Modern Applied Science; Vol. 12, No. 12; 2018
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

On Maximizing Reliability of Network Topology Design Using a Practical Dynamic Programming Approach

Basima Elshqeirat[1], Sieteng Soh[2], Suresh Rai[3] & Saher Manaseer[1]

[1] Department of Computer Science, The University of Jordan, Amman, Jordan

[2] Department of Computing, Curtin University, Perth, Western Australia, Australia

[3] Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA, USA

Correspondence: Basima Elshqeirat, Department of Computer Science, The University of Jordan, Amman, Jordan. E-mail: B.shoqurat@ju.edu.jo

Received: October 9, 2018 Accepted: October 16, 2018 Online Published: November 19, 2018

doi:10.5539/mas.v12n12p163 URL: https://doi.org/10.5539/mas.v12n12p163

Abstract

This paper addresses an NP-complete problem of designing a network topology (NT) with the maximum 2-terminal reliability (R) subject to a cost constraint (C). More specifically, given the locations of the various computer centers (nodes), their connecting links, each link's reliability and cost, and the maximum budget cost to install the links, the NT design problem, called NTD-RC, aims to find an NT that has the maximum reliability with cost within the budget. Since cost is a major issue in NT design, NTD-RC is applicable for critical applications requiring maximized reliability. This paper formulates a dynamic programming (DP) scheme to help solve NTD-RC. A DP approach, called Algo-DP, finds the set of links to be deleted from the original network to obtain an optimal NT. The paper proposes five-link ordering criteria to improve the performance of Algo-DP. Simulation results on different benchmark networks of various sizes are used to compare Algo-DP with existing techniques in the literature and show the merits of using the sorting methods, and the effectiveness of our algorithm. We found that Algo-DP generates NT with the same or better 2-terminal reliability measure (with up to 4.3% improvement) on 92% of the network topologies. Results indicate Algo-DP demonstrated better performance than other existing algorithm. Furthermore, Algo-DP shows that it is computationally more efficient compared to the recent existing approach.

Keywords: dynamic programming, network optimization, network reliability, network topology design

1. Introduction

Communication networks (CN) are complex, sophisticated, automated and computerized. Once a hardware model and technology of a network has been developed, a network designer is challenged with the problem of designing a network that satisfies the performance requirements, *e.g.*, reliability, anticipated by the customer over the intended period of the network's use (Konak & Smith, 1999). The design of network topology (NT) is an important part of network design (Bhupesh, et. al, 2008), since NT is directly associated with network deployment cost and its reliability (Reichelt & Rothlauf, 2005). Consequently, network topology design (NTD) has received considerable attention.

In practical networks, we treated the network topology design (NTD) problem as a single-objective optimization problem, if the design process and consideration of only one objective and one condition (Saad, et. al, 2018).

A well-designed Communication Network is inseparable from the effective running of user applications. For critical applications (*e.g.*, emergency system, rescue and military operations) it is important that CN topology is as reliable as possible since in practice network components, *e.g.*, links, are failure-prone (Elshqeirat, et. al, 2014). A more reliable topology will make the CN operate effectively and without interruption, even in the presence of the component failures (Gen, 2006). The NTD should include finding the best layout of network components. This layout considers network deployment cost and user performance criteria, such as reliability. However, constructing a reliable topology incurs higher installation cost. For a set of various centers (nodes), their possible connecting links, link failure rate and installation cost, a network topology designer must carefully select the most suitable set of links such that the resulting model meets its cost objective and/or required reliability. This paper considers the (s, t) terminal reliability (Won & Karray, 2010), also called 2-terminal reliability, as the measure of reliability (R), which gives the probability that there is at least one operational path between two nodes, *i.e*., the source node *s* and the terminal node *t*. Some applications require a network with a guaranteed minimum reliability in order to properly operate. For such applications, the topology design emphasizes on minimizing the network installation cost subject to the required reliability level. The NTD problem that minimizes the cost (C) subject to required reliability is referred to as NTD-CR. In practice, however, the network service provider has a budget limit to build the topology. Thus, the design must emphasize on constructing the topology within the cost constraint while maximizing its reliability; we call this problem as NTD-RC. Here, the aim is to produce a network topology with the maximum reliability subject to the cost constraint to install the selected links. Both these problems are NP-hard (Abo ElFotoh & Al-Sumait, 2001). Obviously, one must use heuristic and/or approximation solutions to design large sized topologies.

While there are many solutions for NTD-CR problem, *e.g.*, (Abo ElFotoh & Al-Sumait, 2001; Gen, 2006; Konak & Smith, 2005; Atiqullah & Rao, 1993; Papagianni, et. al, 2008; Khan & Engelbrecht, 2008; Chelouah & Siarray, 2000; Mutawa, et. al, 2009; Elshqeirat, et. al, 2015; Elshqeirat, et. al, August 2013; Elshqeirat, et. al, December 2013), we found only a few papers, *e.g.*, (Elshqeirat, et. al, July 2013; Elshqeirat, et. al, 2014; Altiparmak & Dengiz , 2009; Zakir & Abd-El-Barr, 2002), that address the equally important NTD-RC problem (refer to Section 2.2). We noticed that approaches that are good to solve NTD-CR, *e.g.*, (Gen, 2006), (Atiqullah & Rao, 1993; Chelouah & Siarray, 2000), fail to effectively solve the related NTD-RC. They either become computationally expensive or lack the necessary precision to generate an acceptable solution (Elshqeirat, et. al, 2015). In this paper, we provide a heuristic that helps effectively solve NTD-RC problem.

The main contribution of this paper is two folds:

(a) First, it proposes a dynamic programming (DP) formulation and its implementation, called Algo-DP, to solve the NTD-RC problem by deleting links from the original network. Authors in (Elshqeirat, et. al, July 2013) have also used DP approaches. However, their technique potentially require generating all (*s*, *t*) simple paths to solve the problem (refer to Section 2.2). Note that, an arbitrary network that contains a set of V nodes and a set of E links has $O(2^{|E|-|V|+2})$ (*s*, *t*) simple paths (Elshqeirat, et. al, July 2013). In contrast, our DP approach deletes sequentially only up to |E| links to solve the problem, and thus significantly reduces the time complexity.

(b) Second, this paper proposes five link-ordering criteria, *i.e*., LO1, LO2, LO3, LO4 and LO5, discussed in Section 4.3. More specifically, Algo-DP uses each ordering criterion to determine the order of link deletions from the original topology to optimize the generated NT. Our simulations on 25 networks with various sizes with up to 200 nodes, 298 links and 2^{99} (*s*, *t*) paths, reported in Section 5, show the benefits of our method as compared to an existing approach in (Elshqeirat, et. al, July 2013). Specifically, Algo-DP generates NT with the same or better 2-terminal reliability (with up to 4.3% improvement) on all but only two (with about 1.4% worse reliability) of the 25 network topologies.

The layout of this paper is as follows. Section 2 discusses the network model and surveys related problems. Section 3 formulates the NTD-RC problem and provides assumptions. Section 4 describes the proposed solutions. Section 5 presents the simulation results. Finally, Section 6 concludes the paper and discusses the future work.

2. Network Model and Notations

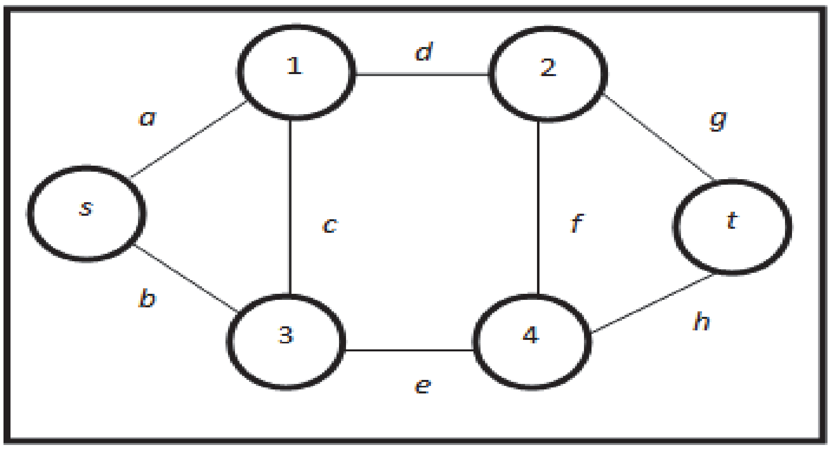

Figure 1. 6-node, 8-link Example Network

A communication network can be modeled by a probabilistic bidirectional simple graph G=(V, E), in which each vertex/node $v_i \in$ V represents a network component and each edge $e_j \in$ E represents the connecting media (*e.g.*, cable, communication link) between the network components. Let *m* be the total number of links in G, *i.e.*, *m*=|E|. It is assumed that all node locations and connecting links are given. Each e_j has a cost $c_j>0$ that represents the cost to install e_j, and a reliability $0 \le r_j \le 1$ that represents the probability that e_j is functioning (UP); all nodes are always UP and use no setup costs. Edge failures are assumed statistically independent and without repair. Fig. 1 shows an example of the graph model of a network topology with six fixedly positioned nodes and eight links; Table 1 provides the c_j and r_j values for each edge e_j.

Table 1. Link Weight for Network in Fig. 1

Link	Link Weight	
e_j	c_j	r_j
a	2	0.9
b	5	0.7
c	3	0.8
d	6	0.6
e	4	0.9
f	3	0.8
g	4	0.7
h	3	0.8

The cost of a network topology G, Cost(G), is calculated by taking the sum of all c_j for each e_j in G. We define Rel(G) as the probability that there exists at least one operational path between source node *s* and terminal node *t*. Calculating Rel(G) in general is an NP-hard problem (Yeh, et. al, 1994). As described in Section 4.1, we use the Monte Carlo Simulation in (Yeh, et. al, 1994) to approximately compute Rel(G).

2.1 Network Topology Design Problem

Let Y_j be a decision variable {0, 1} that indicates if link e_j in G= (V, E) is selected (Y_j=1) or e_j is not selected (Y_j=0). The following two equations describe the NTD-RC problem:

$$\text{Maximize Rel}(G_f=(V, E_f)) \tag{1}$$

$$\text{Subject to } \sum_{j=1}^{|E|} c_j Y_j \le C_{max} \tag{2}$$

Let us define a network topology G *feasible* when Cost(G) ≤ C_{max}. Equation (1) maximizes the 2-terminal reliability of each feasible network topology G_f that contains edges E_f=E – {e_j | Y_j=0} such that its Cost(G_f) is no larger than a given cost constraint C_{max}, shown in Eq. (2). Thus, E_f is a set of selected links in Eq. (2) that form G_f.

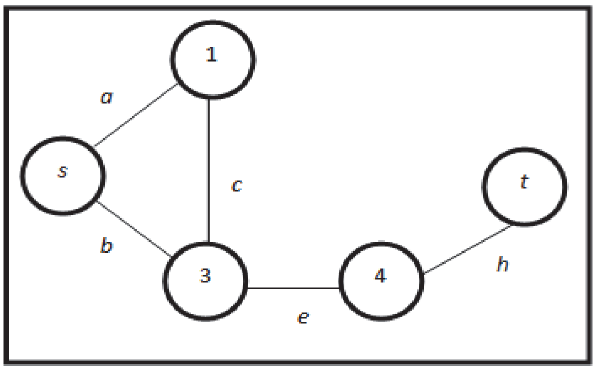

Figure 2. Optimal Solution for Network in Fig. 1

To illustrate the NTD-RC problem, consider the network in Fig. 1. For C_{max}=18, Fig. 2 shows the optimal topology, G_{opt}, with Rel(G_{opt})=0.659 and Cost(G_{opt})=17; we delete links *d*, *f* and *g* from the original network G to obtain G_{opt}. Notice that Cost(G)=30, and therefore if we set $C_{max} \ge 30$, Eq. (2) would have Y_j=1 for each e_j, and (2) considers all links in G. Thus, for this case, Eq. (1) and (2) produce G_{opt}=G with reliability 0.842 without deleting any link.

One possible way to solve NTD-RC problem is by generating each possible set of links in Eq. (2) that form G_f. Then, for each set that has a cost of at most C_{max}, calculate its Rel(G_f) and use Eq. (1) to select any G_f with the maximum reliability as the G_{opt}. Unfortunately, this brute force solution requires generating $O(2^{|E|})$ possible link selections. Further, the reliability calculation in Eq. (1) for each G_f requires exponential time. Therefore, this brute force solution is feasible only for designing small sized NT. Our work in this paper proposes a heuristic algorithm, described in Section 3.1, that generates E_f, and thus G_f, by selectively removing links in E while satisfying Eq. (1) and (2).

2.2 Related Works

The authors in (Elshqeirat, et. al, 2014; Zakir & Abd-El-Barr, 2002). proposed enumerative-based algorithms to solve the NTD-RC problem. (Zakir & Abd-El-Barr, 2002) solve the NTD-RC problem to produce the topology with as many disjoint spanning trees as possible for maximizing *all*-terminal reliability, which gives the probability that at least one spanning tree in generated network topology G is functional. Consequently, their solution (Zakir & Abd-El-Barr, 2002) produces topology that has higher reliability but with a higher cost. However, the solution (Zakir & Abd-El-Barr, 2002) require generating all disjoint spanning trees of the network, and thus are not feasible for designing networks with a large number of disjoint spanning trees (Elshqeirat, 2015) .

(Shao, et. al, 2005) proposed a heuristic approach, called shrinking and searching algorithm (SSA), to maximize all-terminal reliability given a cost constraint. In addition, (Atiqullah & Rao, 1993) described a heuristic approach based on simulated annealing to maximize all-terminal reliability under cost constraint. However, both (Shao, et. al, 2005) and(Atiqullah & Rao, 1993) address the hierarchical network topology; thus they are not useful for general networks.

Saad, et. al (2018) presented an approach based on artificial bee colony algorithm for distributed local area network topology design which was formulated as a discrete multi-objective optimization problem, note that our NTD-RC problem is different from that problem described in (Saad, et. al. 2018) because it is solving a single objective optimization problem.

Elshqeirat, et. al, July (2013) presented a dynamic programming (DP) scheme to solve the NTD-RC problem. They propose a DP approach, called DPA, to generate the topology using all (*s*, *t*) paths in the network to maximize (*s*, *t*) reliability. Five different path-orders are proposed to improve the effectiveness of DPA. Further, the path-orders allow DPA to generate only $k\geq 1$ paths dynamically from the graph model of the network and stops if including path $k+1$ leads to an insignificant improvement to the resulting topology's reliability. In Elshqeirat, et. al, (2014) the same authors used two DPA approaches, called DPA-1 and DPA-2, to solve NTD-RC with *all*-terminal reliability measure by generating the NT using a sequence of spanning trees to maximize the reliability. The first version, *i.e.*, DPA-1, requires all spanning trees of the network to maximize the reliability. The approach proves that DPA-1 able to produce optimal NT given as input an optimal sequence of spanning trees. In addition, authors in (Elshqeirat, et. al, 2014) showed that generating optimal order of the trees is NP-complete, and described five tree order criteria to heuristically generate the best sequence of spanning trees that allow DPA-1 to produce near optimal results. Further, they proposed an alternative DP algorithm, called DPA-2, that uses only *k* spanning trees sorted in increasing weight and lexicographic order to improve DPA-1's time efficiency while producing similar results.

The aforementioned DP approaches in (Elshqeirat, et. al, July 2013; Elshqeirat, et. al, 2014) have been shown effective in addressing the NTD-RC problem. However, in the worst case they require generating all possible (*s*, *t*) paths or spanning trees to form each feasible solution; thus, such approach is infeasible for large networks as, in general, a network contains an exponential number of (*s*, *t*) simple paths and spanning trees, *i.e.*, $O(2^{|E|-|V|+2})$ (*s*, *t*) simple paths and $O(|V|^{|V|})$ spanning trees (Elshqeirat, et. al, 2015). As later described in Section 4.1, in this paper, we propose Algo-DP to generate each optimized topology by deleting some links from the original network. Thus, our approach in the worst case consider only $O(|E|)$ links of the network. We also propose five link-orders each of which is used by Algo-DP to further improve results.

3. Network Topology Design Problem

3.1 Dynamic Programming Formulation for NTD-RC

Let LX_i, for i=1, 2, …, *m*, be a sequence of links selected from *i* links in (e_1, e_2, …, e_i), and $G_i=(V, E_i\subseteq E)$ be an induced graph whose links comprise of all links in G but those in LX_i, *i.e.*, $E_i = E - LX_i$; recall that $m=|E|$. Equivalently, LX_i is a set of links in (e_1, e_2, …, e_i) that are deleted from G, meaning G_i does not contain any link in LX_i. Note that $0\leq|LX_i|\leq i$, and there are $O(2^{|E_i|})$ different possible LX_i. NTD-RC aims to delete set of links in (e_1,

$e_2, \dots, e_m$) such that $\mathrm{Cost}(G_m) \le C_{max}$, and $\mathrm{Rel}(G_m)$ is the maximum. We define a solution or NT, G_i, is *feasible* if its $\mathrm{Cost}(G_i) \le C_{max}$; otherwise, it is a *non-feasible* solution. For Fig. 1 with $C_{max}=20$, $LX_8=(d, f, g)$ is a feasible solution because $\mathrm{Cost}(G_8)=17 \le C_{max}$.

Let DP[1 .. m, C_{max} .. C_G] be a 2-dimensional dynamic programming table, where C_G=Cost(G), *i.e.*, the cost of the original network with no link deletion. Each element DP[i, c], for i=1, 2, …, m, c= C_G, …, C_{max}, stores four pieces of information: a reliability $0 \le R[i, c] \le 1.0$, a sequence of links $L[i, c] \subseteq E$, a cost $C_{max} < C[i, c] \le C_G$, and an integer index $C_{max} \le J[i, c] \le C_G$. Let $R[i, c]=\mathrm{Rel}(G_i)$, for $c= C_G, \dots, C_{max}$, be the maximum reliability of G_i subject to $\mathrm{Cost}(G_i) \le c$. More specifically, R[i, c] is the reliability of $G_i=(V, E - LX_i)$ which is the best subgraph of G that contains only links $E\text{-}LX_i$ with total cost at most c, and maximum reliability. Note that R[m, C_{max}] stores the maximum reliability of G_m subject to $\mathrm{Cost}(G_m) \le C_{max}$, and NTD-RC aims to find the most optimal LX_m whose deletion from E generates the optimal NT, *i.e.*, G_{opt}.

Let $L[i, c]=LX_i$ such that $\mathrm{Cost}(G_i) \le c$, and $C[i, c]=\mathrm{Cost}(E\text{-}L[i, c])$. For each range of columns $c1 \le c \le c2$ in row i that contain the same cost value, we set each $J[i, c]=c1$. Thus, index $J[i, c]=C_G, \dots, C_{max}$ marks the starting column of a range of columns that have the same cost. For example, as later shown in Table 2, we store $J[1, c]=25$ at columns c=29 down to c=25.

Each R[i, c] is calculated using the following three equations:

$$R[i, c]=0;\ \text{if } i=1 \text{ and } \mathrm{Cost}(G\text{-}e_1) > c \tag{3}$$

$$R[i, c]=\mathrm{Rel}(G\text{-}e_1);\ \text{if } i=1 \text{ and } \mathrm{Cost}(G\text{-}e_1) \le c \tag{4}$$

$$R[i, c]=\mathrm{Max}(R[i\text{-}1, c], \mathrm{Rel}(G\text{-}L[i\text{-}1, c_j] - \{e_i\}));$$
$$\text{if } i>1,\ 1 \le c_j \le c \text{ and } \mathrm{Cost}(G\text{-}L[i\text{-}1, c_j]\text{-}\{e_i\}) \le c \tag{5}$$

Note that Rel(G-e_1) and Cost(G-e_1) refer to the reliability and cost of network G after deleting link e_1, respectively. We explain the dynamic programming formulation in Eq. (3) to (5) as follows. Without loss of generality, we consider the link selection start from the first link e_1. In Eq. (3), when Cost(G-e_1)>c, link e_1 cannot be deleted since the resulting NT is non-feasible, *i.e.*, over budget, and thus we set R[1, c]=0. In contrast, if the cost of G without e_1 is within budget c, *i.e.*, it is a feasible solution, in Eq. (4), e_1 is deleted, giving R[1, c]=Rel(G-e_1).

Equation (5) considers the case when deleting each link e_i, together with some previously deleted links in LX_{i-1} for each i>1 is feasible, *i.e.*, $\mathrm{Cost}(G\text{-}L[i\text{-}1, c_j]\text{-}\{e_i\}) \le c$, for each possible $j=J[i, c]= C_G, \dots, C_{max}$, and their induced topology G_i has the maximum reliability. Although deleting e_i produces a feasible NT, will deleting e_i lead to producing optimal topology? If e_i is not deleted, the potential maximum reliability would come from deleting links in ($e_{i+1}, e_{i+2}, \dots, e_m$) with unchanged budget c; *i.e.*, R[i, c]=R[i-1, c]. On the other hand, if e_i is deleted, the remaining allowable cost for deleting links in ($e_i, e_{i+1}, \dots, e_m$) would be reduced from c to c_j<c, for each possible c_j>0, and the resulting reliability would be Rel(G-(L[i-1, c_j]-{e_i}). Thus, Eq. (5) sets R[i, c] to the maximum between the two potential reliability values. When the two options produce the same reliability, our implementation selects the one with lower cost.

4. Proposed Solution

4.1 Dynamic Programming Algorithm

Fig. 3 shows the pseudo code of our proposed dynamic programming algorithm, called Algo-DP, which directly applies the DP formulation in Eq. (3) to (5). For a G=(V, E) that contains m=|E| links with cost constraint C_{max}, Algo-DP *implicitly* constructs a DP table of size $|E| \times C_{max}$. However, as shown in Fig. 3, Algo-DP keeps only two consecutive rows of the table, called *row*1 and *row*2, and therefore requires only a table of size $2 \times C_{max}$. Specifically, Algo-DP computes R[1, j] and L[1, j] in *row*1 using the information in R[2, c] and L[2, c] in *row*2, for all relevant columns c and j. Then, after copying the contents of *row*1 to *row*2, it repeats the step until all links have been considered.

Line 1 implements Eq. (3), while Line 2 to 5 are based on Eq. (4). The remaining of the code, *i.e.*, Line 6 to 18, is used to implement Eq. (5). Lines 19 to 22 copy the contents of *row*1 to *row*2. Function Cost(X) in Fig. 3 computes the total cost of the union of links in network X=(X-L[i-1, c_j]-{e_i}), and function Rel(X) calculates the reliability of the network X using the Monte Carlo Simulation (Yeh, et. al, 1994) .

```
Algo-DP
1.  Initialize L[2, $c$]={} and R[2, $c$]=0 for $c$<Cost (G - $e_1$) // Eq. (3)
2.  for ($c$← Cost (G)-1 down to Cost (G - $e_1$)) do // Eq. (4)
3.          L[2, $c$] ← {$e_1$}
4.          R[2, $c$] ← Rel (G- $e_1$)
5.  end for c
6.  for ($i$ ← 2 to |E|) do     // Eq. (5)
7.          for ($c$← Cost (G)-1   down to $C_{max}$) do
8.                  while($j$ != 0 && $1 \le c_j \le c$)   // **
9.                          if R[2, $c$]< Rel(G-L[2, $c_j$ ]-{$e_i$}) then
10                                  L[1, $c$] ← L[2, $c_j$ ]+ {$e_i$}
11.                                 R[1, $c$] ← Rel(G-L[2, $c_j$ ] - {$e_i$})
12.                         else
13.                                 L[1, $c$] ←   L[2, $c$]
14.                                 R[1, $c$] ← R[2, $c$]
15.                         end if
16.                         $j$ - -
17.                 end while
18.         end for c
19.         for ($Y$ ← $C_{max}$ to Cost(G)-1) do // copy row1 to row 2
20.                 L[2, $Y$ ] ← L[1, $Y$ ]
21                  R[2, $Y$ ] ← R[1, $Y$ ]
22.         end for Y
23. end for i
**$j$ is the different solutions and $c_j$ is the cost of selecting $j^{th}$ solution in row 2
```

Figure 3. Algo-DP Pseudocode

The time complexity of Algo-DP can be computed as follows. Function Cost(X) requires all unique links used in network X. For each c, Cost(X) returns the sum of links that are in network X without the set of the links in L[i-1, c] + {e_i}; recall that the links in L[i-1, c] + {e_i} are the set of links to be deleted from network X. Using the bit implementation (Soh & Rai, 1991), one requires only one bit OR and one bit XOR operation to obtain the links in X that are not in L[i-1, c] + {e_i}, and thus for any X, Cost(X) can be computed in $O(|E|)$. Algo-DP uses the function at most once for every table entry, and therefore the worst case time complexity for using the function is $O(E|^2 \times C_{max})$.

The Rel(X) function can be implemented using any exact reliability calculation (Soh & Rai, 1991), heuristic technique (Dengiz, et. al, 1997) or approximation (bounding) method (Dengiz, et. al, 1997). In this paper, we propose using the Monte Carlo Simulation (Yeh, et. al, 1994) to estimate the reliability Rel(X) of each candidate NT. The simulation has a time complexity of $O(b \times |V|^4)$ (Yeh, et. al, 1994) , where b and |V| are the number of replications and nodes respectively. Note that Rel(X) is used only if Cost(X)≤C_{max}; let ψ be the total number of cases where Cost(X)≤C_{max}. Hence the time complexity of using Rel(X) is $O(\psi \times b \times |V|^4)$, and Algo-DP requires $O(\psi \times b \times |V|^4 + |E|^2 \times C_{max})$ in the worst case.

4.2 Illustrating Example

Consider the network in Fig. 1, with each link's reliability and cost as shown in Table 1, and C_{max}=20. Algo-DP constructs the DP table in Table 2, to obtain the optimal network in Fig. 2; for convenience, we show all |E|=8 rows although our implementation creates only two rows. Further, each element DP[i, c] of the table shows only two fields, *i.e.*, the set of links to be deleted, L[*i*, *c*], and the reliability of its resulting graph, Rel[*i*, *c*].

Table 2. DP table for network in Fig. 1

	29	28	27	26	25	24	23	22	21	20
b	{*b*} 0.65	{*b*} 0.65	{*b*} 0.65	{*b*} 0.65	{*b*} 0.65	{} 0	{} 0	{} 0	{} 0	{} 0
d	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*b*,*d*} 0.59	{*b*,*d*} 0.59	{*b*,*d*} 0.59	{*b*,*d*} 0.59
g	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*d*,*g*} 0.63
c	{*c*} 0.76	{*c*} 0.76	{*c*} 0.76	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*d*,*g*} 0.63
f	{*c*} 0.76	{*c*} 0.76	{*c*} 0.76	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*f*,*d*,*g*} 0.659	{*f*,*d*,*g*} 0.659
a	{*c*} 0.76	{*c*} 0.76	{*c*} 0.76	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*f*,*d*,*g*} 0.659	{*f*,*d*,*g*} 0.659
h	{*c*} 0.76	{*c*} 0.76	{*c*} 0.76	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*f*,*d*,*g*} 0.659	{*f*,*d*,*g*} 0.659
e	{*c*} 0.76	{*c*} 0.76	{*c*} 0.76	{*d*} 0.75	{*d*} 0.75	{*d*} 0.75	{*d*,*g*} 0.63	{*d*,*g*} 0.63	{*f*,*d*,*g*} 0.659	{*f*,*d*,*g*} 0.659

Each row of the table considers each e_i for possible deletion randomly and each column shows the budget cost $c \geq C_{max}$. Since the maximum cost of the network is C_G=Cost(G)=30 and C_{max}=20, Algo-DP requires 30-20=10 columns*, i.e.*, *c* = 20, 21, …, 29. Each element in the table shows the set of links that have been selected for deletion and the (*s*, *t*) reliability of the topology when the links are deleted.

For e_1=*b*, since Cost(G-{*b*})=25, Line 1 initializes the first row with L[1, *c*]={} and R[1, *c*]=0 for each column *c*<25*, i.e.*, *c*=20, …, 24. On the other hand, for each column *c*=25, 26, …, 29, lines 2 to 5 set L[1, *c*]={*b*} and R[1, *c*]=0.65. Note that the reliability of the topology after deleting link *b* is 0.65.

Next, consider e_2=*d*. Since Cost(G-{*d*})=24, when *j*=1, Algo-DP has c_1=24 and obtains Rel(G-L[*i*-1, c_1]-{*d*}=G-{}-{*d*})=0.75, which has higher reliability than the previous result when it considered only e_1=*b*, *i.e.*, R[*i*-1, *c*]=0.65, for each *c*=25, 26, …, 29, and R[*i*-1, 24]=0. Thus, deleting {*d*} for each column *c*≥24 is better. For this case, following Eq. (5), *i.e.*, lines 7-18, Algo-DP sets L[2, *c*]={*d*} and R[2, *c*]=0.75 for each *c*≥24. Further, for *j*=2, with c_2=25 we have Cost(G-L[*i*-1, c_2] - {*d*})=Cost(G-{*b*, *d*})=19. Thus, Algo-DP selects links {*b*, *d*} at each column *c*=20, …, 23, because we have Rel(G-L[*i*-1, c_2]-{*d*})=G-{*b*}-{*d*})=0.59, which is better than R[*i*-1, *c*]=0.

As another example, consider link e_4=*c*; see row 4 in Table 2. For this case, Cost(G-{*c*})=27. Following Eq. (5), Algo-DP deletes link *c*, instead of the previous decision to delete because since Rel(G-{*c*})=0.76 is better than the reliability Rel(G-{*d*})=0.75 of deleting link *d*, for each *c*≥27 .

For the next example, we consider e_5=*f* with Cost(G-{*f*})=27 and Rel(G-{*f*})=0.75; link *f* should not be selected for each *c*≥27 because Rel(G-{*f*})=0.75 is worse than deleting link *c*, *i.e.*, R[i-1, c]=Rel(G-{c})=0.76 is larger than Rel(G-{*f*})=0.75. Since deleting link *f* together with links {*d*, *g*} for *c*=20 and 21 is feasible and improves the reliability, we set L[5, *c*]={*f*,*d*, *g*}and R[5, *c*]=0.659. Finally, e_6=*a*, e_7=h, and e_8=*e* are not selected for each *c*=20, …, 29 because each selection does improve the reliability of the topology. The optimal topology G_{opt} with the maximum reliability of 0.659, shown in Fig. 2, is obtained by deleting three links, *i.e.*, {*f*, *d*, *g*}.

4.2 Link Ordering

As later shown in our simulation results in Section 5, the optimality of our heuristic Algo-DP depends on the ordering of deleted links. To this end, we propose the following five possible links orders: 1) LO1: Increasing link cost c_i; 2) LO2: Decreasing link cost c_i; 3) LO3: Increasing link reliability r_i; 4) LO4: Increasing ratio c_i/r_i; LO5: Increasing ratio r_i/c_i. Table 3 shows an example of the different order criteria LO1-LO5 for the topology in Fig. 1. LO1 aims to exclude less costly links from the topology first. For example, criterion LO1 generates links (*a*, *c*, *f*, *h*, *e*, *g*, *b*, *d*) in increasing cost order, *e.g.*, link *a* with cost of 2 is ahead of link *c* with cost of 3. In contrast, LO2 aims to exclude the most costly links from the topology first. For example, criterion LO2 generates links (*d*, *b*, *g*, *e*, *h*, *f*, *c*, *a*) in decreasing cost order, *e.g.*, link *d* with cost of 6 is ahead of link *b* with cost of 5. On the other hand, LO3 aims to remove links from the least reliable first, and thus the generated topology is expected to contain links with higher reliability. As an example, LO3 produces link order (*d*, *b*, *g*, *f*, *h*, *c*, *e*, *a*), *e.g.*, link *b* is after link *d* because the reliability of the latter link of 0.7 is larger than the former link of 0.6. Finally, LO4 and LO5 aim to find the ration between the link cost and reliability. More specifically, LO4 removes links with the least ratio of c_i/r_i first, while in LO5, we remove links with least ratio of r_i/c_i first. As an example, LO4 produces link order (*a, h, c, f, e, g, b, d*), *e.g.*, link *h* is after link *a* because $c_h/r_h=3.75>c_a/r_a)=2.2$, but LO5 produces link order (*d, b, g, e, f, c, h, a*), *e.g.*, link *a* is after link *h* because $r_h/c_h=0.266<r_a/c_a=0.45$.

Note that one can use any sorting algorithm, *e.g.*, merge sort, to implement each link order with a time complexity of $O(|E|\times\log|E|)$ (Cormen, et. al, 1990).

Table 3. Example of different link orders

No.	LO1	LO2	LO3	LO4	LO5
1	*a*	*d*	*d*	*a*	*d*
2	*c*	*b*	*b*	*h*	*b*
3	*f*	*g*	*g*	*c*	*g*
4	*h*	*e*	*f*	*f*	*e*
5	*e*	*h*	*h*	*e*	*f*
6	*g*	*f*	*c*	*g*	*c*
7	*b*	*c*	*e*	*b*	*h*
8	*d*	*a*	*a*	*d*	*a*

5. Simulation and Discussion

We have implemented our Algo-DP in C language. We used Algo-DP on the 25 networks described in to evaluate the five link orders, LO1 to LO5, as well as to compare the performance of Algo-DP against DPA (Elshqeirat, et. al, July 2013). The networks contain four to 200 nodes and five to 298 links, with four to 2^{99} (*s*, *t*) paths. We use $CN_{n,Cmax}^{|V|,|E|}$ to denote a communication network with |V| nodes, |E| links, *n* (*s*, *t*) paths, and cost constraint C_{max}. For each of the 25 networks in (Elshqeirat, et. al, July 2013), we first assigned the r_i and c_i for each link e_i randomly, and set its cost constraint C_{max} no larger than 60% of the total link costs of the topology. Then, we sorted the links of each topology following the methods described in Section 4.3, *i.e.,* LO1, LO2, LO3, LO4 and LO5 and use Algo-DP to compare the performance of the five link orders. All simulations were run on Intel Core i5 with 2.53 GHz with 4 GB of RAM, running Linux (Ubuntu Core 11.10).

Table 4. The effects of link orders on the performance of Algo-DP

Network	**Rel(G) generated by Algo-DP for different link orders**						
	Random	**LO1**	**LO2**	**LO3**	**LO4**	**LO5**	**DPA**
$CN_{4,18}^{4,5}$	0.8640	0.8640	**0.9120**	0.8886	0.8640	0.8660	**0.9120**
$CN_{9,20}^{5,8}$	0.7237	0.7285	**0.9148**	0.8130	0.7285	0.8573	**0.9148**
$CN_{7,20}^{6,8}$	0.8238	0.8441	0.8330	**0.8705**	0.8441	0.8238	0.8330(-4.3%)
$CN_{13,20}^{6,9}$	0.7211	0.7285	0.7810	**0.7820**	0.7285	0.7810	**0.7930(+1.4%)**
$CN_{13,20}^{9,12}$	0.5889	0.5902	**0.5910**	0.5902	0.5902	**0.5910**	**0.5910**
$CN_{14,20}^{7,15}$	0.6690	0.8450	0.9470	**0.9568**	0.6690	0.9410	0.9470(-1.02%)
$CN_{18,20}^{11,12}$	0.8738	0.9110	0.8770	0.9174	0.8770	**0.9210**	**0.9210**
$CN_{18,20}^{9,13}$	0.6310	0.6310	**0.6910**	0.6869	0.6310	0.6869	**0.6910**
$CN_{20,20}^{8,12}$	0.4220	0.4368	**0.7780**	0.6566	0.4368	0.7020	**0.7780**
$CN_{24,20}^{8,12}$	0.7165	0.7287	0.7296	**0.8541**	0.7296	0.8430	0.8530(-0.128%)
$CN_{25,20}^{7,12}$	0.7300	0.7300	**0.9391**	**0.9391**	0.7300	**0.9391**	**0.9391**
$CN_{29,20}^{8,13}$	0.6864	0.5664	**0.7770**	0.7443	0.6964	0.7610	**0.7770**
$CN_{36,40}^{16,30}$	**0.8820**	**0.8820**	**0.8820**	**0.8820**	**0.8820**	**0.8820**	**0.8820**
$CN_{44,40}^{21,26}$	0.4368	0.5558	**0.6880**	0.6480	0.5324	0.6820	**0.6880**

$CN^{9,14}_{44,40}$	0.8794	0.9074	**0.9110**	0.9074	0.9074	0.9080	**0.9110**
$CN^{10,21}_{64,40}$	0.9662	0.9662	**0.9698**	**0.9698**	0.9662	**0.9698**	0.9640(-0.598%)
$CN^{18,27}_{281,40}$	0.7754	0.7859	**0.9020**	0.8916	0.7859	0.8790	**0.9020**
$CN^{13,22}_{281,40}$	0.9335	0.9335	0.9610	0.9477	0.9335	**0.9660**	**0.9660**
$CN^{20,30}_{780,40}$	0.8122	0.8780	**0.9010**	0.8780	0.8780	0.8420	**0.9010**
$CN^{17,25}_{136,40}$	0.9228	0.8969	**0.9420**	0.9228	0.8973	0.8973	**0.9470(+0.528%)**
$CN^{36,57}_{1262816,28}$	0.6522	0.6592	**0.6631**	0.6592	**0.6631**	0.6528	0.6592(-0.588%)
$CN^{36,60}_{538020,30}$	0.9336	**0.9387**	0.9342	**0.9387**	0.9336	0.9324	**0.9401(+0.149%)**
$CN^{48,77}_{64019921,38}$	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**
$CN^{40,58}_{524288,29}$	0.2159	0.2159	**0.2164**	0.2159	0.2159	0.2110	**0.2164**
$CN^{200,298}_{2^{99},149}$	0.2011	0.2235	**0.2407**	0.2164	0.2164	0.2235	0.2320 (-3.61%)
Total	**2**	**3**	**18**	**9**	**3**	**7**	**19**

Table 4 shows that the LO2 order is the best, because it gets 18 out of 25 best results; see numbers in bold in columns LO1 to LO5 and the last row in the columns that shows number of times each link order produces the best results among the five orders. Notice that LO3 and LO5 are the next best orders, producing nine and seven best results, respectively, while LO1 and LO4 are the worst performers with only three best results. The table also shows the result when using random order in which Algo-DP produces only two best results; note that the two best results are also produced from all other ordering criteria, *i.e.*, LO1-LO5. The results confirm the merits of using the proposed link orders. Further, to generate NT with the maximum reliability, we suggest running Algo-DP using each of the five link orders, and select the NT with highest reliability.

Table 4 shows the maximum reliability of feasible topology for each network generated by DPA (Elshqeirat, et. al, July 2013) ; see the last column. Note that the authors (Elshqeirat, et. al, July 2013) proposed five link orders, CR1, CR2, CR3, CR4, CR5, and Table 4 shows the best outcomes of DPA using all orders. As shown in Table 4, Algo-DP generates topology with higher 2-terminal reliability (up to 4.3% improvement) for 6 of 25 networks, as compared to DPA; see column DPA with (negative %). Further, Algo-DP is able to generate the same results as DPA for the other 15 topologies, while producing only three inferior results that has reliability no worse than 1.4%; see column DPA with (positive %). We need to re-emphasize that, while Algo-DP considers |E| links to generate its DP table, DPA potentially can use $O(2^{|E|-|V|+2})$ (*s*, *t*) paths of the network to generate its DP table. The difference is significant for large sized CN, *e.g.*, $CN^{200,298}_{2^{99},149}$ that has 298 links with 2^{99} paths. Thus, in addition to generate more NT that has higher reliability, Algo-DP using the five link orders also significantly outperforms DPA in terms of computational time complexity.

6. Conclusion

We have formally defined an NP-hard network topology design problem, called NTD-RC, to generate a network topology with maximized 2-terminal reliability, subject to a cost constraint C_{max}. We have proposed a heuristic dynamic programming method, called Algo-DP, to solve the problem and five different link ordering methods to optimize its results. Our simulations on 28 topologies show the effectiveness of Algo-DP and the benefits of using the link orders to find the best results. While Algo-DP does not guarantee to always generate optimal topology, the simulations show that it finds better results as compared to the most recently proposed approach, DPA (Elshqeirat et. al, July 2013). More specifically, Algo-DP generates topology with the same or better 2-terminal reliability (with up to 4.3% improvement) on 92% of the 25 network topologies. Since Algo-DP is computationally more efficient as compared to the existing approach, it becomes the obvious choice for used in large network topology design. For future direction, we plan to use a similar approach to solve the NTD-CR problem.

References

Abo ElFotoh, H., & Al-Sumait, L. (2001). A neural approach to topological optimization of communication networks, with reliability constraints. *IEEE Trans. Reliability, 50*(1)*,* 397-408. https://doi.org/10.1109/24.983401

Atiqullah, M., & Rao, S. (1993). Reliability optimization of communication networks using simulated annealing. *Microelectronics and Reliability*, *33*(1), 1303-1319. https://doi.org/10.1016/0026-2714(93)90132-i

Bhupesh, K., Makarand, S., & Misra, K. (2008). Optimal Reliability Design of a System. In: Misra K. B. (eds), *Handbook of Performability Engineering,* Springer, London. https://doi.org/10.1007/978-1-84800-131-2_32

Chelouah, R., & Siarray, P. (2000). Tabu Search applied to global optimization. *European Journal of Operational Research, 123*(1), 256-270. https://doi.org/10.1016/s0377-2217(99)00255-6

Cormen, T., Leiserson, C., & Rivest, R. (1991). *Introduction to Algorithms*, Cambridge: The MIT Press. https://doi.org/10.2307/2583667

Dengiz, B., & Altiparmak, F. (2009). Cross entropy Approach to Design of Reliable Networks. *European Journal of Operation Research (EJOR), 199*(1), 542-552. https://doi.org/10.1016/j.ejor.2008.11.022

Dengiz, B., Altiparmak, F., & Smith, A. (1997). Efficient optimization of all-terminal reliable networks. *IEEE Transactions on Reliability*, *41*(1), 18-26. https://doi.org/10.1109/24.589921

Dengiz, B., Altiparmak, F., & Smith, A. (1997). Local search genetic algorithm for optimal design of reliable networks. *IEEE Trans. Evolutionary Computation*, *1*(3), 179-188. https://doi.org/10.1109/4235.661548

Elshqeirat, B. (2015). Optimizing reliable network topology design using dynamic programming. *Ph.D. dissertation*, Dept. Comp., Curtin Univ., Perth, Australia.

Elshqeirat, B., Soh, S., Rai, S., & Lazarescu, M. (2014). A Dynamic Programming Algorithm for Reliable Network Design. *IEEE Trans. Reliability*, *63*(2), 443-454. https://doi.org/10.1109/tr.2014.2314597

Elshqeirat, B., Soh, S., Rai, S., & Lazarescu, M. (2015). Topology Design with Minimal Cost Subject to Network Reliability Constraint. *IEEE Trans. Reliability*, *64*(1), 118-131. https://doi.org/10.1109/tr.2014.2338253

Elshqeirat, B., Soh, S., Rai, S., & Lazarescu, M. (August 2013). Dynamic Programming for Minimal Cost Topology with Two Terminal Reliability Constraint. *Proc. IEEE Asian Pacific Conference on Communication,* Indonesia. https://doi.org/10.1109/apcc.2013.6766047

Elshqeirat, B., Soh, S., Rai, S., & Lazarescu, M. (December 2013). Dynamic Programming for Minimal Cost Topology with Reliability Constraint. *Journal of Advances in Computer Networks, 1*(4), 286-290. https://doi.org/10.7763/jacn.2013.v1.57

Elshqeirat, B., Soh, S., Rai, S., & Lazarescu, M. (July 2013). A Practical Algorithm for Reliable Communication Network Design. *International Journal of Performability Engineering*, *9*(4), 397-408.

Gen, L. (2006). A Self-controlled Genetic Algorithm for Reliable Communication Network Design. *Evolutionary Computation IEEE Congress*, 640-647. https://doi.org/10.1109/cec.2006.1688371

Khan, S., & Engelbrecht, A. (2008). A Fuzzy Ant Colony Optimization Algorithm for Topology Design of Distributed Local Area Networks. *Proceedings of IEEE Swarm Intelligence,* 1-7. https://doi.org/10.1109/sis.2008.4668303

Konak, A. & Smith, E. (1999). A Hybrid Genetic Algorithm Approach for Backbone Design of Communication Networks. Proceedings of the 1999 Congress on Evolutionary Computation -CEC99*, 1817-1823. https://doi.org/10.1109/cec.1999.785495*

Konak*,* A*.,* & Smith, E. (2005). Designing Resilient Networks using a Hybrid Genetic Algorithm Approach. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. https://doi.org/10.1145/1068009.1068217

Mutawa, A. Alazemi, H., & Rayes, A. (2009). A novel steady-state genetic algorithm approach to th91e reliability optimization design problem of computer networks. *International Journal of Network Management*, *19*(1), 39-55. https://doi.org/10.1002/nem.687

Papagianni, C. Papadopoulos, K., & Pappas, C. (2008). Communication Network Design Using Particle Swarm Optimization. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 915-920. https://doi.org/10.1109/imcsit.2008.4747351

Reichelt, D., & Rothlauf, F. (2005). Reliable Communication Network Design with Evolutionary Algorithms. *International Journal of Computational Intelligence and Applications*, World Scientific Publishing, *5*(2), 251-266. https://doi.org/10.1142/s146902680500160x

Saad, A., Khan, S., Mahmood, A. (2018). A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. *Swarm and Evolutionary Computation, 38*(1), 187-201. https://doi.org/10.1016/j.swevo.2017.07.010

Shao, F., Shen, X., & Ho, P. (2005). Reliability optimization of distributed access networks with constrained total cost. *IEEE Trans. Reliability*, *54*(3), 421-430. https://doi.org/10.1109/tr.2005.853440

Soh, S., & Rai, S, (1991). CAREL: Computer aided reliability evaluator for distributed computer networks. *IEEE Trans. Parallel and Distributed Systems, 2*(1), 199-213. https://doi.org/10.1109/71.89065

Won, J., & Karray, F. (2010). Cumulative Update of All-Terminal Reliability for Faster Feasibility Decision. *IEEE Trans. Reliability, 59*(3), 551-562. https://doi.org/10.1109/tr.2010.2055924

Yeh, S., Lin, J., & Yeh, W. (1994). New Monte Carlo method for estimating network reliability. *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, 723-726.

Zakir, M., & Abd-El-Barr, M. (2002). Enumerative techniques in topological optimization of computer networks subject to fault tolerance and reliability. *In Proceedings of the 14th IASTED international conference on parallel and distributed computing and systems (PDCS-2002), Cambridge,* 123-128.