[2], SURESH RAI[3], and MIHAI LAZARESCU[2]

[1]*University of Jordan, Amman, Jordan.*
[2]*Curtin University, Perth 6845, Western Australia, AUSTRALIA.*
[3]*Louisiana State University, Baton Rouge, 70803 LA, USA.*

(Received on September 10, 2014, revised on February 16, and February 25, 2015)

Abstract: This paper provides an algorithm to design a communication network topology with minimal cost (C) and maximum (*s, t*) reliability (R) subject to a pre-defined bandwidth (B_{min}) constraint, given (i) locations of the various computer nodes, (ii) their connecting links, (iii) each link's reliability, cost and bandwidth, and (iv) a minimum bandwidth for the network. The bi-objective optimization problem is known NP-hard; henceforth we call the problem NTD-CR/B. We use the *dynamic programming* (DP) formulation as the engine in our proposed approach, DPCR/B, to generate the topology for solving NTD-CR/B. Further, we propose three greedy heuristics that enumerate and order only $k \leq n$ (*s, t*) paths, where *n* is the total number of (*s, t*) paths in the network. Each heuristic allows DPCR/B to obtain the selected paths to form the topology using only *k* paths, which improves the time complexity while producing near optimal topology. Extensive simulations on large networks with various sizes show that DPCR/B is able to generate 91.2% optimal results while using only 1.37^{-27}% of all paths in the grid networks that typically contain up to 2^{99} paths.

Keywords: *Dynamic programming; Lagrange relaxation*; *bi-objective optimization; bandwidth, reliability; network topology design.*

1. Introduction

Many applications require some network Quality of Service (QoS) constraints such as reliability, delay, and/or bandwidth to be operational. In particular, applications for critical time systems, *e.g*., rescue and military application must run without interruption, even in the presence of network component failures [1]. In addition, maximizing network communication bandwidth is equally crucial for many bandwidth-intensive applications, *e.g*., video-on-demand [1]. Thus, optimized network topology design (NTD) is crucial because optimal topology affects network's QoS. However, constructing a network with higher QoS incurs higher installation cost [2]. This paper focuses on an NP-hard problem, called NTD-CR/B, to design networks with minimum cost and maximum reliability subject to satisfying a required operational bandwidth. Specifically, given a set of nodes, their possible connecting links, link failure rate, bandwidth and installation cost, the problem selects the best set of links such that the resulting topology meets its required bandwidth B_{min} while minimizing its installation cost and maximizing its reliability. We consider 2-terminal reliability [3]. NTD-CR/B is NP-hard [4] since computing network reliability in general is NP-hard [5]. Thus heuristic solutions are required to design large sized networks.

Our main contribution is two folds: (i) to propose an efficient dynamic programming (DP) approach, called DPCR/B, to solve NTD-CR/B; (ii) to propose three heuristic path

[1]This work was conducted when the author was at Curtin University
*Corresponding author's email: S.Soh@curtin.edu.au

orders to enumerate only $k \leq n$ paths to reduce the time complexity of DPCR/B; n is the total number of (s, t) paths in the network. Using the path orders, DPCR/B requires only 1.37^{-27}% of all paths in grid networks that contain up to 2^{99} paths while generating 91.2% optimal results, with 11 non-optimal results within 8.8% off from optimal.

The layout of this paper is as follows. Section 2 shows related works. Section 3 discusses network model and notations. Section 4 formulates NTD-CR/B problem. Section 5 describes proposed solution and illustrating example. Section 6 presents simulation results. Finally, Section 7 concludes the paper and discusses future work.

2. Related Work

NTD is a known NP-hard problem [4]. The single objective (SO) versions of the problem that considers only one objective, *e.g.*, maximizing reliability, for a given constraint, *e.g.*, cost, are well studied [2], [3]. In contrast, its multiple objective (MO) versions that consider two or more objectives for a given constraint have not been addressed as much; we call MO that considers two objectives is bi-objective (BO).

Local search techniques have been frequently used to solve NTD problems. However, these techniques generally do not perform well when MO needs to be optimized and/or constraints are present [6]. Kumar *et al.* [7] presented a genetic algorithm (GA), called Pareto Converging Genetic Algorithm (PCGA), to design a network that simultaneously optimizes network delay and costs under reliability and bandwidth constraint. Banerjee and Kumar [8] studied MO network design using a GA heuristic and empirically showed that the GA generally provides better solution than its deterministic counterparts. Papagianni *et al.* [9] used the particle swarm (PS) optimization to solve MO network design problem. They [9] claimed that the approach is more effective than the GA in [8] but the algorithm was only tested on a 16-node network and nothing was said about its efficiency. Duarte and Bar´an [10] proposed a parallel GA for solving NTD problems with cost and reliability as objectives, and some constraints other than bandwidth.

While the meta-heuristic based algorithms, *e.g.*, GA and PS, may reduce time complexity, they still require numerous iterations to converge and thus use a considerable computational effort. Thus, a more time efficient heuristic is still needed, especially for use in large networks. Recently, the authors in [2], [3] have used a dynamic programming (DP) to solve two related problems, NTD-RC (NTD-CR) that aim to design a topology with maximum reliability subject to a cost constraint (minimum cost subject to a reliability constraint). The authors [2], [3] showed the suitability of using recursive heuristics such as DP to solve the problems because it has a tendency to escape a local optimum while oftenly finding a global optimum solution in a reasonable time. This paper uses a similar DP solution, called DPCR/B, to solve NTD-CR/B, defined in Section 4.

3. Network Model and Notation

A communication network (CN) can be modeled by a probabilistic undirected simple graph G=(V, E), in which each vertex $v_i \in V$ represents a computing node and each link $e_j \in E$ represents the connecting media between two nodes. All node locations and connecting links are known. We consider each node has sufficient fault-tolerance and backup components, allowing the node to continue its operation when there is any component failure. Thus, the paper assumes all nodes are always functioning. Each link e_j has a cost $c_j>0$, reliability $0 \leq r_j \leq 1.0$, and capacity $b_j>0$ to respectively represent its installation cost, functioning probability, and bandwidth. Link failures are statistically independent and without repair. This assumption is used to reduce the combinatorial size of the already difficult problems [11]. Figure 1 shows an example of CN with six fixedly positioned nodes, and eight links; the figure also shows a tuple (c_j, r_j, b_j) for each e_j.

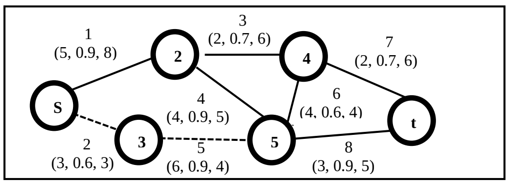

Figure 1: An Example CN

Table 1: Path Set for Network of Figure 1

P_i		Rel(P_i)	Cost(P_i)	BW(P_i)
P_1	(2,5,4,3,7)	0.24	17	3
P_2	(1,4,6,7)	0.34	15	4
P_3	(1,3,6,8)	0.34	14	4
P_4	(2,5,6,7)	0.23	15	3
P_5	(1,4,8)	0.73	12	5
P_6	(2,5,8)	0.49	12	3
P_7	(1,3,7)	0.44	9	6

A (*s*, *t*) simple path *i*, P_i, is a sequence of links from node *s* to node *t* such that no node is traversed more than once. Let L_i be the set of links in P_i, P_G be a set of all paths in G and n=|P_G|; for Fig. 1, P_G=(P_1=(2,5,4,3,7), P_2=(1,4,6,7), P_3=(1,3,6,8), P_4=(2,5,6,7), P_5=(1,4,8), P_6=(2,5,8), P_7=(1,3,7)). Let Cost(P_i) denote the cost of installing all links in path P_i, calculated by taking the sum of c_j of each $e_j \in P_i$; *e.g.*, Cost(P_1)=3+6+4+2+2=17. The cost of a network G, Cost(G), is calculated by taking the sum of all c_j for each $e_j \in$G; for Fig. 1, Cost(G)=29. Let Rel(P_i) denote the reliability of path P_i, calculated by multiplying all r_j of each $e_j \in P_i$; *e.g.*, Rel (P_1)=0.6×0.9×0.9×0.7×0.7=0.24. The 2-terminal reliability of a topology G, Rel(G), is the probability that at least one $P_i \in P_G$ is functional. Calculating Rel(G), in general, is NP-hard [5]; for Fig. 1, Rel(G)=0.902. Let BW(P_i) denote the bandwidth of path P_i, determined by the bottleneck link in P_i; *e.g.*, BW(P_1)= Min(3,4,5,6,6)=3. The bandwidth of G, BW(G), is calculated using the Max-Flow Min-Cut theorem [12]; for Fig. 1, BW(G)=11. Since G can be constructed using all nodes in V and links in E or paths in P_G, Cost(P_G)=Cost(G)=Cost(E), Rel(P_G)=Rel(G)=Rel(E), BW(P_G)=BW(G)=BW(E), Cost(P_i)=Cost(L_i), Rel(P_i)=Rel(L_i) and BW(P_i)=BW(L_i). Table 1 shows each P_i with its Cost(P_i), Rel(P_i) and BW(P_i).

4. Problem Statement

Let $X_i \in \{0, 1\}$ be a variable that indicates if path P_i in G=(V, E) is selected (X_i=1) or is not selected (X_i=0). The following equations (1) and (2) define NTD-CR/B problem:

$$\text{Min (Cost(} \textstyle\bigcup_{i=1}^{|P_G|} P_i\, X_i \text{)) and Max (Rel(} \bigcup_{i=1}^{|P_G|} P_i\, X_i\text{))} \quad (1)$$

$$\text{Subject to} \quad \text{BW(} \textstyle\bigcup_{i=1}^{|P_G|} P_i\, X_i \text{)} \geq B_{min} \quad (2)$$

Eq. (1) aims to minimize the cost and maximize the reliability of the topology induced by the paths selected in Eq. (2) while its bandwidth is at least B_{min}. Specifically, the goal of NTD-CR/B is to remove some links in G such that the remaining topology G_{best} satisfies Eq. (1) and Eq. (2). Note that we call each topology that satisfies Eq. (2) a *feasible* solution. One may generate all 2^n possible combinations of paths, and for each combination that has bandwidth at least B_{min}, use Eq. (1) to select one with the best value as G_{best}. However, this solution is prohibitive for use in large networks since a general network contains $n=2^{|E|-|V|+2}$ (*s*, *t*) simple paths [2].

5. Proposed Dynamic Programming-Based Solutions

This section first explains a Lagrange-relaxation method [13] to convert the BO in NTD-CR/B problem to an SO using weighted sum method. Then, it describes a DP formulation to solve the problem. Next, it presents DPCR/B algorithm followed by an illustrating example. Finally, it shows the time complexity of DPCR/B.

5.1 Converting BO into SO Problem

We use the Lagrange-relaxation method in [13] to combine objectives Cost(G) and Rel(G) and constraint BW(G) for NTD-CR/B problem into a weight Ψ(G) as:

$$\Psi(G)=Cost(G)\times Rel(G)+\eta\times BW(G) \tag{3}$$

The value of η should be set properly to heuristically minimize Cost(G) and maximize Rel(G) in Eq. (1). For our problem, we set η as:

$$\eta=C_{min}\times R_{max}/B_{min} \tag{4}$$

where C_{min} (R_{max}) is the minimum cost (maximum reliability) among all possible paths in the network. One can obtain C_{min} and R_{max} using Dijkstra's algorithm [12]. We use Eq. (3) and Eq. (4) to convert the BO optimization in Eq. (1) into the SO optimization:

$$\text{Max}(\Psi(G)=Cost(\bigcup_{i=1}^{|P_G|} P_i X_i)\times Rel(\bigcup_{i=1}^{|P_G|} P_i X_i)+\eta\times BW(\bigcup_{i=1}^{|P_G|} P_i X_i)) \tag{5}$$

To achieve the objective in Eq. (5), we first combine cost c_j, reliability r_j and bandwidth b_j of each link e_j into a weight w_j. Here, we use the Lagrange-relaxation technique in [13] to compute $w_j=c_j\times r_j+\eta_w\times b_j$, where $\eta_w=c_{min}\times r_{max}/B_{min}$ and c_{min} (r_{max}) is the minimum cost (maximum reliability) of among all links in the network. Note that the computation of each w_j and η_w corresponds to Eq. (3) and Eq. (4), respectively. Then, we generate all (s, t) paths of G in increasing order of their weights; path weight is calculated as the summation of the weight of each link in the path. We refer to such ordering criteria as CR1. Finally, as shown in the following sections, we greedily construct the topology using the links in each selected path, starting from the paths with the largest weight.

5.2 Dynamic Programming Formulation for NTD-CR/B

Let PX_i, for i=1,2, …,n, be a set of paths selected from n paths in $\{P_1,P_2,…,P_{n-1},P_n\}$ and $G_i=(V,E_i\subseteq E)$ be its induced graph whose links comprise all links in PX_i. This paper uses PX_i and its G_i interchangeably since one can generate G_i from PX_i, and vice versa. There are 2^n different PX_i, for $1\le|PX_i|\le n$, and we aim to select PX_n with a bandwidth $BW(G_n)\ge B_{min}$ and the maximum $\Psi(G_n)$; the latter goal aims to maximize the goal in Eq. (5) and hence heuristically minimizes Cost(G_n) and maximizes Rel(G_n) in Eq. (1).

Let DP[1.. n,0.. B_{min}] be a 2-dimension array in which each DP[i,b], for i=1,2,…,n, b=0,1,2,…,B_{min}, stores seven pieces of information: cost $C[i,b]\ge 0$, reliability $0\le R[i,b]\le 1.0$, bandwidth $0\le B[i,b]\le B_{min}$, weight $W[i,b]\ge 0$, set of paths $P[i,b]\subseteq P_G$, set of links $L[i,b]\subseteq E$, and integer index $0\le J[i,b]\le B_{min}$. In essence, each bth column corresponds to a bandwidth constraint b=0,1,…,B_{min}. Each DP[i,b] is used to store the information of each selected topology G_i that has $BW(G_i)\ge b$. Specifically, for each $BW(G_i)\ge b$, $W[i,b]=\Psi(P_i)$, $P[i,b]=PX_i$. Since $W[n,B_{min}]$ is the weight of $G_n=(V,E_n\subseteq E)$ with $BW(G_n)\ge B_{min}$, NTD-CR/B aims to generate DP[n,B_{min}] with maximum $W[n,B_{min}]$ to produce G_{best}. We use Eq. (3) and Eq. (4) to compute $W[n,B_{min}]=\Psi(G_n)$. Our DP approach, called DPCR/B, computes each W[i,b] using the following four equations:

$$W[i,b]=\Psi(P_i), \text{ if } i=1 \text{ and } BW(P_i)\ge b \tag{6}$$

$$W[i,b]=0, \text{ if } i=1 \text{ and } BW(P_i)<b \tag{7}$$

$$W[i,b]=\text{Max}(W[i-1,b], \Psi(P_i)), \text{ if } i>1 \text{ and } BW(P_i)\ge b \tag{8}$$

$$W[i,b]=\text{Max}(W[i-1,b], \Psi(P[i-1,h]\cup P_i)), \text{ if } i>1 \text{ and } BW(P[i-1,h]\cup P_i)\ge b \tag{9}$$

The conditions in the equations are considered in increasing number of the equations, *i.e.*, a lower numbered equation takes precedence over a higher numbered equation. Eq. (6) and Eq. (7) are used for the first path; the path is selected if it has bandwidth of at least b, and otherwise is not selected, giving $W[1,b]=\Psi(P_1)$ and $W[1, b]=0$ respectively. Eq. (8) and Eq. (9) are used for each remaining P_i, for $i=2,3,\ldots,n$. Eq. (8) considers one of two options, selecting or not selecting P_i when $BW(P_i)\geq b$, and selects the option that produces the maximum weight. Specifically, it selects P_i and sets $W[i,b]=\Psi(P_i)$ if its weight is larger than the weight of the previously selected paths, *i.e.*, $W[i\text{-}1,b]$. Thus, Eq. (8) selects the maximum between the two since both options satisfy the bandwidth requirement b. Note that the bandwidth value in $B[i, b]$ would be changed to $BW(P_i)$ if P_i is selected.

Equation (9) considers the case when selecting P_i together with some previously selected paths PX_j results in weight $\Psi(PX_i)$ larger than the weight of the previously selected paths, *i.e.*, $\Psi(P[i\text{-}1,h]\cup P_i)>W[i\text{-}1,b]$, while satisfying the required bandwidth b, *i.e.*, $BW(P[i\text{-}1,h]\cup P_i)\geq b$. This step considers each possible column $h=J[i, b]=0,1,\ldots,B_{min}$. Eq. (8) and Eq. (9) also consider a case when no path has so far been selected in the column, for which both will select P_i. Further, they update the values of $C[i,b]$, $R[i,b]$, $B[i,b]$, $P[i,b]$, $L[i,b]$, and $J[i,b]$ for each selected P_i; otherwise, the values are the same as their corresponding values in the previous row, *e.g.*, $C[i,b]=C[i\text{-}1,b]$.

The DP formulation in Eq. (6) to Eq. (9) is similar to the DP solution for the NP-complete 0/1 knapsack problem (KP) [11]. In KP, there are n items where each *item* has weight and value and its goal is to select a set of *items* that have the *maximum total value* while having *total weight no larger* than a given weight constraint. In contrast, NTD-CR/B aims to select a set of *paths* whose induced topology has *maximum weight* while having *bandwidth no less* than a given B_{min}. However, unlike for KP where the total weight of two items is the sum of each item's weight, in NTD-CR/B, $\Psi(P_i)+\Psi(P_p)\geq\Psi(P_i\cup P_p)$ because P_i and P_p may contain common links, *e.g.*, $P_1=(2,5,4,3,7)$ with $\Psi(P_1)=19.8$ and $P_2=(1,4,6,7)$ with $\Psi(P_2)=17.4$ that share link 4 and 7 have $\Psi(P_1)+\Psi(P_2)=37.2\geq\Psi(P_1\cup P_2)=20.34$. Thus, Eq. (9) considers all possible values of h, *i.e.*, $J[i,b]$, unlike its equivalent step for KP [11].

5.3 DPCR/B Algorithm

One can design a DPCR/B algorithm that directly implements the DP in Eq. (6) to Eq. (9). However, the algorithm requires all n paths of the network, which is not feasible for networks with large n. Alternatively, the following greedy heuristic DPCR/B requires only a small number of $1\leq k\leq n$ paths while producing almost the same result as compared to using all paths in the network. Because there is no effective way for setting the value of k a priori, we set k dynamically following the method in [2]. Specifically, the algorithm repeatedly generates the next largest weight P_{i+1}, if it obtains a feasible solution G_{i+1} with higher weight as compared to G_i, *i.e.*, $\Psi(G_{i+1})>\Psi(G_i)$. The algorithm stops when it generates 10 consecutive paths with total additional weight of less than 1%.

DPCR/B Algorithm:

```
1.  Calculate $w_j=c_j\times r_j+\eta_w\times b_j$, for each link $e_j$
2.  Generate paths ($P_{k+10}, \ldots, P_k, \ldots, P_1$)
3.  Set $W[1,b]=0$, for $b>BW(P_1)$      // Eq. (7)
4.  Set $W[1,b]=\Psi(P_1)$, for $0\leq b\leq BW(P_1)$    // Eq. (6)
5.  Copy row1 to row2 and initialize $m=0$, $i=2$
6.  While ($i\leq k+10$) do    // Eq. (8) and Eq. (9)
7.      if ($m\neq 10$)
```

```
8.        W[2,b]←Max(W[1,b], Ψ(P_i)),0≤b≤BW(P_i))
9.        for (y ← 0 to B_min) do // Eq. (9)
10.           if (J[1,y]≠J[1,y+1])
11.               h=J[1,y], d=BW(P[1, h]∪P_i)
12.           W[2,d] ← Max(W[1,d], Ψ(P[1,h]∪P_i))
13.       if (W[2,B_min]<1.01×W[1,B_min])
14.           m++
15.       else
16.           m=0
17.       Copy row2 to row1, i++, goto step 6
18.   else
19.       Return
```

Step 1 of DPCR/B calculates weight w_j for each link e_j. Step 2 uses Yen's algorithm [12] to generate a sequence of paths $(P_{k+10},P_{k+9},\ldots,P_k,P_{k-1},\ldots,P_2,P_1)$ in order of increasing weights, *i.e.*, P_1 has the largest weight; we set k=10000-10. Step 3 and 4 implement Eq. (7) and Eq. (6) respectively. DPCR/B uses only two rows of DP table, called *row*1 and *row*2. The algorithm repeatedly executes Step 6 to 19 until it generates G_{10000}, *or* $G_{(k<10000)}$ when the last 10 consecutive paths add insignificant improvement in $W[i,B_{min}]$, *i.e.*, less than 1%. For each iteration, the steps only update entries in *row*2 that improve their corresponding entries in *row*1; thus, *row*1=*row*2 at the beginning of each iteration (see Step 5 and 17). Note that the pseudo code shows only updates to $W[i,b]$ to save space, although DPCR/B also updates the other six corresponding set of information, *e.g.*, $R[i,b]$ that is estimated using the Monte Carlo Simulation (MCS) from [5]. Step 8 implements Eq. (8), and Step 9 to 12 use Eq. (9). Step 13 to 16 are used to stop the iteration, *i.e.*, when $\Psi(G_{k+10})<1.01\times\Psi(G_k)$.

Ideally, Step 2 should generate the first k largest-weight paths. However, generating path with the maximum weight is equivalent to generating the longest path [11], an NP-complete problem. Our implementation is ideal for networks with less than k=10000 paths. However, Section 6.2 shows that DPCR/B produces topology only 8.8% off from optimal for networks with 2^{99} paths.

5.4 Illustrating Example for DPCR/B

Consider Fig. 1 with B_{min}=8, c_{min}=2, r_{max}=0.9 and η_w=0.225. DPCR/B constructs the DP table in Table 2 to obtain G_{best}. For convenience, the table shows n=7 rows; each row considers selection of each P_i, and each column represents bandwidth constraint from 1 to B_{min}. To save space Table 2 shows only the values of $P[i,b]$ and $W[i,b]$. Step 1 of DPCR/B obtains w_1=6.3, w_2=2.5, w_3=2.8, w_4=4.7, w_5=6.3, w_6=3.3, w_7=2.8 and w_8=3.8. Step 2 uses Yen's algorithm [12] to generate $P_G=(P_7=(1,3,7),\ P_6=(2,5,8),\ P_5=(1,4,8),\ P_4=(2,5,6,7),\ P_3=(1,3,6,8),\ P_2=(1,4,6,7),\ P_1=(2,5,4,3,7))$ in increasing order of weights; *i.e.*, P_1 has the largest weight. Step 3 initializes $C[1,b]=\infty$, $R[1,b]=0$, $B[1,b]=0$, $P[1,b]=\{\}$, $L[1,b]=()$ and $W[1,b]=0$, for $b=4,\ldots,8$, and thus $J[1,b]=B_{min}=8$. Step 4-5 sets $C[1,b]=17$, $R[1,b]=0.24$, $B[1,b]=3$, $P[1,\ b]=\{1\}$, $L[1,b]=(2,5,4,3,7)$ and $W[1,b]=\Psi(P_1)=6.37$, for $b=1,\ldots,3$ with $h=J[1,b]=3$; Eq. (5) is used to calculate each $\Psi(P_i)$. Step 8 and 9, select $P_2=(1,4,6,7)$ with $\Psi(P_2)=8.384$ and $BW(P_2)=4$, because $i=2<k=7$ and $m=0<10$. Step 10 and 11 use Eq. (8) to update $W[2,b]=\Psi(P_2)$ only for b=4 because $\Psi(P_2)>W[1,4]=0$. Step 12-16 consider h starting from b=0 in *row*1 with two values: $h=J[1,b]=3$ and $h=J[1,b]=8$. For h=3, $\Psi(P[1,h]\cup P_2)=20.34>W[1,b]$; thus Eq. (9) selects both P_1 and P_2 at column b=5 and b=6 and sets $C[2,b]=26$, $R[2,b]=0.59$, $B[1,b]=6$, $P[1,b]=\{1,2\}$, $L[1,b]=\{1,2,3,4,5,6,7\}$ and $W[1,b]=20.3$ for b=5 and 6. For h=8, Eq. (9) selects $P_2(P_2\cup\{\})$ at b=4. Repeating the steps for P_3 to P_7, DPCR/B obtains an optimal topology (Fig. 1 without links 2 and 5) with $P[7,B_{min}=8]=\{P_2,P_3,P_5,P_7\}$, C[7,8]=20, R[7,8]=0.833, B[7,8]=8, L[7,8]=(1,3,4,6,7,8) and W[7,8]=23.2. Note that for Fig. 1, DPCR/B produces exactly the same topologies as those using brute force (generated manually) for 11 different possible values of $B_{min}\leq$BW(G)=11, except for $B_{min}\leq 3$. Using the rc_i ratio, described in Section 6.1, the three different results are no worse than 0.0887% off optimal.

Table 2: DP Table for Network of Figure 1

	1	2	3	4	5	6	7	8
P_1	(1), 6.37	(1), 6.37	(1), 6.37	{}, 0	{}, 0	{}, 0	{}, 0	{}, 0
P_2	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2),20.3	(1,2),20.3	{}, 0	{}, 0
P_3	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2),20.3	(1,2),20.3	(2,3),23.2	(2,3),23.2
P_4	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2,4),20.3	(1,2,4),20.3	(2,3),23.2	(2,3),23.2
P_5	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2,4),20.3	(1,2,4),20.3	(2,3,5),23.2	(2,3,5),23.2
P_6	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2,4),20.3	(1,2,4),20.3	(2,3,5),23.2	(2,3,5),23.2
P_7	(1), 6.37	(1), 6.37	(1), 6.37	(2), 8.38	(1,2,4,7),20.3	(1,2,4,7),20.3	(2,3,5,7),23.2	(2,3,5,7),23.2

5.5 Time Complexity

Steps 1 requires $O(1)$, and Yen's algorithm [12] in Step 2 requires $O(k\times|V|\times(|E|+|V|\times\log|V|))$ time to generate the first k=10000 paths. Function BW(P_1) in Step 3 and $\Psi(P_1)$ in Step 4 each requires $O(|E|)$. Steps 5 and 17 require $O(k\times B_{min})$. For each set X that contains (*s, t*) simple paths, function BW(X) can be implemented using Orlin's algorithm [14] with time $O(|V|\times|E|)$. Since BW(X) is used only for each different *h* in each row *i*, the total time of using BW(X) is $O(\Omega\times|V|\times|E|)$, where $\Omega\leq k\times B_{min}$ is the total number of different *h* in the table (Step 11). The Cost(X) includes every unique link in X, and for each *b*, it returns the sum of C[*i*-1,*b*] and the cost of links in P_i that are not in L[*i*-1,*b*]. Using the bit implementation [15], one requires only one bit OR and one bit XOR operation to obtain the links in P_i that are not in L[*i*-1,*b*], and thus Cost(X) is computed in $O(|E|)$. DPCR/B uses the function at most once for every table entry. Thus, the worst case time for using Cost(X) is $O(\Omega\times|E|)$. We use the MCS in [5] that has complexity of $O(U\times|V|^4)$ to estimate Rel(X) for each candidate network; *U* is the number of replication. In Step 12, Dijkstra's algorithm takes $O(|V|^2\times\log|V|)$ time to find C_{min} and R_{max}. We use Cost(X), Rel(X) and BW(X) to calculate Ψ(X) in Step 12. Note that Cost(X), Rel(X) and Ψ(X) are used only when BW(X)$\geq b$. Thus, DPCR/B has a time complexity of $O(1+(k\times|V|\times(|E|+|V|\times\log|V|))+k\times B_{min}+|E|+|E|+(\Omega\times(|V|\times|E|+|E|+U\times|V|^4+1))+(|V|^2\times\log|V|))$ $=O(\Omega\times(|V|^4))$, since k, B_{min} and U are constants and $|E|\leq|V|^2$.

6. Results

We have implemented DPCR/B in C and run all simulations on Intel Core i5 (2 cores) with 2.53 GHz with 4 GB of RAM, running Linux (Ubuntu 11.10). In Section 6.1, we run DPCR/B on 20 networks from [16] to observe the effects of using CR1 path ordering on the effectiveness of the algorithm. We compare its performance against three other orders, *i.e.*, CR2, CR3, and random. For CR2, we generate, for each link e_j, weight $w_j=c_j/r_j$; this criterion has been shown effective for the NTD-RC problem [2]. For CR3, we set $w_j=b_j$. The random order, as in CR1, computes each link's weight $w_j=c_j\times r_j+\eta_w\times b_j$. However, unlike CR1 that selects paths in decreasing order of their weights, it selects paths randomly. To implement CR2 and CR3, we replace the link weight calculation in Step 1 of DPCR/B in Section 5.3 with w_j=c$_j$/r$_j$ and w_j=b$_j$, respectively. In Section 6.2, we use DPCR/B on 125 benchmark networks to gauge its effectiveness.

Table 3: Effects of Path Set Orders on the Performance of DPCB/B

CN	**Cost(G$_{best}$), Rel(G$_{best}$)**				
	***B**$_{min}$*	**Random**	**CR1**	**CR2**	**CR3**
$CN_{9}^{5,8}$	9	**(14,0.902)**	**(14,0.902)**	**(14,0.902)**	**(14,0.902)**
$CN_{7}^{6,8}$	6	(25,0.761)	**(20,0.833)**	**(20,0.833)**	**(20,0.833)**
$CN_{13}^{6,9}$	10	(25,0.843)	**(26,0.937)**	(28,0.917)	(27,0.923)
$CN_{13}^{9,12}$	12	**(27,0.837)**	(30,0.707)	(29,0.817)	(28,0.768)
$CN_{14}^{7,15}$	12	(31,0.648)	**(30,0.702)**	(28,0.648)	**(30,0.702)**
$CN_{18}^{11,21}$	10	(33,0.979)	**(34,0.984)**	(33,0.979)	(33,0.979)
$CN_{18}^{9,13}$	14	(66,0.994)	**(53,0.991)**	(57,0.985)	(55,0.987)
$CN_{24}^{8,12}$	10	(40,0.907)	**(34,0.892)**	**(34,0.892)**	**(34,0.892)**
$CN_{20}^{8,12}$	10	**(37,0.899)**	**(37,0.899)**	(38,0.886)	(36,0.704)
$CN_{25}^{7,12}$	12	(35,0.896)	**(36,0.920)**	(34,0.894)	(34,0.894)
$CN_{29}^{8,13}$	14	**(34,0.97)**	(36,0.95)	**(34,0.970)**	(36,0.972)
$CN_{36}^{16,30}$	10	(38,0.867)	**(39,0.956)**	(43,0.935)	(40,0.935)
$CN_{44}^{21,26}$	12	(93,0.998)	**(89,0.996)**	(93,0.998)	**(89,0.996)**
$CN_{44}^{9,14}$	10	(79,0.783)	**(77,0.799)**	**(77,0.799)**	**(77,0.799)**
$CN_{64}^{10,21}$	15	(46,0.913)	**(43,0.917)**	**(43,0.917)**	(45,0.910)
$CN_{136}^{17,25}$	10	(64,0.970)	(64,0.970)	**(58,0.985)**	(65,0.982)
$CN_{205}^{18,21}$	8	(80,0.971)	**(78,0.969)**	(80,0.971)	(85,0.973)
$CN_{281}^{13,22}$	10	(78,0.980)	**(68,0.982)**	(75,0.965)	**(68,0.982)**
$CN_{282}^{18,27}$	10	(95,0.963)	**(93,0.972)**	(97,0.975)	**(93,0.977)**
$CN_{780}^{20,30}$	15	(82,0.974)	**(75,0.981)**	(80,0.977)	(79,0.972)

6.1 The Effect of Path Orderings on DPCR/B

For each of the 20 networks [16] in Table 3, we randomly assigned r$_j$, c$_j$ and b$_j$ for each e$_j$, and set B_{min} randomly between 40% to 80% of the bandwidth of the original network with complete links; each $CN_n^{|V|,|E|}$ denotes a CN with |V| nodes, |E| links and *n* paths. Reference [16] gives the details of r$_j$, c$_j$, b$_j$, and B_{min} for each $CN_n^{|V|,|E|}$. Then, we obtain weight w_j for each link e$_j$ for CR1, CR2, and CR3. Next, we use Yen's algorithm [12] to generate each path set; for CR1, CR2, and CR3, we select the paths in decreasing order of their weights, and for random, select the paths randomly. Note that selecting the better solution between two feasible solutions is not straightforward. For example, for Fig. 1 with $B_{min=}$8, one may obtain solutions {Cost(G$_1$)=25, Rel(G$_1$)=0.878, BW(G$_1$)=11} and {Cost(G$_2$)=20, Rel(G$_2$)=0.833, BW(G$_2$)=8}. Notice that neither solution satisfies the BO in Eq. (1) because Cost(G$_2$)<Cost(G$_1$) but Rel(G$_2$)<Rel(G$_1$). In general, a feasible solution G$_i$ for NTD-CR/B problem may only minimize cost ***or*** maximize reliability. Here, we define a ratio rc$_r$=Rel(G$_r$)/Cost(G$_r$), and consider a feasible topology G$_i$ *better* than G$_j$ if (i) rc$_i$>rc$_j$, or (ii) rc$_i$=rc$_j$ and BW(G$_i$)≥BW(G$_j$). For the previous example, G$_2$ is better than G$_1$ because rc$_2$=Rel(G$_2$)/Cost(G$_2$)=0.833/20=0.041>rc$_1$=0.878/25 =0.035.

As shown in Table 3, DPCR/B with each path order produces between 7 and 17 of the best results (in bold) as compared to only 4 using random path order. The table shows that CR1 is the best performer with 17 best results, followed by CR3 with 8, and CR2 with only 7 best results. One may run DPCR/B with all three path orders and select the best results among them; for the simulations, we would have obtained all 20 best results.

6.2 The Performance on Benchmark Networks.

Since we could not find any network with known optimal solution, we used the following steps to generate 125 benchmark networks. We consider the optimal solution of each network G=(V, E) with α=1 deleted link $e_i \in E$. For each of possible |E| deletions, generate G_i=(V, E-{e_i}), for i=1,2,…,|E|, and compute BW(G_i). Then, select a G_i with the largest rc_i =Rel(G_i)/Cost(G_i) as G_{best}, and set B_{min}=BW(G_i). Note that given G and B_{min}, an optimal solution would generate G_{best}=G_i. We repeat the steps only for α=2,3,4,5 because for larger α and networks, there are huge number of G_i, and finding G_{best} among them needs computing each Rel(G_i), which is a huge task; *e.g.*, for |E|=30, |V|=20 and α=6, there are 593775 G_i's Using the steps, we generated 100 networks from the 20 topologies in [15] and 25 networks from five grid networks, $Grid_{6\times6}$, $Grid_{3\times12}$, $Grid_{3\times16}$, $Grid_{2\times20}$, $Grid_{2\times100}$, that contain 36 to 200 nodes, 57 to 298 links, and 538020 to 2^{99} (*s*, *t*) paths; reference [16] provides details of link reliability and cost for each benchmark network.

Table 4: Comparison between DPCR/B and Optimal Results

Input			***k***	**DPCR/B**	
CN	α	B_{min}		**Rel(G_{best})**	**Cost(G_{best})**
$CN_{13}^{9,12}$	1	10	8(61.5%)	0.807(-0.07%)	27(0%)
$CN_{18}^{11,21}$	2	12	11(61.1%)	0.979(1.01%)	33(1.03%)
$CN_{24}^{8,12}$	2	9	15(60%)	0.920(-0.7%)	36(1.05%)
$CN_{36}^{16,30}$	3	12	12(33.3%)	0.956(1.02%)	39(1.05%)
$CN_{205}^{18,21}$	3	11	37(18%)	0.969(-0.4%)	78(0%)
$CN_{281}^{13,22}$	4	13	66(23.4%)	0.982(-0.02%)	68(-0.98%)
$CN_{136}^{17,25}$	5	7	41(30.1%)	0.974(-0.09%)	77(-0.96%)
$CN_{282}^{18,27}$	5	15	102(36.1%)	0.97(-0.04%)	82(-0.94%)
$CN_{64019921}^{48,77}$	5	23	354(5.53^{-6}%)	0.95(-0.58%)	70(-2.77%)
$CN_{524288}^{40,58}$	5	25	227(4.3^{-4}%)	0.75(-1.29%)	53(0%)
$CN_{2^{99}}^{200,298}$	5	32	873(1.37^{-27}%)	0.25(-0.78%)	290(-2.6%)

We run DPCR/B thrice, once each using CR1, CR2, CR3, to generate topologies from the 125 networks, and take the best results among them. For each network G_i, we set B_{min}=BW(G_{best}), and use DPCR/B to generate its topology G'_{best}. Notice that our method evaluates the worst case performance of DPCR/B because each B_{min} is the tightest constraint. Nevertheless, DPCR/B generates 91.2% optimal results; due to limited space, Table 4 shows only the 11 non-optimal results. Interestingly, of the non-optimal results, DPCR/B produces network with reliability no worse than 1.29% off optimal (see % at column Rel(G_{best})) with some of which have lower cost than that of optimal by up to 2.77%. Further, it uses only 1.37^{-27}% of the 2^{99} paths in $Grid_{2\times100}$; see the % at column '*k*'.

7. Conclusion

We have defined NTD-CR/B problem to generate a topology with minimum cost and maximum reliability subject to a bandwidth constraint B_{min}, and proposed a heuristic DP method, DPCR/B, to solve it. Our method incrementally selected only k+10 paths from the network and, thus, is scalable on networks with large number of paths. We have proposed three path orders, CR1, CR2 and CR3, to optimize our method's effectiveness and efficiency. The experimental study shows that DPCR/B generate 91.2% optimal solutions. We plan to extend our approach for network design with other performance constraints, *e.g*., delay and throughput, and/or maximizing bandwidth as an objective.

References

[1]. Loh, R.C. *Using Edge-Disjoint Paths to Improve the QoS in Computer Communication*, Ph.D. dissertation, Dept. Comp., Curtin Univ., Perth, Australia, 2010.

[2]. Elshqeirat, B., S. Soh, S. Rai, and M. Lazarescu, *A Practical Algorithm for Reliable Communication Network Design,* International Journal of Performability Engineering, 2013; 9(4): 397-408.

[3]. B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu, *Dynamic Programming for Minimal Cost Topology with Two Terminal Reliability Constraint*, *Proc. IEEE APCC*, Indonesia, 2013.

[4]. Frank,H. and W. Chou, *Topological optimization of computer networks,* Proc. IEEE, 1972; 60(11):1395-1397.

[5]. Yeh, S., J. Lin and W. Yeh, *New Monte Carlo Method for Estimating Network Reliability*, Proc. ICCIE, 1994; 723-726.

[6]. Khan, A., and A. P. Engelbrecht, *A Fuzzy Particle Swarm Optimization Algorithm for Computer Communication Network Topology design*, Appl Intell. , 2012; 36(1): 161-177.

[7]. Kumar, R., P. P. Parida, and M. Gupta, *Topological Design of Communication Networks using Multiobjective Genetic Optimization*, Proc. CEC-02, 2002; 425-430.

[8]. Banerjee, N., and R. Kumar, *Multiobjective Network Design for Realistic Traffic Models*, Proc. GECCO, 2007; 1904-1911.

[9]. Papagianni, C., *et al.*, *Communication Network Design Using Particle Swarm Optimization, Proc. Int. Multiconf. on Computer Science and Information Technology*, 2008; 915-920.

[10]. Duarte, S., and B. Bar´an, *Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms*, XXVII CLEI’2001, 2001.

[11]. Altiparmak, F., B. Dengiz, and O. Belgin, *Design of reliable communication networks: A hybrid ant colony optimization approach for the design of reliable networks*, IIE Transactions, 2010; 42: 273–287.

[12]. Cormen,T. H., C. E. Leiserson and R. L. Rivest, *Introduction to Algorithm*, Cambridge, Massachusetts: The MIT Press, 2009.

[13]. Fisher, L. *The Lagrangian Relaxation Method for Solving Integer Programming Problems*, I NFORMS Management Science, 2004; 50(12): 1861-1871.

[14]. Orlin, J. B. *Max Flows in O(nm) Time, or Better*, ACM Symposium on Theory of Computing, 2013.

[15]. Rai, S. and S. Soh, *CAREL: Computer Aided Reliability Evaluator for Distributed Computer Networks*, IEEE Trans. Parallel Distrib. Syst., 1991; 2(2): 199-2

[16]. Elshqeirat, B. *Optimizing Reliable Network Topology Design Using Dynamic Programming,* Ph.D. dissertation, Dept. Comp., Curtin Univ., Perth, Australia, 2015.