ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcs226932020.pdf
https://doi.org/10.30534/ijatcse/2020/226932020

On Minimizing Cost of Reliable Network Topological Design using a Practical DP Approach

Basima Elshqeirat
Department of Computer Science, the University of Jordan,
P.O. Box 11942, Amman, Jordan.
B.shoqurat@ju.edu.jo

ABSTRACT

This paper addresses an NP-hard problem called NTD-CR that using dynamic programming (DP) scheme. This problem referred to as Network Topology Design with minimum Cost subject to a Reliability constraint. It designs a minimal-cost communication network topology that satisfies a pre-defined (s, t) terminal reliability constraint. Our DP approach, called DPCR-L, mainly select the set of possible links to be deleted from the original network to generate an optimal Network Topology. The NTD-CR design problem aims to find a Network Topology that has minimal cost with the required reliability for the network. Five link-ordering criteria are proposed to improve the performance of DPCR-L. Each greedy heuristic order allows DPCR-L to enumerate the selected deleted links, which improves the time complexity while producing near optimal topology. Extensive Simulations based on different benchmark networks of various sizes are used to compare DPCR-L with existing state-of-the-art techniques and show the merits of using the ordering methods, and the effectiveness of our algorithm. Our simulations show that DPCR-L produces 93% optimal results. Interestingly, of the non-optimal results, DPCR-L produces a network with reliability no worse than 5.38% off optimal, and most of the non-optimal results have a lower cost than that for optimal up to 0.17%. Typically, for the most of the network topologies, DPCR-L generates NT with the same or better 2-terminal reliability measure and speeds up its running time up to 31.71%. Furthermore, simulation results on large size networks show that DPCR-L speeds up the process with up to (47.28%) compared to the recent existing approach. Finally, the results present the applicability of DPCR-L on networks containing a large number of links and demonstrated better performance and computationally more efficient than other existing algorithms.

Key words*:* Dynamic programming; Network optimization; Network reliability; Communication Network; 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 For example, critical applications (*e.g.*, emergency services, critical time system, rescue, and military operations) must run on a network topology with guaranteed minimum reliability so that they can operate without interruption, even in the presence of component failures [1]. The design of network topology is an important part of a network design [1, 2]. Therefore, it is crucial to design network topology that can meet its applications' QoS requirement, since in general, optimal design of network topology directly affects network QoS [2]. In this work, a single-objective optimization problem for network topology design problem is considered , this mean that we consider only one objective and one condition for the design process [3]. However, constructing a network with higher QoS such as reliability incurs higher installation cost, since for example link reliability is directly proportional to its installation cost. Therefore, the most suitable set of links such that the resulting best layout meets its cost objective and required reliability must carefully select by a network designer. We called this situation a network topology design with cost objective and reliability constraint (NTD-CR) problem. Specifically, given (a) locations of the various computer centers (nodes), (b) their connecting links, (c) each link's reliability and cost, and (d) the required operational reliability of the network, the final solution will have the best set of links such that the resulting layout meets its required (*s*, *t*) terminal reliability while minimizing its installation cost.

In this paper, we consider the (*s*, *t*) terminal reliability [4], , as the measure of reliability (R), which is the probability that at least one simple (*s*, *t*) path in the network is functional between source node *s* and the terminal node *t*, it is also called 2-terminal reliability. Further, some applications must run on a topology with a guaranteed minimum reliability, to properly operate. However, constructing a reliable topology incurs higher installation cost as we said above. Therefore, for such applications, the topology design emphasizes on minimizing the network installation cost subject to the required reliability level. In practice, however, when the network service provider or decision maker has a limited budget to build the network, the aim will be to produce a network topology from the selected links with the maximum reliability subject to the cost constraint; we call this as NTD-RC problem. Both these problems (NTD-CR an NTD-RC) are NP-hard [5]. Obviously, heuristic and/or approximation solutions must be use to design large sized topologies that contain many nodes, links and (*s*, *t*) paths.

2. RELATED WORK

There are some proposed techniques that find optimal or approximately optimal solutions for the NTD-CR problem [6, 7]. Jan, *et al.* [8] considered a network G whose links have the same reliability values and developed an algorithm that combines decomposition, B&B techniques to find an optimal solution. Later, Koide *et al.* [10] generalized the problem in [8] for graph G with non-homogeneous link reliabilities, and developed another B&B algorithm to solve the problem. The B&B approaches are computationally expensive, and thus are suitable only for small sized networks with up to nine nodes [8, 9].

Kumar, *et al.* [10] have developed a GA-based approach to solve NTD-CR that includes two additional constraints, *i.e.*, diameter and average distance, and applied it to four test networks with up to nine nodes. Although the problem in [10] is a superset of NTD-CR, its solution cannot be used to solve the NTD-CR problem because the problem considers only links with identical reliability and cost. Deeter and Smith [11] presented a GA approach to solving the NTD-CR problem. However, their solution considers alternative link reliabilities, and thus cannot be used to solve our NTD-CR problem in which each link may have different reliability values. Dengiz, *et al.* [12] proposed a heuristic GA approach, called NGA, to solve the NTD-CR problem. In [13], the same authors have developed another GA-based solution, called Local search GA (LS-NGA), using a special encoding structure, crossover, and mutation operators. These GA methods yield poor quality solutions for networks with more than 10 nodes [14]. Later, Lin and Gen [15] proposed a self-controlled GA to solve the NTD-CR problem. However, these GA methods require the development, coding, and testing of a problem-specific GA, complicating the solution process [12, 15]. Mutawa, *et al.* [16] proposed a steady-state GA, and Shao, *et al.* [17] proposed an algorithm, called a shrinking and searching algorithm, to solve NTD-RC problem that maximizes network reliability under a cost constraint, which is a related NTD-CR problem, discussed before in this Section. Ramirez-Marquez and Rocco [14] have presented a population-based heuristic approach called the probabilistic solution discovery algorithm. However, their approach is shown less effective compared to the more recent approach in [5], who developed a NN heuristic algorithm, and the authors in [12] used an artificial NN for the NTD-CR problem. As stated in [14], while the NN and artificial NN algorithms produce good results, they use a long procedure that needs extensive time and significant parameter tuning. A deterministic version of simulated annealing (SA) was used by Atiqullah and Rao [18] to find the optimal design of small networks, *i.e.*, five nodes or less. Pierre, *et al.* [19] also used SA to find optimal designs for packet switch networks where delay and capacity were considered, but reliability was not. Recently, a new metaheuristic called Cross-Entropy method was developed for the NTD-CR problem [20]. In addition, Papagianni, et al. [21] proposed a Multiple TS algorithm was used to solve the NTD-CR problem with 19 nodes; however, the algorithm may not reach the global optimum solution in reasonable computation time when the initial solution is far away from the region where the optimal solution exists. In [22], a Binary Decision Diagram (BDD) is used to solve the same design problem for networks containing up to 81 nodes. This approach is based on a decomposition of Boolean functions called *Shannon decomposition*. BDD structure is a compact, implicit representation of the entire set of the functioning and failing network states. Dengiz, et al. [23] proposed a hybrid approach based on Ant Colony Optimization and Simulated Annealing, called ACO-SA, for the NTD-CR problem for networks with up to 50 nodes. Note that the results in [7] for the related Dynamic programming approach show that the DP techniques produced better results as compared to the BDD approach in [22] and ACO-SA approach in [23].

As a summary, the existing algorithms that generate approximation solutions are mainly based on meta-heuristic techniques, such as Genetic Algorithm [10, 11, 14, 22], Neural Network [5, 21], Swarm Particle [24] [21], Simulated Annealing (SA) [17, 18] and Ant Colony Optimization (ACO) [23]. While the meta heuristic-based algorithms may significantly reduce time complexity, they still require numerous iterations to converge and thus use a considerable computational effort to produce near optimal solutions [23]. Therefore, a more time efficient heuristic approach that can produce better results is still needed, especially for use in large scale networks. At this end, most of the existing approaches either lack the necessary precision to generate an acceptable solution or have expensive computational effort and time [7]. Here, we developed a heuristic based approach that has better performance and computationally more efficient than other existing algorithms to solve NTD-CR problem, discussed in Simulation Section.

Note that, for the NTD-CR problem, each iteration includes a reliability computation of the approximated topology to be tested against the required constraint. Because reliability evaluation, using both exact or approximation methods, is computationally expensive (a typical Monte Carlo (MC) simulation iterates 10^6 times), which generally uses a considerable computational effort [6]. For example, Deeter and Smith [11] proposed a GA based approach that uses MC simulation to calculate the reliability of each candidate solution, and thus, for a typical GA solution with a population size of 6000 and 40 generations, it needs to run MC 240000 times. Therefore, the approaches are computationally expensive for use in large networks.

Elshqeirat, *et al.* [6] proposed a dynamic programming (DP) formulation and implementation, called DPCR-P, to solve the NTD-CR problem from all (*s*, *t*) simple paths in the network. They also presented two different path-orders to improve the effectiveness of DPCR-P. These two different path-orders dynamically allow DPCR-P to generate only *k*≥1 paths from the network and it is stop if there is insignificant improvement when adding path *k*+1 to the resulting topology's cost. Third, we describe how DPA can use only *k* (*s*, *t*) paths to reduce its computation time while producing similar results as compared using all (*s*, *t*)simple paths of the network. Furthermore, [7]solved NTD-CR problem by used DPCR-ST approach to generating the NT using a sequence of spanning trees to maximize *all*-terminal reliability. The approach proves that if we generate an optimal sequence of spanning trees an inputs, then DPCR-ST is able to generate optimal NT Note that the first version of the approach requires all spanning trees of the network to minimize the cost. Further, they proposed a second version of the approach that uses only *k* spanning trees sorted to improve DPCR-ST's time efficiency while producing similar results. Moreover, authors in [7] described three spanning tree order criteria to heuristically generate the best sequence of spanning trees that allow DPCR-ST to produce near optimal results and also showed that producing optimal order of spanning trees is consider as NP-complete.

The aforementioned DP approaches in [6, 7] have been shown good results in addressing our NP-hard problem, NTD-CR, but both approaches will generating all possible (*s*, *t*) paths or spanning trees to produce each feasible topology solution in the worst case,. Thus, in general, for large networks contains an exponential number of (*s*, *t*) paths and spanning trees , *i.e.*, $O(2^{|E|-|V|+2})(s,t)$ simple paths and $O(|V|^{|V|})$ spanning trees such an approach is infeasible [25]. Our main goal in this paper is proposing a new version of DP approach that called DPCR-L, which generating each optimized topology result by deleting unusefull links from the original network (graph model) without any needs of generating all possible (*s*, *t*) paths or spanning trees. Thus, our proposed approach DPCR-L will consider only *O*(|E|) links of the network in the worst case. As a consequence, this is generally will need less computational time and effort, so it is computationally more efficient than other existing DP algorithms such as DPCR-P and DPCR-ST for NTD-CR problem. Furthermore, five link-orders criteria are proposed and utilized by DPCR-L to further improve results.

Furthermore, the authors of [1, 26] have proposed a *dynamic programming algorithm* to solve a related NTD problem, called NTD-RC, to construct a topology that maximized 2-terminal or all-terminal reliability subject to a cost budget constraint. However, these DP approaches cannot be used directly to solve the NTD-CR problem because the two related problems require two different dynamic programming formulations, and thus need different solutions [25]. Further, to maximize the reliability, for each entry of a dynamic programming table, the NTD-RC problem needs to compute an exact reliability value, which is a very time consuming step because computing the reliability, in general, is known to be NP-hard [27]. On the other hand, to minimize network cost, the NTD-CR problem needs to generate only an approximated reliability, which can be solved using a significantly faster heuristic technique such as Monte Carlo Simulation [28]. Please note that DP approach proposed in ELSHQEIRAT *et al.* 2013b (ELSHQEIRAT *et al.* 2014a), called DPA,(DPA-1) needs generating all possible (*s*, *t*) paths (spanning trees) to produce each feasible topology solution to solve NTD-RC problem with consider 2-terminal reliability (all terminal reliability), respectively. Furthermore, another recent DP approach proposed, called Algo-DP, to solve the related problem NTD-RC in Elshqeirat *et al.* 2018. Algo-DP generating each optimized topology result by deleting set of links from the original network (graph model) without any needs of generating all possible (*s*, *t*) paths or spanning trees. As we said above, we can't directly use these DP approaches that solve NTD-RC problem to Solve NTD-CR problem because the two related problems require two different dynamic programming formulations, and thus need different solutions [25].

The contribution of this paper is threefold:

1- It presents DPCR-L approach that utilizes dynamic programming (DP) formulation, solution and its implementation, to solve the NTD-CR problem by deleting selected set of links from the original network. As we said before, Authors in [6] also have used DP technique called DPCR-P to solve the NTD-CR problem, but, their algorithm potentially requires generating all (*s*, *t*) simple paths. 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 [6]. In contrast, our DP approach significantly reduces the time complexity because it deletes sequentially only up to |E| links to solve the NTD-CR problem.

2- DPCR-L approach in this paper presents five link-ordering criteria, *i.e.*, LO1- LO5, discussed later in Section 4.4; DPCR-L utilizes each link-ordering criterion to optimize the generated NT by determining the order of link deletions from the original topology.

3- 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 [6]. Specifically, DPCR-L generates NT with the same or better 2-terminal reliability on all but only two (with about 1.8% worse cost) of the 20 network topologies, but it speeds up the running time up to 31.71% compared to the existing approach in [6], simulations show that DPCR-L produces 93% optimal results. Interestingly, of the non-optimal results, DPCR-L produces a network with reliability no worse than 5.38% off optimal, and most of the non-optimal results have a lower cost than that for optimal up to 0.17%. In large size networks, results show that DPCR-L speeds up the

process up to (47.28%) compared to the recent existing approach in [6], and prove that DPCR-L is computationally more efficient than existing algorithms. These results present the applicability of DPCR-L on networks containing a large number of links and indicate that DPCR-L demonstrated better performance than other existing algorithms. Please note that we didn't compare our results with DPCR-ST approach in Elshqeirat *et al.* 2014b, because their DP approach measure all-terminal reliability while in our problem we measure (*s*, *t*) reliability similar to the problem presented by DPCR-P approach in Elshqeirat *et al.* 2013a.
The layout of the paper is as follows. Section 3 presents brief description to the network model and Notations. Section 4 formulates the NTD-CR problem, provides assumptions, and describes the proposed solution with the time complexity and different order criteria. Section 5 shows the simulation results. Finally, conclusion of the paper presented in Section 6.

3. NETWORK MODEL AND NOTATIONS

A communication network (CN) can be modelled by a probabilistic bidirectional simple graph G = (V, E), in which each node $v_i \in V$ represents a network component (*e.g.*, router, computer site) and each edge $e_j \in E$ represents the connecting media (*e.g.*, communication link) between the network components. We assume that (a) all locations of the various computer centers (nodes), (b) their connecting links, (c) each link's reliability and cost, and (d) the required operational reliability of the network are given. Each e_j has c_j and r_j , $c_j>0$ is the cost to install e_j, and r_j , $0 \le r_j \le 1$, is probability that e_j is functioning (UP) [10]; Let *m* be the total number of edges in G, *i.e.*, *m*=|E|. We assume that all nodes are always UP, use no setup costs, and edge failures are independent and without repair. Fig. 1 shows a graph model of a CN that has six nodes and eight links.**Error! Reference source not found.** provides the c_j and r_j values for each edge e_j.

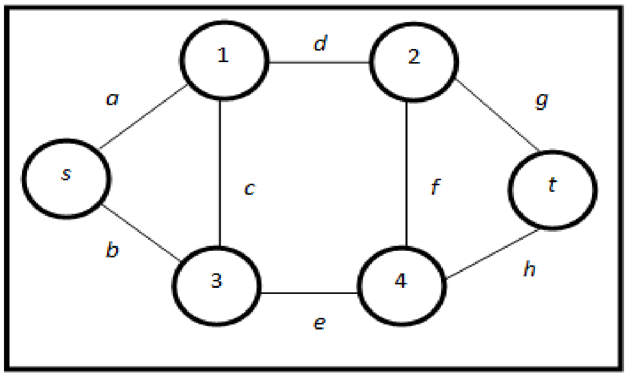

Figure 1: Network Example

Table1: Link Weight for Network

Link e_j	Link Weight
	(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 function of a network topology or graph G, Cost(G), is the summation of all c_j for each e_j in G. R(G) is the function that calculate the probability if there is at least one operational path between source node (*s)* and terminal node (*t*). In general, computing Rel(G) is an NP-hard problem [28]. As described in Section 4.3, the Monte Carlo Simulation in [28] is used to approximately compute Rel(G).

1.1 Design Problem

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

$$\text{Minimize} \sum c_j\ x_j \qquad (1)$$

$$\text{Subject to Rel } (G_i=(V, E_i)) \ge R_{min} \qquad (2)$$

Let us define a network topology G *feasible* when Rel(G) ≤ R_{min}. Equation (1) calculates the minimum cost of a network topology G_i=(V, E_i) that contains links E_i=E–{e_j | X_j=0}; *i.e.*, E_i is a set of selected links from Equation (2) that form G_i that has a reliability of at least R_{min}. One may solve the NTD-CR problem by generating each possible set of links in (2) that form G_i. Then, calculate Cost(G_i) for each G_i that has reliability Rel(G_i)≥R_{min}, and use Eq. (1) to select a G_i with the minimum cost as G_{min}. Unfortunately, this brute force solution, called BF-1 **requires** generating $O(2^{|E|})$ possible link selections, *i.e.*, the G_i. Further, the reliability calculation in (2) for each G_i requires exponential time; thus BF-1 is feasible only for designing small topologies. Our work in this paper proposes a heuristic algorithm, described in Section 4.1, which generates E_f, and thus G_f, by selectively removing links in E while satisfying Eq. (1) and (2).

For Example, consider the CN in Figure 1, for NTD-CR problem. If we have R_{min}=0.65, then Figure **2** shows the optimal network topology, G_{opt}, with Rel(G_{opt}) =0.659 and Cost(G_{opt})=17; after delete links *d*, *f* and *g from network G* to obtain G_{opt}.

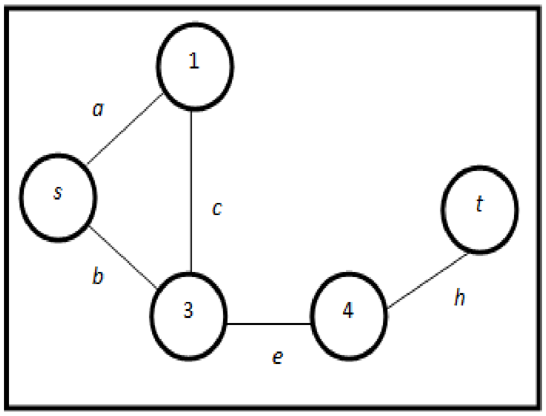

Figure 2: Optimal Solution for Network

4. DYNAMIC PROGRAMMING APPROACH FOR NTD PROBLEM

4.1 Dynamic Programming Formulation for NTD-RC

Let G_i=(V, E_i⊆E) be its induced graph and LX*i*, for *i*=1, 2, …, *m*, be a set of links sequence selected from *i* links in (e_1, e_2, …, e_i), recall that *m*=|E|, In other words, LX_i is a set of deleted links in (e_1, e_2, …, e_i) from G, *i.e.*, E_i = E – LX_i; . which mean that our G_i will not contain any link in LX_i. Note there are $O(2^{|Ei|})$ different possible LX_i and $0 \leq |LX_i| \leq i$. NTD-CR aims to delete set of links in (e_1, e_2, …, e_m) such that $Rel(G_m) \geq R_{min}$ and $Cost(G_m)$ is the minimum. We said that a solution or NT, G_i, is *feasible* if its (*s*, *t*) reliability, $Rel(G_i) \geq R_{min}$. Otherwise, it is a *non-feasible* solution. For example, NT in Figure 1 with R_{min} =0.65, LX_8=(*d*, *f*, *g*) can be consider a feasible solution because $Rel(G_8)=0.659 \geq R_{min}$.
Let DP[1 .. *m*, $Ř_G$ .. $Ř_{min}$] be a 2-dimensional dynamic programming (DP) table, where $Ř_G$= *round*(δ×Rel(G)), *i.e.*, the Reliability of the original network with no link deletion and $Ř_{min}$=*round*(δ×R_{min}), for a positive integer multiplier δ and a function *round*(•) that returns the closest integer value of (•). For example, the function returns $Ř_{min}$=92 ($Ř_{min}$=93) when we set δ=100 and R_{min}=0.9216 (R_{min} = 0.9261).

Each element DP[*i*, *ř*], for *i*=1, 2, …, *m*, *ř*= $Ř_G$, …, $Ř_{min}$, stores four pieces of information: a cost 0≤c[*i*, *ř*]≤Cost(G), a sequence of links L[*i*, *ř*]⊆E, a reliability $Ř_{min}$<R[*i*, *ř*]≤$Ř_G$, and an integer index $Ř_{min}$≤*J*[*i*, *ř*]≤$Ř_G$. Let C[*i*, *ř*]be Cost(G_i), for *ř*= $Ř_G$, …, $Ř_{min}$ be the minimum cost of G_i subject to $Rel(G_i) \geq$ *ř*. In essence, the columns of DP table partition the reliability constraint R_{min} into δ consecutive reliability constraints, *i.e.*, R_{min}/δ, (2×R_{min})/δ, …, δ×R_{min}/δ= R_{min}. In other words, each column index *ř*=0, 1, …, $Ř_{min}$, corresponds to a reliability constraint *r*=0, 1/δ, …, ($Ř_{min}$/δ)≈R_{min}, *i.e.*, *r*=*ř*/δ and *ř*=*round*(δ×*r*). Each DP[*i*, *ř*] is used to store four pieces of information of each selected topology G_i that has $Rel(G_i) \geq r$. C[*i*, *ř*] is the cost of G_i=(V, E – LX_i) with total reliability at least *ř*, and minimum cost, easily we set C[*i*, *ř*]=Cost(G_i), R[*i*, *ř*]=Rel(G_i).. Note that we store the minimum cost of G_m subject to $Rel(G_m) \leq R_{min}$ in C[*m*, $Ř_{min}$]. We try to find the most optimal LX_m whose deletion from E will produces the optimal NT, *i.e.*, G_{opt}. Let L[*i*, *ř*]=LX_i such that $Rel(G_i) \geq r$, *where r*=*ř*/δ, and C[*i*, *ř*]=Cost(E-L[*i*, *ř*]). W set each *J*[*i*, *ř*]=*ř*1, for columns *ř*1≤*ř*≤*ř*2 in row *i* that have the same reliability value. For example, we store *J*[*i*, *ř*]=38 at columns *ř*=0 to *ř*=38 if R[*i*, 0]=R[*i*, 1]=…=R[*i*, 38]. Note that we set *J*[*i*, *ř*]=*ř* when *ř*1=*ř*2, *i.e.*, when the length of the range is one. DPCR-L using the following three equations to calculate C[*i*, *ř*]:

$$C[i, ř]=0; \text{ if } i=1 \text{ and } Rel(G-e_1) < r \quad (3)$$

$$C[i, ř]=Cost(G-e_1); \text{ if } i=1 \text{ and } Rel(G-e_1) \geq r \quad (4)$$

$$C[i, ř]=Min(C[i-1, ř], Cost(G-L[i-1, ř_j] - \{e_i\})); \quad (5)$$
$$\text{if } i>1, 1 \leq ř_j \leq ř \text{ and } Rel(G-L[i-1, ř_j]-\{e_i\}) \geq r$$

Note that $Rel(G-e_1)$ present network reliability of G after deleting link e_1 and $Cost(G-e_1)$, refer to the cost of network G after deleting link e_1 and *r*=*ř*/δ. The link selection starts from the first link e_1. In Eq. (3), when $Rel(G-e_1) \geq r$, link e_1 cannot be deleted since the resulting NT is non9c-feasible, *i.e.*, does not meet the constraint *r*, and thus we set C[1, *ř*]=0. In contrast, if the Reliability of G without e_1 satisfies the reliability requirement *r*, *i.e.*, it is a feasible solution, in Eq. (4), e_1 is deleted, giving C[1, *ř*] =Cost(G-e_1).

In case of delete each link e_i, together with some previously deleted links in LX_{i-1}, Equation (5) will be used, *i.e.*, Rel(G-L[*i*-1, $ř_j$]-{e_i})≥*r*, for each possible *j*=*J*[*i*, *ř*]= $Ř_g$ .. $Ř_{min}$, and G_i has the minimum cost. In case of e_i is not deleted, the potential minimum cost would come from deleting links in (e_{i+1}, e_{i+2},…, e_m) with unchanged reliability *ř*; *i.e.*, C[*i*, *ř*]= C[*i*-1, *ř*]. However, if e_i is deleted, the resulting cost would be Cost(G-(L[*i*-1, $ř_j$]-{e_i}). Thus, Eq. (5) sets C[*i*, *ř*] to the minimum between the two potential cost values. When the two options produce the same cost, our implementation selects the one with higher reliability.

4.2 Dynamic Programming Proposed Solution

Shows the pseudo code of our proposed DP algorithm, called DPCR-L, that directly applies the DP Eq. (3) to (5). DPCR-L *implicitly* constructs a DP table of size *m*× $Ř_{min}$ for a G=(V, E) that contains *m*=|E| links with reliability constraint R_{min}. However, as shown in, DPCR-L keeps only two consecutive rows, called *row*1 and *row*2, and therefore we require only a table of size 2×$Ř_{min}$. Typically, by using the information in C[2, *ř*] and L[2, *ř*] in *row*2, DPCR-L computes C[1, *j*] and L[1, *j*] in *row*1 for all columns *ř*. After copying the contents of *row*1 to *row*2, DPCR-L repeats the steps until all links are considered. Line 1, Line 2 to 5, Line 6 to 15, are used to implements Eq. (3), Eq. (4), Eq. (5), respectively. Lines 16 to 18 copy *row*1 to *row*2. Function Cost(X) in computes the total cost of the union of links in network X=(X-L[*i*-1, r_j]-{e_i}), and function Rel(X) calculates the reliability of the network X using the Monte Carlo Simulation [28].

4.3 DP time complexity

The time complexity of DPCR-L can be computed as follows.

The Rel(X) function is calculated using any heuristic technique [12], exact (*s*, *t*) reliability method [27], or approximation (bounding) method [13]. In this paper, In this paper, we use Monte Carlo simulation with time complexity $O(b\times|V|^4)$ [11] to estimate Rel(X) of each candidate network; *b* is the number of replication and |V| are the numbers of replications and nodes respectively [27]. Notice that Rel(X) is used only for each different *j* in each row *i* and if $\text{Rel(X)}\geq\check{R}_{min}$ then Rel(X) is calculated only.

The Cost(X) Function requires all unique links used in network X. Cost(X) will returns the sum of links that are in network X without the set of the links in $L[i\text{-}1, \check{r}] + \{e_i\}$ for each *c*. Using the bit implementation [27], one requires only one bit OR and one bit XOR operation to obtain the links in X that are not in $L[i\text{-}1, \check{r}] + \{e_i\}$, and thus for any X, Cost(X) can be computed in $O(|E|)$. DPCR-L uses the function at most once for every table entry, and therefore the worst case time complexity for using the function is $O(E|^2\times\check{R}_{min})$. Let ψ be the total number of cases where $\text{Rel(X)}\geq\check{R}_{min}$. Hence the time complexity of using Rel(X) is $O(\psi\times b\times|V|^4)$.Thus, in the worst case, DPCR-L requires $O(\psi\times b\times|V|^4+|E|^2\times\check{R}_{min})$.

4.4 Improving the Efficiency of DPCR-L using Link Ordering

As shown by our simulation results in Section 5, the ordering of deleted links will determine the optimality of our heuristic DPCR-L for its input. We propose the following five possible link-orderings: 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 shows an example of the different ordering criteria LO1-LO5 for CN in Figure 1. For example, LO1 aims to exclude less costly links from the topology first. Link-Order Criterion LO1 produces links (*a*, *c*, *f*, *h*, *e*, *g*, *b*, *d*) in increasing cost order, *e.g.*, link *a* with cost of 2 then link *c* with cost of 3. You can use any sorting algorithm, *e.g.*, merge sort, for any link order with time complexity of $O(|E|\times\log|E|)$ [29],[30],[31].

5. SIMULATION AND DISCUSSION

We have implemented our DPCR-P in C language. In Section 5.1, we consider DPCR-L on the 20 networks described in [25] to observe the effects of using the five link-orders, LO1 to LO5, on the effectiveness of our algorithm. Moreover, to compare the performance of DPCR-L against DPCR-P [6]. In Section 5.2, the DPCR-L is utilized on 100 networks with a known optimal network to gauge DPCR-L's effectiveness. Finally, in Section 5.3, the DPCR-L is utilized on large grid networks that contain up to 2^{99} paths to evaluate its efficiency and effectiveness.

```
                         DPCR-L
** r_j is the reliability of selecting j^th solution in row 2  and j is
 the different solutions
1.  Initialize m= |E|, L[2,ř]={} and C[2, ř]=0 for ř>Rel (G - e_1) // Eq.
(3)
2.  for (ř← Rel(G) down to Rel (G - e_1)) do // Eq. (4)
3.      L[2, ř] ← {e_1}
4.      C[2, ř] ← Cost(G- e_1)
5.  end for ř
6.    for   (i from 2 to m)    // Eq. (5)
7.      for ( ř  from   Ř_min to Rel(G) )
8.          while( 1≤r_j≤r j  and  != 0  )  // **
9.              if  (Cost(G-L[2, r_j ]-{e_i}<C[2,  ř ])
10                  L[1,  ř ] = L[2, r_j ]+ {e_i}
11.                 C[1,  ř ] =Cost(G-L[2, r_j ] - {e_i})
12.             else
13.                 L[1,  ř] ←  L[2,  ř]
14.                 C[1,  ř] ← C[2,  ř]
15.           j - -
16.     for (K ← Ř_min to Rel(G)) //copy content row 1 to row 2
17.         L[2, k ] ← L[1, k ]
18.          C[2, k ] ← C[1, k ]
```

Figure 3: DPCR-L Pseudo code

Table 2: Example of different link orders

No.	Link-1	Link-2	Link-3	Link-4	Link-5	Link-6	Link-7	Link-8
LO1	a	C	f	h	e	g	b	d
LO2	d	B	g	e	h	f	c	a
LO3	D	B	g	f	h	c	e	a
LO4	A	H	c	f	e	g	b	d
LO5	D	B	g	e	f	c	h	g

5.1 The Effect of link Orderings on DPCR-L

The DPCR-L is utilized on 20 networks in [9] to observe the effects of using LO1- LO5 link orderings (described in Section 4.3 on the effectiveness of our algorithm. The networks contain four to 20 nodes and five to 30 links, with seven to 780 (*s*, *t*) paths. We use $CN_n^{|V|,|E|}$ to denote a CN with |V| nodes, |E| links, *n*(*s*, *t*) paths, and reliability constraint R_{min}. For each of the 20 networks in [6],[30],[32], we first randomly assigned the r_i and c_i for each e_i, and set R_{min} randomly between 40% to 80% of the reliability of the original network (*i.e.*, the reliability of the network with complete links). Then, we sort all links of each topology using the link order criteria, LO1-LO5 and use DPCR-L to compare the performance of the five link-orders and confirm the merit of using these order criteria. As shown in

Table 3, it is clear that the LO3 order is the best as compared to others because it gets 14 out of 20 best results. Please see the numbers in bold in columns LO1 to LO5 and the last row in the columns that shows the number of times each link order produces the best results among the five orders).; Notice that LO1, LO2 and LO5 are the next best orders, producing 9,8 and 7 best results, respectively, while LO4 is the worst performers with only six best results. It also shows that DPCR-L produces only three best results when using the random order; note that all other ordering criteria, *i.e.*, LO1-LO5 are also produce the three best results. The results confirm the merits of using the proposed link orders. Further, to generate NT with the minimum cost, The DPCR-L using each of the five link orders is run, and selects the NT with the lowest cost.

Table 3 shows that the LO3 order is the best and LO1, LO2 and LO5 are the next best orders, while LO4 is the worst performers, because LO3 order gets 14 out of 20 best results; see numbers in bold in columns LO1 to LO5. Notice that LO1, LO2 and LO5 producing eleven, eight and seven best results, respectively, Note that when DPCR-L using the random order , it produces only three best results; and all these three best results are also produced from LO1-LO5. Moreover, to produce network topology with the minimum cost, The DPCR-L using each of the five link orders is run, and selects the NT with the lowest cost.

Furthermore, Table 4 shows the best minimum cost for each network generated by DPCR-P [6] and the CPU time in second to generate the best topology; see the last two columns. Note that the authors [6] proposed Two link orders, CR1, CR2, and Table 4 shows the best outcomes of DPCR-P using all orders. As shown in Table 4, DPCR-L generates topology with lower cost for 11 of 20 networks, and speeds up its running time up to 31.71%, as compared to DPCR-P see column DPCR-L CPU time result with(%) and DPCR-P CPU time. Further, DPCR-L is generating the same results as DPCR-P for the other **7** topologies, while producing only two inferior results that has a lower cost for network $CN_9^{5,8}$ and $CN_{18}^{9,13}$. It is clear from the CPU's time of both approaches that DPCR-P potentially require generating all (*s*, *t*) simple paths to solve the problem, while our DPCR-L approach deletes sequentially only up to |E| links to solve the problem, so this will speed up the process and thus significantly reduces the time complexity.

5.2 The Performance of DPCR-L on Benchmark Networks.

We generated 100 networks from the 20 topologies in Section 5.1 to benchmark the optimality of DPCR-L. Let $\alpha>0$ be the number of edges deleted from a network. For $\alpha=1$, an optimal $G_i = (V, E-\{e_i\})$, for any $e_i \in E$, is generated as follows. For each possible $e_i \in E$, the G_i was first generated, where there are different |E| for each G_i. For each G_i, the Rel (G_i), is calculated, and the selected G_i with the maximum reliability as G_{min} with cost Cost(G_{min}). The steps are repeated for $\alpha=2, 3, 4, 5$. For larger α, the benchmark networks were not generated because it would be very time consuming. For example, when |E|=30 and $\alpha=6$, there are 593775 different G_i and thus finding G_{min} among them require a significant amount of time since computing the reliability value of each G_i takes exponential time. For each generated network G_i, we set R_{min}=Rel(G_{min}), the tightest possible constraint, and thus each benchmark evaluates the worst possible performance for our DP algorithms. The DPCR-L was run using LO3, to generate topologies from the 100 benchmark networks, and the better results between them are considered. Our simulations show that DPCR-L produces 93% optimal results. Interestingly, of the 7 non-optimal results, DPCR-L produces a network with reliability no worse than 5.38% off optimal, and most of the non-optimal results have a lower cost than that for optimal up to 0.17%.

Table 3: Best Cost(G) for each ordering criteria using DPCR-L

Input		DPCR-L Best Cost(G) for each ordering criteria					
	R_{min}	Random	LO1	LO2	LO3	LO4	LO5
$CN_7^{6,8}$	0.88	14	14	14	14	14	14
$CN_9^{5,8}$	0.85	27	26	22	26	25	24
$CN_{13}^{6,9}$	0.90	21	21	19	21	19	21
$CN_{13}^{9,12}$	0.90	25	25	25	25	25	25
$CN_{14}^{7,15}$	0.80	33	33	31	30	31	33
$CN_{18}^{11,21}$	0.95	21	20	21	18	21	19
$CN_{18}^{9,13}$	0.95	28	24	25	25	25	26
$CN_{24}^{8,12}$	0.90	42	30	38	40	38	31
$CN_{20}^{8,12}$	0.85	33	31	31	31	31	31
$CN_{25}^{7,12}$	0.90	28	26	26	24	26	24
$CN_{29}^{8,13}$	0.90	22	22	21	18	22	21
$CN_{36}^{16,30}$	0.90	31	29	30	30	31	29
$CN_{44}^{21,26}$	0.90	23	21	21	21	26	22

$CN_{44}^{9,14}$	0.75	29	25	27	25	27	29
$CN_{64}^{10,21}$	0.85	32	33	33	30	33	32
$CN_{136}^{17,25}$	0.90	28	30	30	27	30	27
$CN_{205}^{18,21}$	0.90	42	40	42	42	42	42
$CN_{281}^{13,22}$	0.90	34	29	27	30	29	30
$CN_{282}^{18,27}$	0.90	48	42	45	42	46	45
$CN_{780}^{20,30}$	0.90	40	37	36	31	31	35
Total		3	11	8	14	6	7

Table 4: The minimum cost of feasible topology for each network generated by DPCR-L and DPCR-P.

		DPCR-L			**DPCR-P**	
	R_{min}	Link Ordering	DPCR-L Cost(G) Best Result	CPU CPU time (seconds)	DPCR-P Cost(G) Best Result	CPU time (seconds)
$CN_{7}^{6,8}$	0.88	Random, LO1-LO5	14	0.01(0.0%)	14	0.01
$CN_{9}^{5,8}$	0.85	LO2	22	0.08(12.5%)	21	0.09
$CN_{13}^{6,9}$	0.90	LO2,LO4	19	0.63 (30.15%)	1 9	0.82
$CN_{13}^{9,12}$	0.90	Random, LO1-LO5	25	1.07(7.47%)	27	1.15
$CN_{14}^{7,15}$	0.80	LO3	30	2.74(9.85%)	31	3.01
$CN_{18}^{11,21}$	0.95	LO3	18	3.53(16.71%)	18	4.12
$CN_{18}^{9,13}$	0.95	LO1,LO3	24	4.40(2.50%)	22	4.51
$CN_{24}^{8,12}$	0.90	LO1	30	5.18(2.51%)	31	5.31
$CN_{20}^{8,12}$	0.85	LO1-LO5	31	7.19(7.37%)	31	7.72
$CN_{25}^{7,12}$	0.90	LO3,LO5	24	6.26(3.67%)	26	6.49
$CN_{29}^{8,13}$	0.90	LO3	18	6.44(4.96%)	20	6.76
$CN_{36}^{16,30}$	0.90	LO1,LO5	29	8.15(17.91%)	30	9.61
$CN_{44}^{21,26}$	0.90	LO1,LO2,LO3	21	8.84(13.57%)	21	10.04
$CN_{44}^{9,14}$	0.75	LO1,LO3	25	5.16(16.66%)	27	6.02
$CN_{64}^{10,21}$	0.85	LO3	30	5.37(21.78%)	33	6.54
$CN_{136}^{17,25}$	0.90	LO3,LO5	27	16.48(11.10%)	27	18.31
$CN_{205}^{18,21}$	0.90	LO1	40	18.94(25.97%)	42	23.86
$CN_{281}^{13,22}$	0.90	LO2	27	17.27(23.91%)	29	21.40
$CN_{282}^{18,27}$	0.90	LO1,LO3	42	19.54(31.78%)	44	25.75
$CN_{780}^{20,30}$	0.90	LO3,LO4	31	21.49(10.42%)	31	23.73

Table 5: Comparison between DPCR-L and optimal results

Input			DPCR-L	
CN	α	R_{min}	Rel(G_1)	Cost(G_1)
$CN_{18}^{9,13}$	3	83.6	83.8(0.23%)	20(-0.17%)
$CN_{136}^{17,25}$	4	87.1	87.5(0.45%)	23(-0.12%)
$CN_{44}^{9,14}$	1	72.8	75.9(4.25%)	26(-0.04%)
$CN_{64}^{10,21}$	3	89.1	90.3(1.34%)	37(-0.09%)
$CN_{136}^{17,25}$	2	83.5	88.0(5.38%)	32(-0.11%)
$CN_{282}^{18,27}$	5	75.6	75.9(0.39%)	27(-0.13%)
$CN_{281}^{13,22}$	3	92.8	92.8(0.00%)	36(-0.05%)

5.3 DPCR-L for large networks

To evaluate the time efficiency of the proposed DPCR-L, The DPCR-L is utilized to solve the NTD-CR for grid networks in [25] with the number of nodes, links and (*s*, *t*) paths range from 36 to 200, 57 to 298, and 538020 to 2^{99}, respectively. We set c_j=1 and r_j=0.9 for each of the five grid networks.
Note that it is the same result produced by DPCR-P in [6], but DPCR-L needs only 32.81 seconds to generate the result, DPCR-L speeds up the process up to (47.28%) compared to DPCR-P. These results show the applicability of DPCR-L on networks containing a large number of links. However, for these large networks, we are unable to gauge the optimality of its generated topologies. Note that DPCR-L needs only |E| links to produce its DP table, DPCR-P potentially needs O ($2^{|E|-|V|+2}$) (*s*, *t*) paths to produce its DP table. For large sized CN, there is significant difference, *e.g.*, $CN_{2^{99},149}^{200,298}$ that has 298 links with 2^{99} paths. Therefore, DPCR-L generates more NT that has lower cost, further; using the five link-orders also significantly outperforms DPCR-P in terms of computational time complexity.

6. CONCLUSION

In this research, an NP-hard network topology design problem, called NTD-CR, have been define to generate a network topology design with minimized cost subject to (s, t) terminal reliability. We have proposed a heuristic dynamic programming method, called DPCR-L, and five different link ordering criteria to to solve NTD-CR problem. Extensive Simulation based on different benchmark networks of various sizes are used to compare DPCR-L with existing state-of-the-art techniques and shows the merits of using the ordering methods, and the effectiveness of our algorithm. Comparing to the most recently proposed approach, DPA [6] simulations show that DPCR-L finds better results. Since DPCR-L is computationally more efficient as compared to the existing approaches, it becomes an obvious choice to be used in large network topology design. Furthermore, simulation results on large size networks show that DPCR-L speeds up the process with up to (47.28%) compared to the recent existing approach and show the practicality of our techniques. Finally, our simulations show that DPCR-L produces 93% optimal results.

REFERENCES

1. B. ELSHQEIRAT,S. Soh, S. RAI, and M. Lazarescu. **A Practical Algorithm for Reliable Network Topology Design**, *International Journal of Performability Engineering,* vol. 9, 2013.
2. B. K. Lad, M. S. Kulkarni, and K. B. Misra. **Optimal reliability design of a system**, in *Handbook of Performability Engineering*, ed:Springer, 2008,pp. 499-519.
3. A. Saad, S. A. Khan, and A. Mahmood. **A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design**, *Swarm and Evolutionary Computation,* vol. 38, pp. 187-201, 2018.
4. J.-M. Won and F. Karray. **Cumulative update of all-terminal reliability for faster feasibility decision,** *IEEE Transactions on reliability,* vol. 59, pp. 551-562, 2010.
5. H. M. AboElFotoh and L. S. Al-Sumait. **A neural approach to topological optimization of communication networks, with reliability constraints,** *IEEE Transactions on reliability,* vol. 50, pp. 397-408, 2001.
6. B. Elshqeirat, S. Soh, M. Lazarescu, and S. Rai. **Dynamic programming for minimal cost topology with two terminal reliability constraint**, in *2013 19th Asia-Pacific Conference on Communications (APCC)*, 2013, pp. 740-745.
7. B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu. **Topology design with minimal cost subject to network reliability constraint,** *IEEE Transactions on Reliability,* vol. 64, pp. 118-131, 2014.
8. R.-H. Jan, F.-J. Hwang, and S.-T. Chen. **Topological optimization of a communication network subject to a reliability constraint,** *IEEE Transactions on Reliability,* vol. 42, pp. 63-70, 1993.
9. T. Koide, S. Shinmori, and H. Ishii. **Topological optimization with a network reliability constraint,** *Discrete Applied Mathematics*, vol. 115, pp. 135-149, 2001.
10. A. Kumar, R. M. Pathak, Y. P. Gupta, and H. R. Parsaei. **A genetic algorithm for distributed system topology design,** *Computers & Industrial Engineering,* vol. 28, pp. 659-670, 1995.
11. D. L. Deeter and A. E. Smith. **Economic design of reliable networks,** *IIE transactions,* vol. 30, pp. 1161-1174, 1998.
12. B. Dengiz, F. Altiparmak, and A. E. Smith. **Efficient optimization of all-terminal reliable networks, using an evolutionary approach,** *IEEE transactions on Reliability*, vol. 46, pp. 18-26, 1997.
13. B. Dengiz, F. Altiparmak, and A. E. Smith. **Local search genetic algorithm for optimal design of reliable networks,** *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 179-188, 1997.
14. J. E. Ramirez-Marquez and C. M. Rocco. **All-terminal network reliability optimization via probabilistic solution discovery,** *Reliability Engineering & System Safety*, vol. 93, pp. 1689-1697, 2008.
15. L. Lin and M. Gen. **A self-controlled genetic algorithm for reliable communication network design,** in *2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 640-647.
16. A. Mutawa,H.Alazemi, and A. Rayes. **A novel steady-state genetic algorithm approach to the reliability optimization design problem of computer networks,** *International Journal of Network Management,*vol. 19,pp. 39-55, 2009.
17. F.-M. Shao, X. Shen, and P.-H. Ho. **Reliability optimization of distributed access networks with constrained total cost,** *IEEE transactions on reliability,* vol. 54, pp. 421-430, 2005.

18. M. M. Atiqullah and S. S. Rao. **Reliability optimization of communication networks using simulated annealing,** *Microelectronics reliability,* vol. 33, pp. 1303-1319,1993.
19. S. Pierre, M. A. Hyppolite, J. M. Bourjolly,and O. Dioume. **Topological design of computer communication networks using simulated annealing,** *Engineering Applications of Artificial Intelligence,*vol. 8,pp. 61-69, 1995.
20. F. Altiparmak and B. Dengiz. **A cross entropy approach to design of reliable networks,** *European Journal of Operational Research*, vol. 199, pp. 542-552, 2009.
21. C. Papagianni, K. Papadopoulos, C. Pampas,N. D. Tselikas,D.Kaklamani, and I. S. Venieris. **Communication network design using particle swarm optimization,** in *2008 international multiconference on computer science and information technology*, 2008, pp. 915-920.
22. G. Hardy, C. Lucet, and N. Limnios. **A BDD-based heuristic algorithm for design of reliable networks with minimal cost,** in *International Conference on Mobile Ad-Hoc and Sensor Networks*, 2006, pp. 244-255.
23. B. Dengiz,F. Altiparmak,and O.Belgin. **Design of reliable communication networks:A hybrid ant colony optimization algorithm,** *IIE Transactions,*vol. 42, pp. 273-287, 2010.
24. N. Kyriakou, E. Loukis, and S. Arvaniti. **Enterprise Systems and Innovation--An Empirical Investigation,** in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 4686-4696.
25. B. A. H. Elshqeirat. **Optimizing reliable network topology design using dynamic programming,** Doctoral dissertation, Doctoral dissertation, Curtin University, 2015.
26. B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu. **A dynamic programming algorithm for reliable network design,** *IEEE Transactions on reliability*, vol. 63, pp. 443-454, 2014.
27. S. Soh and S. Rai. **CAREL:Computer aided reliability evaluator for distributed computing networks,** *IEEE Transactions on Parallel and Distributed Systems,*vol. 2, pp. 199-213, 1991.
28. M.-S. Yeh. **A new Monte Carlo method for estimating network reliability,** in *Proceedings of the 16th International Conference on Computers & Industrial Engineering, 1994*, 1994.
29. T. H. Cormen, C. Leiserson, R. L. Rivest, and C. Stein. **Introduction to Algorithms Cambridge, Massachusetts,** ed: The MIT Press, 1990.
30. B. Elshqeirat, S. SOH, S. RAI, and M. Lazarescu. **Bi-Objective Topology Design of Communication Networks Using Dynamic Programming,** *International Journal of Performability Engineering,* vol. 11, pp. 265-274, 2015.
31. M. Zemzami, N. Elhami, M. Itmi, N. Hmina. **An evolutionary hybrid algorithm for complex optimization problems**, *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 8, No.2, pp. 126-133, 2019.
32. F. Yakoubi, N. ElKattabi, M. El Marraki. **The Reliability of the Recursive Corona Product Network,** *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 8, No.3, pp. 601-604, 2019.