School of Electrical Engineering and Computing
Department of Computing

Optimizing Reliable Network Topology Design Using Dynamic Programming

Basima Ahmad Haroun Elshqeirat

This thesis is presented for the Degree of

Doctor of Philosophy
of
Curtin University

February 2015

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university. To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

Basima Elshqeirat	Date

To my husband, Radi Elshqeirat

To my four daughters Abrar, Dan, Byan and Jana.

To my parents, Mr. Ahmad Elshqeirat and Mdm. Kafa Almajali

To my brothers and sister, Sami, Basam, Mohamad, Anas and Fatima

Abstract

Many applications require some network Quality of Service (QoS) constraints, *e.g.*, reliability and/or bandwidth to be operational. In particular, critical systems in emergency services must run without interruption, even in the presence of network component failures. Maximizing network bandwidth is equally crucial for many bandwidth-intensive applications, *e.g.*, video-on-demand. Therefore, it is important to design network topology (NT) that can meet its applications' QoS requirement, since in general, optimal NT directly affects network QoS. However, constructing NT with better QoS incurs higher installation cost, since for example link bandwidth and reliability are directly proportional to its installation cost. This thesis addresses several NP-hard optimization problems to design optimal NT that considers cost, reliability, and bandwidth.

The first part of the thesis focuses on a single objective problem, called NTD-R/C, to construct NT with the objective of maximizing reliability subject to a given cost constraint. The thesis considers either all-terminal reliability (Rel_{all}) or two-terminal reliability (Rel_2), and describes an efficient algorithm, called DPR/C, to solve the problem. DPR/C uses dynamic programming (DP) concept and requires as input a set of the network's spanning trees or (s, t) paths for Rel_{all} or Rel_2 objective, respectively. The first version of the approach, called DPR/C-1, uses all spanning trees or (s, t) paths of the network to produce optimal NT provided the input is optimally ordered. The thesis formally proves that generating optimal ordered input is NP-complete, and proposes five heuristic approaches to order the input using which DPR/C-1 produces near optimal results. Further, the ordered input also allows the second version of the solution, called DPR/C-2, to use only k spanning trees or (s, t) paths.

The second part of the thesis focuses on another single objective problem, called NTD-C/R, to construct NT with the objective of minimizing the network installation cost subject to the required reliability, Rel_{all} or Rel_2, level. It describes an efficient DP algorithm, called DPC/R, to solve the NTD-C/R problem. DPC/R also requires as input a set of the network's spanning trees or (*s*, *t*) paths. The first version of the approach, called DPC/R-1, uses all spanning trees or (*s*, *t*) paths of the network. The approach is shown to always obtain a feasible solution and produce optimal NT if its input is optimally ordered. The thesis formally proves that generating optimal ordered input is NP-complete, and proposes six heuristic approaches to generate effective order of the input. Moreover, an alternative algorithm, called DPC/R-2, that uses only *k* spanning trees or (*s*, *t*) paths is shown to produce almost optimal results.

The last part of the thesis considers two bi-objectives problems, *i.e.*, NTD-CR/B and NTD-CB/R. NTD-CR/B is to design NT with minimum cost and maximum Rel_2 subject to satisfying a required bandwidth, while NTD-CB/R aims to obtain NT with minimum cost and maximum bandwidth subject to satisfying a required operational Rel_2. It uses two Lagrange Relaxation formulations and two DP approaches, called DPCR/B and DPCB/R, to solve NTD-CR/B and NTD-CB/R, respectively. Finally, the thesis extends the two approaches to DPCR/B-P and DPCB/R-P approaches, using Pareto Optimal Set concepts.

Extensive simulations using hundreds of benchmark networks that contain up to 1.899^{102} spanning trees and 2^{99} paths show the effectiveness of all algorithms proposed in the thesis.

Acknowledgements

All praise to the God. We repose our trust in him, and look forward to him for assistance, whom his generosity and blessing was the stimulus for the success in my studies.

I would like to express my appreciation and thanks mainly to my supervisor Dr. Sieteng Soh for his guidance, advice and help over the past three years and six months, without which this thesis will not be possible. Also, I would like to thank Prof. Mihai Lazarescu for his support and help. In addition, I am also thankful for Prof. Suresh Rai from Louisiana State University, for sacrificing his personal time to review my papers.

My special gratitude for my husband, Radi Elshqeirat for loving and taking care of my four daughters, while I spent many hours till middle of many nights and many weekends to meet the conference and journal papers submission deadlines and writing this thesis. I am also grateful for my parents, my brothers and my sister whose patient and tolerance provided me with a never failing source of inspiration, which glittered the way for the successful completion of my thesis.

Thanks also to the University of Jordan for providing me with a scholarship to do this PhD.

Contents

List of Figures

List of Tables

This thesis is based upon several works that have been presented in conferences and published in journals over the course of the author's PhD.

Presented Work

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2012). A Dynamic Programming Approach for Network Design with Cost Constraint, *9th Postgraduate Elec. Eng. And Comp. Symp., PEECS2012*. The paper received the Commendation Award.

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2013). Dynamic Programming for Minimal Cost Topology with Two Terminal Reliability Constraint, *Proc. IEEE Asian Pacific Conference on Communication, 2013*, Indonesia.

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2013). A Practical Algorithm for Reliable Communication Network Design, *International Journal of Performability Engineering*, *9*(4), 397–408.

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2013). Dynamic Programming for Minimal Cost Topology with Reliability Constraint, *Proceedings of the 5th International Conference on Communication Software and Networks*, *Journal of Advances in Computer Networks*, *1*(4).

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2014). A Dynamic Programming Algorithm for Reliable Network Design, *IEEE Trans. Reliability*, *63*(2), 443–454.

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu (2015). Topology Design with Minimal Cost Subject to Network Reliability Constraint, *IEEE Trans. Reliability*. *In press*.

- Basima Elshqeirat, Sieteng Soh, Suresh Rai and Mihai Lazarescu. Bi-Objective Network Topology with Reliability Constraint. The 6th International Conference on Information and Communication Systems (ICICS 2015), Amman, Jordan, April 7-9, 2015 (Accepted, Feb 2015).

ACRONYMS

NT	Network Topology
NTD	Network Topology Design
CN	Communication Network
SO	Single Objective
MO	Multiple Objective
BO	Bi-Objective
NTD-R/C	Network Topology Design with maximum Reliability subject to a Cost constraint
NTD-C/R	Network Topology Design with minimum Cost subject to a Reliability constraint
NTD-CR/B	Network Topology Design with minimum Cost and maximum Reliability subject to a Bandwidth constraint
NTD-CB/R	Network Topology Design with minimum Cost and maximum Bandwidth subject to a Reliability constraint
DP	Dynamic Programming
DPA	Dynamic Programming Approach
DPR/C-1	The first version of DPA to solve NTD-R/C
DPR/C-2	The second version of DPA to solve NTD-R/C
DPC/R-1	The first version of DPA to solve NTD-C/R
DPC/R-2	The second version of DPA to solve NTD-C/R
DPCR/B-1	The first version of DPA to solve NTD-CR/B
DPCR/B-2	The second version of DPA to solve NTD-CR/B
DPCB/R-1	The first version of DPA to solve NTD-CB/R
DPCB/R-2	The second version of DPA to solve NTD-CB/R
POS	Pareto Optimal Set
DPCR/B-P	The DPA to solve NTD-CR/B using POS
DPCB/R-P	The DPA to solve NTD-CB/R using POS
OCx	Order Criterion *x*, for x=1, 2, …, 10
SDP	Sum of Disjoint Products
CAREL	Computer Aided Reliability Evaluator, an SDP technique
COM	COMpare operator of CAREL
RED	REDuce operator of CAREL
CMB	CoMBine operator of CAREL
GEN	GENerate operator of CAREL
MCS	Monte Carlo Simulation
SA	Simulated Annealing
GA	Genetic Algorithm
TS	Tabu Search
ACO	Ant Colony Optimization
B&B	Branch and Bound
NGA	Network Genetic Algorithm
LS-NGA	Local Search-Network Genetic Algorithm
NN	Neural Network
BDD	Binary Decision Diagram
ACO-SA	Ant Colony Optimization and Simulated Annealing

PCGA	Pareto Converging Genetic Algorithm
SP	Swarm Particle
MOIKP	Multiple Objective Integer Knapsack Problem

NOTATION

G=(V, E)	A graph/network with |V| nodes and |E| links
X_i or Y_j	A decision variable {0, 1}
v_i	Node i in graph G, where $v_i \in V$
e_j	Link j in graph G, where $e_j \in E$
c_j	Cost of e_j, $c_j > 0$
r_j	Reliability of e_j, $0 \leq r_j \leq 1.0$
b_j	Bandwidth of e_j, $b_j > 0$
w_j	Weight of each link $e_j \in E$, calculated using any OC*x*, *e.g.*, $w_j = c_j / r_j$ using OC1
C_{max}	Cost constraint of NTD-R/C problem
R_{min}	Reliability constraint of NTD-C/R or NTD-CB/R problems
B_{min}	Bandwidth constraint of NTD-CR/B problem
Rel_{all}	All-terminal reliability measure
Rel_2	Two-terminal reliability measure
ST_G	A set containing all spanning trees in G
n	Total number of spanning trees in G, $n = \|ST_G\|$
ST_i	A spanning tree i, for $i = 1, 2, \ldots, n$; $ST_i \in ST_G$
STX_i	A sequence of spanning trees $STX_i \subseteq (ST_1, ST_2, \ldots, ST_i)$
P_G	A set containing all (s, t) simple paths in G
m	Total number of (s, t) simple paths in G, $m = \|P_G\|$
P_i	A (s, t) simple path i, for $i = 1, 2, \ldots, m$; $P_i \in P_G$
PX_i	A sequence of (s, t) simple paths $PX_i \subseteq (P_1, P_2, \ldots, P_i)$
L_i	A set of links in $ST_i \in ST_G$ or $P_i \in P_G$
k	The total number of spanning trees or (s, t) simple paths in graph G that are used by DPA to produce its result; $k \leq n$ or $k \leq m$
$G_i = (V, E_i \subseteq E)$	A subgraph of G=(V, E) constructed from all links of spanning trees in STX_i or (s, t) simple paths in PX_i.
•	An entity that can be a graph G, a subgraph G_i, a spanning tree ST_i, a (s, t) simple path P_i, a sequence of spanning trees STX_i, a sequence of (s, t) simple paths PX_i, a union $X[i\text{-}1, c] \cup ST_i$, or a union $X[i\text{-}1, c] \cup P_i$
Link(•)	A function that returns all links in •
Cost(•)	A function that returns the total cost of all links in •
Rel_{all}(•)	A function that computes Rel_{all} of the network constructed from Link(•)
Rel_2(•)	A function that computes Rel_2 of the network constructed from Link(•)
BW(•)	A function that computes the bandwidth of the network constructed from Link(•)
$\Re_i$	The reliability contribution of including ST_i or P_i into a topology
DT	Disjoint Terms generated by CAREL [22]
$\|DT_i\|$	Total number of disjoint terms generated for ST_i
T_{max}	$\max\{\|DT_i\|\}$

G_{opt}	The topology that has maximum reliability among all possible topologies with $Cost(G_{opt}) \leq C_{max}$. Note that $Rel_{all}(G_{opt}) \leq Rel_{all}(G_n)$
G_{min}	The topology that has minimum cost among all possible topologies with $Rel_{all}(G_{min}) \geq R_{min}$. Note that $Cost(G_n) \geq Cost(G_{min})$
σ	A positive integer multiplier σ
round(■)	A function that returns the closest integer value of ■
$\check{R}_{min}$	$\check{R}_{min}=round(\sigma \times R_{min})$; we set $\sigma=100$, and thus $\check{R}_{min} \leq 100$
r	The reliability constraint for each column $\check{r}$, calculated as $r=\check{r}/\sigma$
DP[i, *]	V Element in row i=1, 2, …, n, and column *, of Dynamic Programming (DP) table that stores five pieces of information: C [i, *], R[i, *], B[i, *], W[i, *], X[i, *], L[i, *], and J[i, *], where column * can be c=0, 1, …, C_{max} in NTD-R/C, $\check{r}$=0, 1, …, $\check{R}_{min}$ in NTD-C/R and b=0, 1, …, B_{min} in NTD-CR/B and NTD-CB/R.
X[i, *]	A data structure that stores the spanning trees in STX_i or (s, t) simple paths in PX_i
L[i, *]	The set of links contained in X[i, *]
C[i, *]	The total cost of all links in L[i, *]
R[i, *]	The reliability of a selected STX_i or PX_i
B[i, b]	The bandwidth of a selected PX_i
W[i, b]	The weight of a selected PX_i with $BW(G_i) \geq B_{min}$
J[i, c]	An integer index that marks the starting column c of a range of columns in row i that have the same cost C[i, c]
J[i, $\check{r}$]	An integer index that marks the ending column $\check{r}$ of a range of columns that have the same reliability of r
J[i, b]	An integer index that marks the starting column c of a range of columns in row i that have the same weight W[i, b]

Chapter 1

Introduction

Tremendous developments and improvements in information and communication technologies (ICT) in recent years have rapidly increased users' dependency on various ICT applications, *e.g.*, Internet banking, online shopping, web surfing and emailing. Many of the applications require a set of operational Quality of Service (QoS) constraints, *e.g.*, reliability, bandwidth, and delay, and thus it is crucial for the applications' underlying communication network (CN) to satisfy their constraints. In other words, a well-designed CN is inseparable from the effective running of user applications. For various critical applications (*e.g.*, emergency system, rescue, and military operations), their designed CN must be as reliable as possible so that they can operate effectively and without interruption in the events of network component (*e.g.*, communication link) failures. On the other hand, for many bandwidth-intensive applications (*e.g.*, video conferencing, video-on-demand and large file transfers), their CN must provide sufficient bandwidth capacity; thus it is also essential to consider network bandwidth in designing CNs. However, constructing a reliable and high bandwidth CNs incurs higher installation cost, since link bandwidth and reliability are directly proportional to its installation cost [4].

Typically, a large scale CN has a multilevel hierarchical structure consisting of a backbone network and several local access networks (LANs), each of which grants

users accesses to its hosts and local servers. The backbone network is used to route data from each source node (*s*) to destination node (*t*) via switching nodes. Each backbone network is usually constructed as a mesh network topology (NT) that provides any-to-any connections among the nodes in the network. This thesis focuses on the backbone network design, and thus the CN referred in the thesis is analogous to a backbone network.

There are two major phases for designing an efficient and effective CN: Physical Design Phase (PDP) and Logical Design Phase (LDP). The PDP involves in selecting the most suitable NT (*e.g*., star, bus), and equipment (*e.g*., Ethernet, fiber, ISDN). On the other hand, the LDP mainly involves in selecting the communication protocols, message routing, flow control, *etc*.; this thesis considers only PDP. Each NT is a high-level outline of a CN. Designing the most suitable NT is the first step in the PDP; for an analogy, this step is comparable to an architectural drawing for a building.

The NT design includes finding the best layout of network components that minimizes cost while meeting the user's performance criteria, such as reliability, bandwidth, transmission delay and throughput. In practice, a network service provider has a budget limit to build its NT. Therefore, given a set of various centers (nodes), their possible connecting links, link failure rate, bandwidth capacity, and installation cost, an NT designer must carefully select the most suitable set of links to optimize one or more design objectives (*e.g*., reliability) while satisfying a given constraint (*e.g*., cost).

This thesis considers four related NT design optimization problems. The first problem, called Network Topology Design with maximum Reliability subject to a Cost constraint (NTD-R/C), emphasizes on constructing NT within a given cost budget constraint while maximizing its reliability; this problem is known NP-hard [5]. In

contrast, some applications must run on a topology with a guaranteed reliability for proper operation. For such applications, the NT design aims to minimize the network installation cost subject to the required reliability level; we call this second problem Network Topology Design with minimum Cost subject to Reliability constraint (NTD-C/R). In other words, NTD-C/R considers the topology with the minimum total link installation cost subject to the application's reliability constraint; this problem has also been shown NP-hard [6]. The third and fourth problems, addressed in the thesis, consider both network bandwidth and reliability and thus applicable for reliable bandwidth intensive applications. Specifically, the third problem, called Network Topology Design with minimum Cost and maximum Reliability subject to Bandwidth constraint (NTD-CR/B), aims to minimize cost and maximize reliability of the NT subject to satisfying a required operational bandwidth. In contrast, the fourth problem, called Network Topology Design with minimum Cost and maximum Bandwidth subject to Reliability constraint (NTD-CB/R) aims to minimize cost and maximize bandwidth of NT subject to satisfying a required operational reliability. Both NTD-CR/B and NTD-CB/R are also NP-hard problems [8] because computing network reliability in general is NP-hard [9].

This thesis proposes several heuristic solutions to solve each of the four problems. However, our heuristic-based approaches do not guarantee to produce optimal solutions. For the first two problems, *i.e.*, NTD-R/C and NTD-C/R, this thesis considers two CN reliability measures: all-terminal reliability (Rel_{all}) and two-terminal reliability (Rel_2); the other two problems, *i.e.*, NTD-CR/B and NTD-CB/R, are relevant only for Rel_2. Note that, as later discussed in Section 2.1, the all-terminal reliability of a network G, $Rel_{all}(G)$, is the probability that at least one spanning tree in

G is functional. On the other hand, the two-terminal reliability for a network G, $Rel_2(G)$, is the probability that at least one simple (*s*, *t*) path in G is functional. Note that a spanning tree is a subgraph of G that comprises of all the nodes in G, and a (*s*, *t*) simple path is a sequence of links from a node *s* to a node *t* such that no node is traversed more than once.

1.1 Aim and Approach

This thesis aims to address several optimization problems to design reliable NT that considers three performance metrics, *i.e.*, cost, reliability and bandwidth as either its objective or its constraint. We consider two problem categories: (i) problems that consider one objective and one constraint (O_1/C_1), and (ii) problems that consider two objectives and one constraint (O_2/C_1). Table 1.1 shows six possible O_1/C_1 problems that involve the three metrics. As shown in the table, this thesis aims to address only the first two problems for the following three reasons:

1) The focus of the thesis is to design reliable NTs, and thus the problem must include the reliability metric as either objective or constraint.
2) The DPR/C solution in Chapter 3 can be directly modified to solve the NTD-B/C problem, which maximizes NT Bandwidth subject to a Cost constraint. Similarly, the DPC/R solution in Chapter 4 can be used to solve the NTD-C/B problem by replacing the reliability calculation function, *i.e.*, $Rel_{all}(G)$ or $Rel_2(G)$, with a function that computes network bandwidth; NTD-C/B aims to minimize NT Cost subject to a Bandwidth constraint. Note that the NTD-B/C and NTD-C/B problems can be solved in pseudo-polynomial time, since the cost and bandwidth calculations have polynomial time complexity, in contrast to computing $Rel_{all}(G)$ or $Rel_2(G)$ – a known NP-hard problem [10].

3) The NTD-R/B and NTD-B/R to design NT with maximum Reliability subject to a Bandwidth constraint and NT with maximum Bandwidth subject to a Reliability constraint, respectively, do not make sense. Excluding cost metric, the original topology would be the optimal solution to the problems.

Table 1.1: Summary of the O_1/C_1 problems addressed in this thesis

	Objective O_1	Constraint C_1	Problem	Solution	Chapter
O_1/C_1	Reliability	Cost	NTD-R/C	DPR/C	Chapter 3
	Cost	Reliability	NTD-C/R	DPC/R	Chapter 4
	Bandwidth	Cost	NTD-B/C		
	Cost	Bandwidth	NTD-C/B		
	Reliability	Bandwidth	NTD-R/B		
	Bandwidth	Reliability	NTD-B/R		

Table 1.2: Summary of the O_2/C_1 problems addressed in this thesis

	Objective O_2		Constraint C_1	Problem	Solution	Chapter
O_2/C_1	Cost	Reliability	Bandwidth	NTD-CR/B	DPCR/B DPCR/B-P	Chapter 5
	Cost	Bandwidth	Reliability	NTD-CB/R	DPCB/R DPCB/R-P	Chapter 5
	Bandwidth	Reliability	Cost	NTD-BR/C		

Table 1.2 shows the three possible O_2/C_1 NT design problems that involve cost, reliability and bandwidth metrics. This thesis considers only the first two problems, *i.e.*, NTD-CR/B and NTD-CB/R. To the best of our knowledge, the problems have not been addressed in the literature. Note that our proposed solutions, DPCR/B or DPCB/R in Chapter 5, can be directly modified to solve the third problem NTD-BR/C that aim to maximize Bandwidth and Reliability subject to a given Cost constraint.

In summary, the aims of this thesis are as follows.

Aim 1: To propose an effective and efficient heuristic solution to solve NTD-R/C. This thesis considers both Rel_{all} and Rel_2 for reliability measures in the problem and uses a Dynamic Programming Approach (DPA) to solve the problem. Specifically, to maximize $Rel_{all}(G)$, DPA constructs a network topology G using a sequence of spanning trees such that the total cost of the selected trees satisfies a given cost constraint. Similarly, to maximize $Rel_2(G)$, it constructs G using a sequence of (s, t) paths such that the total cost of the selected paths satisfies the given cost constraint. Chapter 3 describes the DPA solution, called DPR/C, for NTD-R/C with Rel_{all} measure, while Appendix A shows how to use DPR/C for Rel_2 measure.

Aim 2: To propose an effective and efficient heuristic solution to solve NTD-C/R. This thesis considers both Rel_{all} and Rel_2 for reliability measures of the problem. Our proposed DPA solution constructs the NT using a sequence of spanning trees for Rel_{all} constraint, or (s, t) paths in the network for Rel_2 such that the reliability of the selected trees or paths satisfies the given reliability constraint while minimizing its cost. Chapter 4 describes the proposed DPA solution, called DPC/R, to solve the problem for Rel_{all} metric, while Appendix B describes how to use the solution for Rel_2 constraint.

Aim 3: To propose two effective and efficient heuristic solutions, called DPCR/B and DPCB/R, to solve NTD-CR/B and NTD-CB/R respectively. This thesis considers Rel_2 for reliability measure in the problems. In both solutions, the thesis describes how to use the Lagrange Relaxation (LR) technique to convert the O_2/C_1 problems into their corresponding O_1/C_1 problems. DPCR/B is a DPA technique that constructs NT using a selected (s, t) paths that minimize cost and maximize $Rel_2(G)$ such that the NT

satisfies a given bandwidth constraint. Similarly, DPCB/R constructs an NT using (*s*, *t*) paths to minimize cost and maximize bandwidth such that the NT satisfies the given Rel_2 constraint. Chapter 5 first describes DPCR/B and DPCB/R that produce only one optimum solution for each problem. Then, the chapter extends DPCR/B and DPCB/R to DPCR/B-P and DPCB/R-P, respectively, to produce a set of trade-off optimal solutions (known as the Pareto Optimal Set (POS) [11]).

1.2 Significance and Contributions

The main contributions and significance of the thesis are threefold.

1) Chapter 3 of the thesis addresses NTD-R/C problem and proposes a novel DPA, called DPR/C, to solve the problem. This solution is important for sensitive/critical applications such as those used in rescue and military operations. The chapter describes two versions of the algorithm: DPR/C-1 and DPR/C-2. For Rel_{all} metric, DPR/C-1 generates the solution from all spanning trees of the network. The chapter proves that DPR/C produces optimal networks using an optimal sequence of spanning trees. It also describes five ordering criteria, OC1, OC2, OC3, OC4 and OC5, to heuristically generate the best sequence of spanning trees that allow DPR/C-1 to produce near optimal results. The chapter describes how DPR/C-2 utilizes the OCs to use only $k \leq n$ spanning trees; where n is the total number of all spanning trees in the network. For Rel_{all}, DPR/C-2 can be used on grid networks that contain up to 1.899^{102} spanning trees, and requires no more than 26.32 CPU seconds to produce near optimal results. Appendix A shows how to use DPR/C for Rel_2, which, on grid networks that contain up to 2^{99} (*s*, *t*) simple paths, produces near optimal results in time no more than 22.64 CPU seconds.

2) Chapter 4 of the thesis addresses NTD-C/R problem, and presents another DPA, called DPC/R, to solve the problem. Further, it describes two versions of the algorithm: DPC/R-1 and DPC/R-2. For Rel_{all} measure, the chapter provides a formal proof to show that DPC/R-1 will produce an optimal solution using an optimal sequence of spanning trees. It shows that generating the optimal sequence is NP-complete, and thus the chapter uses five ordering criteria, *i.e*., OC1, OC2, OC3, OC4 and OC5. Utilizing the OCs, DPC/R-2 uses only $k \leq n$ spanning trees, which significantly reduce the time complexity of DPC/R-1 while producing almost the same results. Specifically, for a typical 2×100 grid network that contains up to 1.899^{102} spanning trees, DPC/R-2 requires only k=1214 spanning trees to generate a topology with a reliability within 5.05% off from optimal and requires only 27.11 CPU seconds. Further, simulations on 76 fully connected networks that contain up to 2.3×10^9 spanning trees show the effectiveness of the algorithm vis-à-vis to four meta-heuristic state-of-the-art techniques [12]-[15], in terms of optimality as well as time complexity. Appendix B shows how to use DPC/R for Rel_2, which is able to generate 91% best solutions and uses only between 8.89% and 27.5% of each network's total paths to generate the results.

3) Chapter 5 presents two O_2/C_1 problems, called NTD-CR/B and NTD-CB/R. These two problems, to the best of our knowledge, have not been addressed in the literature. The solutions of these problems are crucial for bandwidth-intensive critical applications such as telesurgery (Remote surgery), video-conferencing, software updates and large file transfers where large amounts of bandwidth are required [7]. It proposes a novel DPA solution, called DPCR/B, to solve NTD-CR/B, and another algorithm, called DPCB/R, to solve NTD-CB/R. For both

algorithms, it introduces a metric that quantifies a trade-off between reliability and cost for NTD-CR/B, and bandwidth and cost for NTD-CB/R. Then the chapter proposes two Lagrange Relaxation formulations, which integrate each problem's constraint and two objectives using a weighted sum method to convert the problem into its corresponding O_1/C_1 problem. Further, it describes two versions of DPCB/R, *i.e.*, DPCR/B-1 and DPCR/B-2, and two versions of DPCB/R, *i.e.*, DPCB/R-1 and DPCB/R-2. Both DPCR/B-1 and DPCB/R-1 require as input all (s, t) simple paths of the network, while DPCR/B-2 and DPCB/R-2 utilize three heuristic ordering criteria, *i.e.*, OC8, OC9, and OC10, so that they can use only $k \leq m$ paths to generate results more efficiently; m is the total number of all (s, t) simple paths of the network. DPCR/B-2 and DPCB/R-2, respectively, reduce the time complexity of DPCR/B-1 and DPCB/R-1 while producing almost equal results. Extensive simulations on large networks with various sizes show the efficiency and effectiveness of both versions of DPCR/B and DPCB/R. Finally, the chapter extends DPCR/B and DPCB/R to DPCR/B-P and DPCB/R-P, respectively, such that they generate the POS of non-dominated solutions to solve NTD-CR/B and NTD-CB/R problems. The extensions improve the results generated using NTD-CR/B and NTD-CB/R problems by 2.17% and 3.37%, respectively.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 describes background, CN model and notations, and methods to calculate network reliability and bandwidth. Further, it explains the concept of dynamic programming (DP) and POS. A review of related works on single objective and multiple objectives problems is also presented. Finally, the chapter describes the simulation environment used to evaluate the performances of all proposed algorithms in this thesis.

Chapter 3 formulates the NTD-R/C problem and describes our proposed DPR/C solution for the problem with an illustrating example. The chapter also describes spanning tree preprocessing, using five order criteria, to improve the performance of DPR/C. The chapter provides theoretical analysis of DPR/C, and presents our simulation methodology and results.

Chapter 4 formally defines NTD-C/R, and presents our proposed solution, DPC/R, to solve the problem using a sequence of spanning trees of the network. The chapter also provides theoretical analysis of the solution. Further, it provides performance comparisons against four existing state-of-the-art techniques [12]-[15].

Chapter 5 discusses two related O_2/C_1 problems, *i.e.*, NTD-CR/B and NTD-CB/R, and proposes two metrics to measure their feasible solutions. After that, it uses Lagrange Relaxation techniques and weighted sum to transform each of the O_2/C_1 problems into its corresponding O_1/C_1 problem. It then describes two solutions, called DPCR/B and DPCB/R, with example and analysis for NTD-CR/B and NTD-CB/R respectively. After extending the two approaches to DPCR/B-P and DPCB/R-P approaches to produce POS solutions, the chapter presents simulation results to show the effectiveness and efficiencies of all approaches.

Chapter 6 presents the conclusion of the thesis and some possible future works.

This thesis includes three Appendices. Appendix A contains details on how to use DPR/C in Chapter 3 to solve NTD-R/C for Rel_2 metric. Appendix B is an extension of Chapter 4; it proposes two versions of DPC/R to solve NTD-C/R problem for Rel_2 metric. Appendix C provides the network topologies and their link information, *i.e.*, cost, reliability and bandwidth, which are used in our simulations.

Chapter 2

Background

This chapter, divided into five main sections, provides the preliminaries and reviews of the literature that is pertinent to this thesis. Section 2.1 describes network model and notations that are used throughout the thesis. Note that additional notations that are used only in specific chapters will be described in their corresponding chapters. Section 2.2 presents the QoS calculation methods used in the thesis, *e.g.*, exact and approximation network reliability calculation and bandwidth calculation. Section 2.3 discusses dynamic programming technique and Section 2.4 explains the concept of Pareto Optimal Set that contains non-dominated solutions to optimize bi-objectives one-constraint (O_2/C_1) problems. Section 2.5 reviews the existing algorithms for single-objective one-constraint (O_1/C_1) network topology design (NTD) problems, *i.e.*, NTD-R/C and NTD-C/R. The section then presents some related works for O_2/C_1 problems, *i.e.*, NTD-CR/B and NTD-CB/R. Section 2.6 describes the practical implementation issues in NTD. Section 2.7 summarizes this chapter.

2.1 Network Model

A communication network (CN) can be modelled by a probabilistic undirected simple graph G=(V, E), in which each node $v_i \in V$ represents a network component (*e.g.*, router, computer centre) and each link $e_j \in E$ represents the connecting media (*e.g.*, communication link) between the network components. All nodes' location and connecting links are known when the centres are established. Fig. 2.1 shows a CN with four fixed-position nodes, and five links. Each e_j has a cost $c_j>0$ that represents the cost

to install e_j, and reliability $0 \leq r_j \leq 1$ that represents the probability that e_j is functioning (a state referred to as UP) and $b_j > 0$ that represents the bandwidth capacity of e_j. Table 2.1 shows the link weight for the CN in Fig. 2.1. We assume that (i) all nodes are always UP and use no setup costs, and (ii) link failures are statistically independent and without repair. Both assumptions (i) and (ii) are commonly used in research works on NTD-R/C [16]-[19] as well as NTD-C/R [3], [12]-[15], and O_2/C_1 problems [8]. Assumption (ii) is used to reduce the combinatorial size of the already difficult problems; NTD is an NP-hard combinatorial optimization problem where the search space for a fully connected network with V nodes and L links is $L^{(|v|\times(|v|-1)/2)}$ [6]. Further, the assumption does not limits the practicality of the problem since a node may represent a very expensive large data centre that includes sufficient fault-tolerant and backup components, which allow the centre to continue its operation when there is any component failure. In this case, each node would be more reliable than its communication links, and its setup cost is very expensive and has been implicitly included when the node is established.

A spanning tree *i*, ST_i for $i=1, 2, \ldots, n$, is a subgraph of G which is a tree that comprises of all the nodes in G. Each spanning tree in a network with |V| nodes contains (|V|-1) links. Let ST_G be a set of all spanning trees in G, $n=|ST_G|$, and L_i be the set of links in $ST_i \in ST_G$. Note that a fully connected network contains $n=O(|V|^{|V|})$ spanning trees [20]. Let $Cost(ST_i)$ denote the cost of installing all links in spanning tree ST_i, calculated by taking the sum of the c_j of each e_j in ST_i; as an example, $Cost(ST_1)=3+4+6=13$. $Link(ST_i)$ denotes a function that returns links in ST_i. Let $Rel_{all}(ST_i)$ denote a function that computes the reliability of spanning tree ST_i, calculated by multiplying all r_j of each e_j in ST_i, *e.g.*, $Rel_{all}(ST_1)=0.6\times0.9\times0.9=0.486$. Table 2.1 provides the spanning

trees in ST_G of the CN in Fig. 2.1, and their reliability and cost. The cost of a network topology G, Cost(G), is calculated by taking the sum of all c_j for each $e_j \in G$; for Fig. 2.1, Cost(G)=20. Since G can be constructed using nodes in V and all links in E or all spanning trees in ST_G, the thesis uses $Cost(G)=Cost(ST_G)=Cost(E)$.

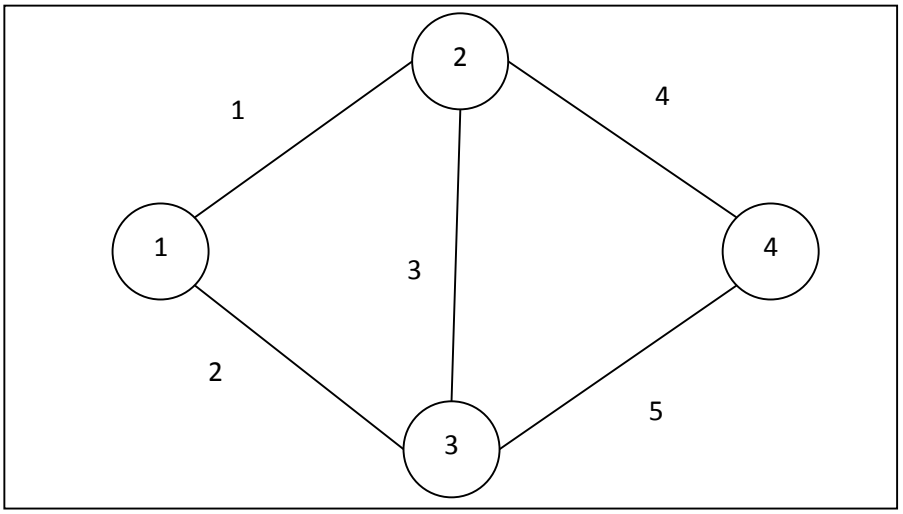


Fig. 2.1: An example CN with four nodes and five links

Table 2.1: Link weight and spanning tree set for CN in Fig. 2.1

ST_G				Link Weight		
i	ST_i	$Rel_{all}(ST_i)$	$Cost(ST_i)$	e_j	c_j	r_j
ST_1	{2, 4, 5}	0.486	13	1	5	0.9
ST_2	{1, 4, 5}	0.729	15	2	3	0.6
ST_3	{1, 2, 4}	0.486	12	3	2	0.7
ST_4	{2, 3, 5}	0.378	11	4	4	0.9
ST_5	{1, 2, 5}	0.486	14	5	6	0.9
ST_6	{2, 3, 4}	0.378	9			
ST_7	{1, 3, 4}	0.567	11			
ST_8	{1, 3, 5}	0.567	13			

A (*s*, *t*) simple path *i*, P_i, is a sequence of links from a source node *s* to a destination node *t* such that no node is traversed more than once. Note that a general network (fully connected network) contains $m=2^{|E|-|V|+2}$ (*s*, *t*) simple paths [21]. Let L_i be the set of links in P_i, P_G be a set of all paths in G and $m=|P_G|$. Table 2.2 provides the (*s*, *t*) simple paths in P_G of the CN in Fig. 2.2. Let $Cost(P_i)$ denotes the cost of installing all links in path P_i, calculated by taking the sum of c_j of each $e_j \in P_i$. As an example,

Cost(P_1)=3+6+3=12; see Table 2.2. The cost of a network topology G, Cost(G) is calculated by taking the sum of all c_j for each $e_j \in G$; for Fig. 2.2, Cost(G)=29. Let $Rel_2(P_i)$ denotes the reliability of path P_i, calculated by multiplying all r_j of each $e_j \in P_i$; *e.g.*, $Rel_2(P_1)=0.6\times0.9\times0.9=0.486$. Let $BW(P_i)$ denotes the bandwidth of path P_i, determined by the bottleneck link in the path; *e.g.*, $BW(P_1)=Min(3,4,8)=3$. Since G can be constructed using nodes in V and all links in E or all paths in P_G, the thesis uses $Cost(G)=Cost(P_G)=Cost(E)$.

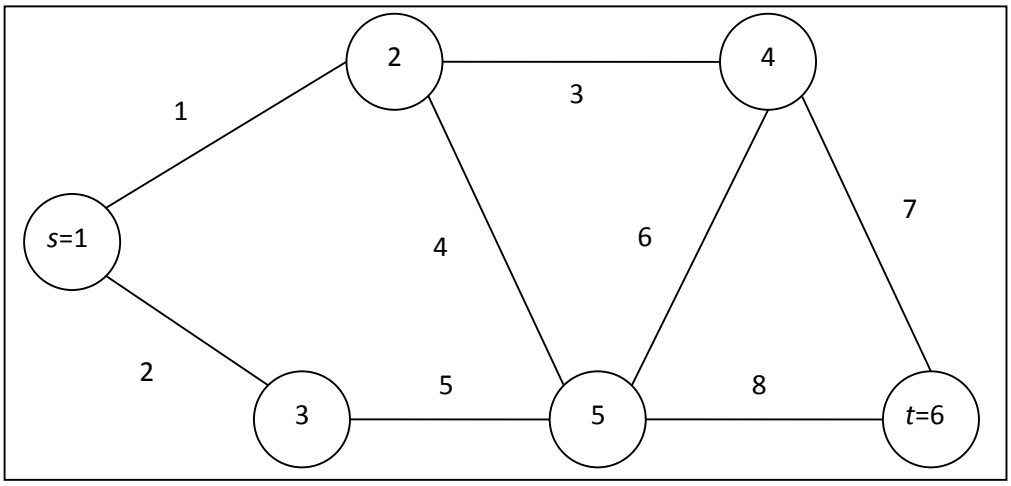


Fig. 2.2: An example CN with six nodes and eight links

Table 2.2: Link weight and path set for CN in Fig. 2.2

P_G					Link Weight			
i	P_i	$Rel_2(P_i)$	$Cost(P_i)$	$BW(P_i)$	e_j	c_j	r_j	b_j
P_1	(2,5,8)	0.486	12	3	1	5	0.9	8
P_2	(2,5,6,7)	0.226	15	3	2	3	0.6	3
P_3	(2,5,4,3,7)	0.238	17	3	3	2	0.7	6
P_4	(1,3,6,8)	0.340	14	4	4	4	0.9	5
P_5	(1,3,7)	0.441	9	6	5	6	0.9	4
P_6	(1,4,8)	0.729	12	5	6	4	0.6	4
P_7	(1,4,6,7)	0.340	15	4	7	2	0.7	6
					8	3	0.9	8

2.2 QoS Computations

Section 2.2 presents the QoS methods used in the thesis; mainly the exact reliability calculation using CAREL [22] method and approximation reliability calculation using Monte Carlo Simulation [10], followed by the bandwidth calculation using Max-Flow Min-Cut algorithm.

2.2.1 Reliability Calculation

We use network reliability to refer to the ability of a network G to continue working, when some of its nodes or links fail. As stated in Section 2.1, this thesis considers only networks with link failure; one can use the approach in [23] to calculate the reliability, when both nodes and links can fail, with an extra computational complexity.

The literature [24] considers three network reliability measures: (i) *K-terminal reliability* (Rel_K) – the probability that there exists at least one operating path between every pair of nodes in *K*, for a given set of *K* nodes in G; (ii) *Two-terminal reliability* (Rel_2) – the probability that a set of operational links provides communication path between a source node *s* and a terminal node *t* in G; and (iii) *All-terminal reliability* (Rel_{all}) – the probability that a set of operational links provides a communication path between every pair of nodes in G. This means that there are some operational links which can provide communication path among the $|K|=|V|$ nodes that can compensate for the failed links.

In this thesis, we only consider Rel_{all} and Rel_2. Let $Rel_{all}(G)$ and $Rel_2(G)$ be functions to compute Rel_{all} and Rel_2 measures of network G, respectively. Calculating $Rel_{all}(G)$, in general, is an NP-hard problem [6]. Similarly, calculating $Rel_2(G)$, in general, is also NP-hard problem [25]. $Rel_{all}(G)$ or $Rel_2(G)$ can be computed using an exact

method, *e.g.*, [22], using heuristic technique, *e.g.*, [10], or using approximation (bounding) method, *e.g.*, [25]. This thesis uses CAREL [22] – a sum of disjoint products (SDP) technique, to compute the exact value of $Rel_{all}(G)$ and $Rel_2(G)$ for NTD-R/C problem and NTD-CR/B in Chapter 3 and 5, respectively; and the approximation method, *i.e.*, Monte Carlo (MC) simulation [10] for the other problems in Chapter 4 and 5. The thesis uses exact reliability calculation in Chapter 3 and 5 for two reasons: 1) computing an exact Rel_{all} or Rel_2 measure avoids our algorithm from going into a local maximum state, and 2) using CAREL [22] for our DP approaches is efficient because it computes only the reliability contribution of each additional spanning tree or (*s*, *t*) simple path; see Section 3.3.1 for more detailed discussion. Since G can be constructed using nodes in V and all links in E or all spanning trees in ST_G, the thesis uses $Rel_{all}(G)=Rel_{all}(ST_G)=Rel_{all}(E)$; similarly, $Rel_2(G)=Rel_2(P_G)=Rel_2(E)$. In the following, we briefly describe CAREL [22] and MC simulation [10], called MCS.

CAREL:

CAREL is an SDP technique that utilizes four operators: COM, RED, CMB, and GEN [22]. For $Rel_{all}(G)$, given a sequence of all spanning trees of G, (ST_1, ST_2, …, ST_n) for i=1, 2, 3, …, n, CAREL generates an equivalent set of disjoint terms DT_i for each ST_i, which in turn is used to calculate the reliability contribution of including ST_i, called $\Re_i$, as follows. Firstly, it uses COM to generate a set of conditional cubes {E_1, E_2, …, E_{i-1}} when each spanning tree in (ST_1, ST_2, …, ST_{i-1}) is compared with ST_i. Secondly, it uses the RED operator to remove any redundant conditional cubes from {E_1, E_2, …, E_{i-1}}; a cube E_a is redundant with respect to E_b if $E_b \subseteq E_a$. As described in [22], removing a redundant cube can speed up the processing of the CMB operator. Thirdly,

it uses the CMB operator to generate a set of disjoint terms DT_i from the non-redundant cubes. Finally, GEN is used to calculate the reliability value of each DT_i, *i.e.*, $\Re_i$, which represents the reliability contribution of including ST_i in the network. Because a term in each set DT_i is mutually disjoint with respect to each other term within the same set as well as with respect to every term in other disjoint terms generated for all other spanning trees, GEN computes the total reliability contribution of all DT_i, *i.e.*, $\sum_{i=1}^{n} \Re_i$, to obtain the network's reliability; note that $\Re_1 = Rel_{all}(ST_1)$. Similarly, CAREL uses the four operators, *i.e.*, COM, RED, CMB, and GEN, to compute $Rel_2(G)$ from a sequence of all (s, t) simple paths of G, (P_1, P_2, …, P_m) for i=1, 2, 3, …, m. Readers are referred to [22] for detailed descriptions of CAREL.

MCS:

MCS contains two sampling loops to estimate either $Rel_{all}(G)$ or $Rel_2(G)$ of network G. In the following discussion, let us consider $Rel_{all}(G)$; $Rel_2(G)$ can be computed similarly. In the first loop, failed links are simulated to generate the probability distribution of the number of failed links (*i.e.*, $f[l=d]$; d=0, . . ., |E|). In the second loop, the network connectivity calculation is called nr times to simulate β_d values depending on whether or not the topology instance is connected. Here, reliability estimator values, $R(G)$, are computed based on $f[l=d]$ and β_d for each i=1, . . ., nr. The final reliability, $Rel_{all}(G)$, is estimated as the mean of the $R(G)$ values. The steps of the MCS [10] are as follow:

Step 1. Obtain the distribution of the number of failed links, $f[l=d]$, for all d=0, 1, 2, . . , |E|:

1.1. l_d=0 for d=0, 1, 2, . . . , |E|.

1.2. Perform the following steps *nr* times.

1.2.1. Perform the following steps for each *i*, *i*=1, 2, . . . , |E|.

1.2.1.1. Generate random number u_i from $U(0,1)$.

1.2.1.2. If $u_i < f$ then $x_i=1$ else $x_i=0$.

1.2.2. If the number of failed links is equal to *d* then $l_d=l_d+1$.

1.3. $f[l=d]=l_d/nr$ for *d*=0, 1, 2, . . . , |E|.

Step 2. Perform following steps *nr* times:

2.1. Perform the following steps for each *d*, *d*=0, . . ., |E|.

2.1.1. Arbitrarily choose *d* links from the network.

2.2.2. If the network is connected after removing the chosen *d* links, then $\beta_d=1$, else $\beta_d=0$.

2.2. $R_j(G)=\sum_{d=0}^{|E|} \beta_d \times f[l=d]$.

Step 3. Estimate all-terminal reliability as $Rel_{all}(G)=\sum_{j=1}^{nr} \frac{R_j(G)}{nr}$.

2.2.2 Bandwidth Calculation

The bandwidth of network topology G, BW(G), between a source node *s* and destination node *t*, can be calculated using the well-known Max-Flow Min-Cut theorem [26], where *the value of the maximum flow is* equal *to the capacity of the minimum cut*. Note that a cut between (*s*, *t*) node pair is a set of links whose removal disconnects the two nodes. The bandwidth of a cut set, $BW(Cut_i)$, for a minimal cut Cut_i is the sum of bandwidth b_j of all links $e_j \in Cut_i$. From Max-Flow Min-Cut theorem [26], the maximum capacity flow, W_{max}, through the graph G=(V, E) is:

$$BW(G)=W_{max}=\text{Min}\{BW(Cut_i)\} \quad (2.1)$$

As an example, consider the CN shown in Fig. 2.2. The cut set between source node *s* and destination node *t* is $Cut_{s,t}$={Cut_1=(1,2), Cut_2=(1,5), Cut_3=(3,4,5), Cut_4=(2,3,4), Cut_5=(3,4,8), Cut_6=(3,6,8), Cut_7=(1,4,6,8), Cut_8=(2,4,6,7), Cut_9=(4,5,6,7), Cut_{10}=(7,8)}. The bandwidth of each cut in $Cut_{s,t}$ is: BW(Cut_1)=11, BW(Cut_2)=12, BW(Cut_3)=15, BW(Cut_4)=14, BW(Cut_5)=19, BW(Cut_6)=18, BW(Cut_7)=25, BW(Cut_8)=20, BW(Cut_9)=19 and BW(Cut_{10})=14. Using Eq. (2.l), for Fig. 2.2, BW(G)=Min(11,12,15,14,19,18,25,20,19,l4)=11. Since G can be constructed using nodes in V and all links in E or all (*s*, *t*) simple paths P_G, the thesis uses BW(G)=BW(P_G)=BW(E).

2.3 Dynamic Programming Technique

The term *dynamic programming* (DP) was first used in the 1950s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another [27]. In general, DP is an optimization approach that breaks a complex problem into some simpler sub problems whose solutions are used to solve the problem. In contrast to the Divide and Conquer approach, the sub problems in DP are overlapping.

At every stage, DP makes decisions based on all the decisions made in the previous stage, and may reconsider the previous stage's algorithmic path to the solution. DP solves each sub problem only once and thus it reduces the number of computations. This is especially useful when the number of repeating sub problems is exponentially large.

The DP method has been used to solve several reliable design problems. For example, Alice *et al*. [27] used DP to solve reliability and redundancy allocation for system

design. Further, Yutaka *et al*. [28] showed how to solve the topology optimization for wireless sensor network using a DP technique. To the best of our knowledge, DP technique has not been used to solve the NT design problems in Table 1.1 and 1.2. In this thesis, we propose using DP to solve the four NT design optimization problems.

2.4 Pareto Optimal Set

Single-objective single constraint (O_1/C_1) optimization NTD problems, *e.g.*, NTD-R/C and NTD-C/R in Chapter 3 and 4, respectively, aim to find one solution that optimizes only one objective function subject to a given constraint. Extending the idea to bi-objective single constraint (O_2/C_1) optimization problems, *e.g.*, NTD-CR/B and NTD-CB/R in Chapter 5, one may find that the two objectives cannot be optimized at the same time. Specifically, one needs to quantify a trade-off between the two objectives, *e.g.*, minimizing cost and maximizing reliability in NTD-CR/B. The set of the best solutions that trade-off between the two objectives is called the Pareto Optimal Set (POS) [11].

Most multi-objective optimization algorithms with M objective functions use the concept of domination [11] to find the POS; note that O_2/C_1 is a multi-objective with M=2 objective functions. In these algorithms, two solutions are compared on the basis of whether one dominates the other or not. Formally, we define the domination concept as follows.

Definition 2.1. A solution **x** is said to dominate the other solution **y** if the following two conditions are both true: (i) The solution **x** is no worse than **y** in all objectives M; (ii) The solution **x** is strictly better than **y** in at least one objective.

If either of these two conditions is violated, solution **x** doesn’t dominate solution **y**.

To illustrate the concept, let us consider the bi-objective NTD-CR/B problem, described in Chapter 5, and its five feasible solutions, shown in the objective space in Fig. 2.3. Notice that each feasible solution satisfies the constraint B_{min}, and the first objective function (*i.e.*, cost) needs to be minimized while the second objective function (*i.e.*, reliability) needs to be maximized. Further, it is unlikely that there exists any solution that produces minimum cost and maximum reliability at the same time. Since both objective functions are equally important, we can use Definition 2.1 to decide which solution is better between any pair of solutions. For example, comparing solution 1 (cost=15, reliability=0.6) and solution 2 (cost=20, reliability=0.5) one can observe that the former is better than the latter in both objective functions, satisfying both conditions of Definition 2.1. For another example, consider solutions 1 (15, 0.6) and 5 (15, 0.9). Here, solution 5 is better than solution 1 in the reliability objective function and the solution is no worse than solution 1 in the cost objective (in fact, they are equal). Thus, both conditions for domination are satisfied and solution 5 dominates solution 1. Since the concept of domination allows a way to compare solutions with multiple objectives, most multi-objective optimization methods use the concept to search for POS that contain non-dominated solutions [11]. Section 5.6 explains how to extend our proposed algorithms to produce POS with non-dominated solutions for NTD-CR/B and NTD-CB/R problems.

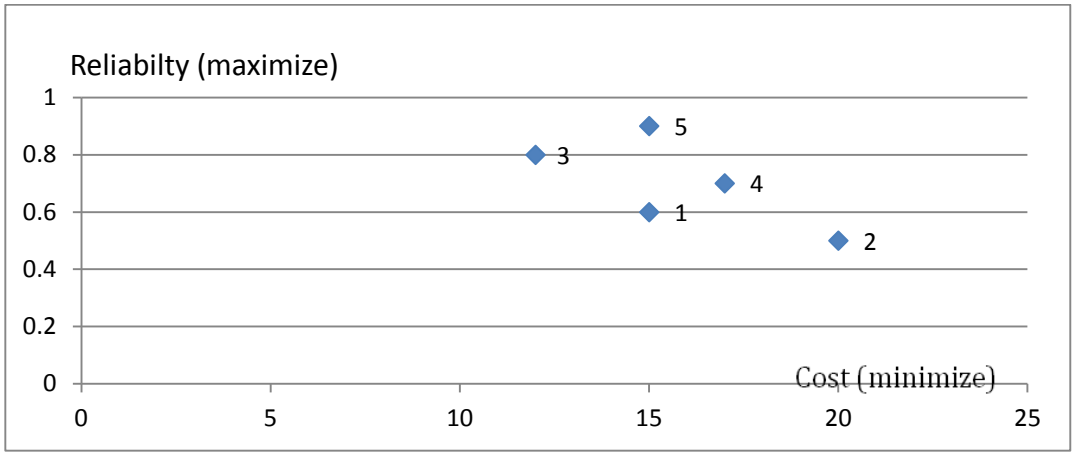


Fig. 2.3: Example of five feasible solutions for O_2/O_1 problem

2.5 Related Work

This section first discusses two related O_1/C_1 NTD problems, *i.e.*, NTD-R/C and NTD-C/R, each of which aims to optimize one objective while satisfying one constraint parameter. Then, it describes the existing works on O_2/C_1 NTD that aims to optimize two objectives subject to one constraint parameter.

2.5.1 O_1/C_1 NTD Problems

Both NTD-R/C and NTD-C/R problems have been shown NP-hard [5], [6]. Their solutions in the literature can be categorised into either enumerative methods or heuristic methods. Enumerative–based approaches in [16], [17], [28], [29] are applicable only for small network sizes. For larger networks**,** heuristic-based approaches are used; however, they do not guarantee optimality. While there are many solutions for the NTD-C/R problem, *e.g.*, ([3], [6], [12]-[15], [28]-[36]), we found only a few papers, *i.e.*, [16]-[19], [37], that address the equally important NTD-R/C problem.

Aggarwal *et al.* [16] employed a greedy enumerative technique to solve the NTD-R/C problem that maximizes Rel_2. The authors in [17] showed that the solution can also be

used to maximize Rel_{all}. Zakir *et al.* [19] extended the NTD-R/C problem in [17] to produce the topology with as many disjoint spanning trees as possible for maximizing Rel_{all}. Consequently, their solution [19] produces topology that has a higher reliability but with a higher cost. However, the solutions [16], [17], [19] require generating all spanning trees or paths of the network, and thus are not feasible for designing networks with a large number of spanning trees or paths. Atiqullah and Rao [37] have proposed a two-stage heuristic approach based on simulated annealing (SA) to solve the design problem for multilevel hierarchical network that consists of backbone network layer and local access networks layer. One stage of the heuristic solves the design for the backbone network while the other for each access network; and the two stages can run independently. Note that the first stage in essence is the same as NTD-R/C, but it produces inferior solutions as compared to ours. Specifically, our DPR/C-1 solution, described in Chapter 3, produces topology, on average, with 5.87% higher reliabilities [44] as compared to the optimum solutions presented in Table 2 of [37]. Shao *et al.* [18] have proposed a heuristic approach, called Shrinking and Searching Algorithm, to maximize Rel_{all} given a cost constraint. However, the approach in [18], considers only an upper bound of the network reliability to solve NTD-R/C problem, while we need to compute an exact reliability value.

References ([3], [6], [12]-[15], [28]-[36]) address a closely related O_1/C_1 problem, NTD-C/R, to design a topology with the minimum link cost subject to a reliability constraint. The existing solutions in the literature for the NTD-C/R aim to generate either an optimal topology or an approximated topology from a given network G. The heuristic-based methods mainly use meta-heuristic techniques, *e.g.*, genetic algorithms

(GA) ([3], [6], [13], [14], [33]), SA [35], tabu search (TS) ([32], [36]) and ant colony optimization (ACO) [12]. Further, all these algorithms consider only Rel_{all} measure.

Jan *et al*. [28] considered a network G whose links have the same reliability values, and developed an algorithm that combines decomposition, Branch and Bound (B&B) techniques to find an optimal solution, *i.e*., a topology G_{min} with a minimum Cost(G_{min}), and a $Rel_{all}(G_{min}) \geq R_{min}$, where R_{min} is the required reliability constraint. Later, Koide *et al.* [29] generalized the problem in [28] for graph G with non-homogeneous link reliabilities, and developed another B&B algorithm to solve the problem. The B&B approaches [28], [29] are computationally expensive, and thus are suitable only for small sized networks with up to nine nodes.

Kumar *et al.* [33] have developed a GA-based approach to solve NTD-C/R that includes two additional constraints, *i.e*., diameter and average distance, and applied it to four test networks with up to nine nodes. Their approach [33] calculates the network reliability exactly. Although the problem in [33] is a superset of NTD-C/R, its solution cannot be used to solve the NTD-C/R problem because the problem considers only links with identical reliability and cost. Deeter and Smith [6] presented a GA approach to solve the NTD-C/R problem. Their solution considers a set of link types where links in each type have the same cost and reliability. In contrast, for NTD-C/R problem, each link may have different reliability and cost values, and thus the solution cannot be used to solve the problem.

Dengiz *et al*. [13] proposed a heuristic GA approach, called NGA, to solve the NTD-C/R problem. In [14], the same authors have developed another GA-based solution, called Local search GA (LS-NGA), using a special encoding structure, crossover, and mutation operators. Both techniques have been shown effective in generating near

optimal solutions [13], [14]. These GA methods yield poor quality solutions for networks with more than 10 nodes as reported by Marquez and Rocco [34]. Later, Gen [3] proposed a self-controlled GA to solve the NTD-C/R problem. However, these GA methods ([3], [6], [13], [14], [33]) require the development, coding, and testing of a problem-specific GA, complicating the solution process.

Abo Elfotoh and Al-Sumait [30] developed a neural network (NN) heuristic algorithm, and the authors in [32] used an artificial NN for the NTD-C/R problem. As stated in [34], while the NN and artificial NN algorithms produce good results, they use a long procedure that needs extensive time and significant parameter tuning [34]. Pierre *et al.* [35] used SA to find optimal designs for packet switch networks where delay and capacity were considered, but reliability was not. Recently, a new meta-heuristic called Cross-Entropy method [31] was developed for the NTD-C/R problem. In addition, the Multiple TS algorithm [36] was used to solve the NTD-C/R problem with 19 nodes; however, the algorithm may not reach the global optimum solution in a reasonable computation time when the initial solution is far away from the region where the optimum solution exists. Hardy *et al.* [15] proposed to use Binary Decision Diagram (BDD) 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* to implicitly represent the entire set of the functioning and failing network states. Our simulations in Chapter 4 show that DPC/R-2 produce better results as compared to the BDD approach in [15].

Marquez and Rocco [34] have presented a population-based heuristic approach called the probabilistic solution discovery algorithm. However, their approach is shown less effective as compared to the more recent approach in [12]. Altiparmak *et al*. [12]

proposed a hybrid approach based on Ant Colony Optimization and Simulated Annealing, called ACO-SA, to solve NTD-C/R for networks with up to 50 nodes. ACO-SA first generates a ring network, onto which it adds some links to produce a seed network topology that satisfies the reliability constraint; initially, the seed is considered the best global topology. Then, it uses ACO and SA to reduce the cost of the global topology. If a population in ACO contains better networks than the global topology, then these topologies are put in a set, and SA is used as a local search procedure to further improve those networks in the set to generate the best possible topology. ACO-SA repeats these procedures until a given stopping criterion is met. Our simulations, described in Chapter 4, show that DPC/R-1 outperforms ACO-SA.

In general, each NTD-C/R solution requires calculating Rel_{all} or Rel_2 to be compared with the required reliability constraint R_{min}. The calculation can use either a simulation or an analytic method. The analytic method [22] produces an exact reliability result. However, its time complexity grows exponentially in the order of network size, and thus approaches that use the analytic method, *e.g.*, [28], [29], are suitable only for use in small sized networks. The simulation methods for reliability calculation [25], [10] reduce the time complexity, but produce only estimated reliability values, acceptable for the heuristic solutions to NTD-C/R because the values are used only to test for the feasibility of the resulting topology. Note that each estimated value is used only to see if a solution is feasible; an exact reliability solver is needed only to compute the reliability of the final solution. In contrast, a NTD-R/C solver needs to compute exact Rel_{all} or Rel_2 values to search for the topology with the maximum reliability; otherwise, the search might lead to a local maximum. Therefore, the existing meta-heuristic can be used for computing NTD-R/C only if it uses exact reliability values in

each of its steps, which is computationally expensive because, for example, a GA needs the reliability value for each chromosome reevaluation.

In contrast, the population based meta-heuristics, such as GA, require numerous iterations before converging. Note that, for the NTD-C/R problem, each iteration includes reliability computation of the approximated topology to be tested against the required reliability constraint. Because reliability evaluation, using both exact or approximation methods, is computationally expensive (a typical MCS iterates 10^6 times), the approach in general uses a considerable computational effort. For example, the GA based approach in [6] uses MCS 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 MCS 240000 times. Therefore, the approaches are computationally expensive for use in large networks. As described in Chapter 4, our proposed approach DPC/R-2 uses a MCS [10] to estimate the reliability value of each topology, and runs the simulation only up to $\sigma \times k$ times while producing 81.5% best results, for σ=100, and $k \leq 1214$.

2.5.2 O_2/C_1 NTD Problems

This subsection presents related works to two O_2/C_1 NTD problems: (i) NTD-CR/B that maximizes network reliability and minimizes its cost subject to a bandwidth constraint; (ii) NTD-CB/R that maximizes network bandwidth and minimizes its cost subject to a reliability constraint. Note that NTD problem with multiple objectives has not been addressed in the literatures as much. In the rest of the thesis, we call problem involving two objectives as Bi-Objective (BO).

Local search techniques have been frequently used to solve NTD problems. However, these techniques generally do not perform well when BO needs to be optimized and/or

constraints are present [38]. Duarte and Bar´an [39] proposed a parallel GA to solve network design problems with cost and reliability as objectives, and other constraints but not bandwidth, unlike NTD-CR/B. Kumar *et al*. [40] presented a GA approach called Pareto Converging Genetic Algorithm (PCGA) to design a network that simultaneously optimizes network delay and design costs under a reliability and bandwidth constraint. Banerjee and Kumar [41] studied BO network design using a GA heuristic and empirically showed that the GA generally provides better solution than its deterministic counterparts. Papagianni *et al*. [8] used the particle swarm (PS) optimization to solve multi criteria network design problem. Papagianni *et al*. [8] claimed that the approach is more effective than the GA in [41] but the algorithm was only tested on a 16-node network and nothing was said about its efficiency.

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. Therefore, a more time efficient heuristic approach that can produce better results is still needed, especially for use in large scaled networks. Recursive heuristics such as *Dynamic programming* (DP) have a tendency to escape a local optimum and can often find a global optimum solution in a reasonable amount of computational time. Recently, Figueira *et al.* [42] presented a new DP approach to solve the Multiple Objectives Integer Knapsack Problem (MOIKP). This problem consists of determining the numbers of copies for each item such that the overall weight does not exceed W and the total profits are maximized. As later discussed in Section 5.2, MOIKP approach proposed by Figueira *et al.* [42] cannot be used to solve NTD-CR/B and NTD-CB/R.

This thesis proposes to use DP to solve the four NTD problems, *i.e.*, NTD-R/C, NTD-C/R, NTD-CR/B and NTD-CB/R, shown in Table 1.1 and Table 1.2. Our solutions are similar to the well-known NP-complete 0-1 knapsack problem [43]. Given a set of items where each item has value and weight, 0-1 knapsack problem selects a set of items such that their total value is maximized and their total weight is no larger than a given weight constraint. Note that the items in the problem are either spanning trees or (*s*, *t*) paths in our problems. Further, comparing 0-1 knapsack to NTD-R/C, the value and weight in the former are, in the latter, Rel_{all} or Rel_2 and cost, respectively. The 0-1 knapsack problem can be efficiently solved using a DP approach, which is a stage-wise search method. However, as discussed in Section 3.3.3.1 and 4.2, one cannot directly use the efficient solution to 0-1 knapsack for our problems. Nevertheless, using DP for the four NTD problems is advantageous because this strategy prunes earlier partial solutions that are not expected to lead to an optimal solution. Further, it solves each sub-problem only once by saving the solution in a table for later use. This thesis describes bottom-up DP solutions for the four NTD problems and discusses the differences between the DP for these problems and 0-1 knapsack problem.

2.6 Simulations Environment

2.6.1 Topologies

The performance of the algorithms proposed in this thesis is evaluated using three sets of topologies: (i) 20 arbitrary network topologies, (ii) 76 fully connected topologies, and (iii) five grid topologies.

As shown in Table 2.3 and Table 2.4, the number of nodes and links of the topologies in (i) range from 4 to 21 and 5 to 30, respectively; see Fig C.1 in Appendix C for the description of each CN configuration. Note that we obtain 19 of the 20 topologies in the tables from [22]. We use the 20 networks in Chapter 3, 4 and 5 to observe the effects of using different orders of spanning tree and (*s*, *t*) paths on the effectiveness of our algorithms. As shown in Table 2.3, the number of spanning trees in the CNs ranges from 8 to 24173; each $CN_n^{|V|,|E|}$ in Table 2.3 denotes a CN with |V| nodes, |E| links and *n* spanning trees. Table 2.4 shows that the CNs have between 7 and 780 (*s*, *t*) paths. Each $CN_m^{|V|,|E|}$ denotes a CN with |V| nodes, |E| links and *m* (*s*, *t*) paths; in Fig C.1 in Appendix C, source (terminal) node *s* (*t*) is the node with the smallest (largest) number and for each of the 20 CNs we randomly assigned r_j, c_j and b_j for each link. For Table 2.3 and 2.4, we use CAREL [22] to compute $Rel_{all}(G)$ and $Rel_2(G)$ for each original CN (*i.e.*, with complete links), and the Max-Flow Min-Cut algorithm [26] for each BW(G); Cost (G) is directly computed by taking the sum of c_j for all links in G. Further, we set R_{min}, B_{min} and C_{max} randomly between 40% to 90% of $Rel_{all}(G)$ or $Rel_2(G)$, BW(G) and Cost (G), respectively. Table 2.3 and 2.4 show the results (see the % value in bracket next to each R_{min}, B_{min} and C_{max} value). As an example, for $CN_8^{4,5}$ in Table 2.3, C_{max}=Cost($CN_8^{4,5}$)×0.90=18.

For the networks in set (ii), we obtained the 76 fully connected networks from [12]. We used the networks to evaluate the performance of our DPC/R-2, presented in Chapter 4, against four state-of-the-art approaches, *i.e.*, ACO-SA [12], NGA [13], LS-NGA [14], and BDD [15]. Note that we obtained the results for each of the four approaches as reported in their corresponding papers. The CNs have 6 to 11, 15 to 55, and 1269 to 2.3×10^9 nodes, links, and spanning trees respectively. We obtained the 26 cost matrices of the fully connected networks from the authors in [12], and use them for all link costs of all networks. Note that the authors in [12] randomly generated the integer costs with values between 1 and 100. We presented the cost matrices in Appendix C. Like in [12], we set R_{min} to either 0.9 or 0.95, and use equal link reliabilities with values of either 0.9 or 0.95. The 76 topologies are generated from the 26 cost matrices as follows. Using the first 25 cost matrices with link reliability of 0.9 and R_{min}=0.9 or R_{min}=0.95, we obtain 2×25=50 different CNs. Then, setting link reliability of 0.9 and R_{min}=0.95 on the 25 cost matrices, we generate 25 additional topologies. The last topology is obtained by using the last cost matrix for the fully connected topology G with 11 nodes with link reliability of 0.9 and R_{min}=0.9. Note that the 76 topologies will be used in Table 4.4 of Chapter 4. Table 2.5 shows the five grid networks in set (iii), *i.e.*, $Grid_{6\times6}$, $Grid_{3\times12}$, $Grid_{3\times16}$, $Grid_{2\times20}$, $Grid_{2\times100}$, that contain 36 to 200 nodes, and 57 to 298 links respectively. We use the networks in Chapter 3, 4 and 5 to evaluate the efficiency of the corresponding proposed algorithms, in terms of its required number of spanning trees $k\leq n$ or paths $k\leq m$, and CPU time as well as its effectiveness.

Table 2.3: Topologies information used for the simulations with spanning trees

CN	Original *CN*			Cost		Reliability	
	|V|	|E|	*n*	Cost*(CN)*	C_{max}	$Rel_{all}(CN)$	R_{min}
$CN_{8}^{4,5}$	4	5	8	20	18(90%)	0.927	0.82(88%)
$CN_{21}^{5,8}$	5	8	21	29	24(82%)	0.972	0.8(82%)
$CN_{21}^{6,8}$	6	8	21	29	20(69%)	0.968	0.80(82%)
$CN_{55}^{6,9}$	6	9	55	31	25(81%)	0.977	0.8(81%)
$CN_{368}^{7,12}$	7	12	368	40	33(83%)	0.969	0.8(82%)
$CN_{1033}^{7,15}$	7	15	1033	46	38(83%)	0.966	0.8(82%)
$CN_{247}^{8,12}$	8	12	247	40	35(81%)	0.946	0.8(84%)
$CN_{256}^{8,12}$	8	12	256	40	35(81%)	0.975	0.8(82%)
$CN_{576}^{8,13}$	8	13	576	43	35(81%)	0.984	0.8(81%)
$CN_{171}^{9,12}$	9	12	171	40	33(83%)	0.964	0.8(81%)
$CN_{327}^{9,13}$	9	13	327	43	35(81%)	0.968	0.8(81%)
$CN_{647}^{9,14}$	9	14	647	47	35(74%)	0.971	0.8(82%)
$CN_{2112}^{10,21}$	10	21	2112	68	55(80%)	0.904	0.75(83%)
$CN_{1598}^{11,21}$	11	21	1598	68	56(81%)	0.974	0.8(81%)
$CN_{3666}^{13,22}$	13	22	3666	71	60(84%)	0.948	0.7(74%)
$CN_{7683}^{16,30}$	16	30	7683	96	80(83%)	0.971	0.8(81%)
$CN_{9471}^{17,25}$	17	25	9471	80	65(81%)	0.980	0.8(81%)
$CN_{21456}^{18,27}$	18	27	21456	87	70(80%)	0.987	0.8(81%)
$CN_{24173}^{20,30}$	20	30	24173	96	80(83%)	0.985	0.8(81%)
$CN_{18257}^{21,26}$	21	26	18257	84	70(83%)	0.982	0.8(81%)

The networks have m=524288 to m=2^{99} (s, t) paths [49]. We use the formula in [20] to count the number of spanning trees in $Grid_{2x\underline{h}}$, *i.e.*, $Grid_{2\times20}$, $Grid_{2\times100}$. However, the paper [20] does not provide other counting equations for the other grid sizes, *i.e.*, $Grid_{6x6}$, $Grid_{3x12}$ and $Grid_{3x16}$. For Table 2.5, we assume that, when two grid networks have the same number of nodes, the wider grid, *e.g.*, $Grid_{3x12}$, contains more spanning trees than the other, *e.g.*, $Grid_{2x18}$. Thus, we show $Grid_{6x6}$, and $Grid_{3x12}$ as $CN_{\geq 3.557^{18}}^{36,60}$, and $CN_{\geq 3.557^{18}}^{36,57}$, respectively; because $Grid_{2x18}$ contains 3.557^{18} spanning trees. Further, we show $Grid_{3x16}$ as $CN_{\geq 2.5^{24}}^{48,77}$ because $Grid_{2x24}$ contains 2.5^{24} spanning trees. As shown in Table 2.5, the five grid networks contain n=3.557^{18} to n=1.899^{102} spanning trees. We set c_j=1 and r_j=0.9 for all of the five grid networks. Because DPR/C-2 in Chapter 3, does not calculate the exact reliability values of its generated topologies for large network, we have used the MCS [10] to compute the estimated

Rel_{all}(CN) and Rel_2(CN) for each generated topology; we used a sample size of 10^6 in the simulation.

Table 2.4: Topologies information used for the simulations with (*s*, *t*) simple paths

	Original *CN*			**Cost**		**Reliability**		**Bandwidth**	
CN	**|V|**	**|E|**	***m***	**Cost(*CN*)**	C_{max}	**Rel_2(*CN*)**	R_{min}	**BW(*CN*)**	B_{min}
$CN_4^{4,5}$	4	5	4	20	18(90%)	0.939	0.85(90%)	8	6(75%)
$CN_9^{5,8}$	5	8	9	29	20(68%)	0.969	0.8(82%)	9	6(66%)
$CN_7^{6,8}$	6	8	7	29	20(68%)	0.902	0.7(77%)	11	9(81%)
$CN_{13}^{6,9}$	6	9	13	31	20(65%)	0.925	0.75(81%)	12	10(83%)
$CN_{25}^{7,12}$	7	12	25	40	20(50%)	0.98	0.80(81%)	16	12(75%)
$CN_{14}^{7,15}$	7	15	14	46	20(43%)	0.968	0.80(82%)	16	12(75%)
$CN_{20}^{8,12}$	8	12	20	40	20(50%)	0.955	0.8(76%)	15	10(66%)
$CN_{24}^{8,12}$	8	12	24	40	20(50%)	0.919	0.60(65%)	12	10(83%)
$CN_{29}^{8,13}$	8	13	29	43	20(47%)	0.965	0.50(51%)	18	14(77%)
$CN_{13}^{9,12}$	9	12	13	40	20(50%)	0.878	0.6(68%)	18	12(66%)
$CN_{18}^{9,13}$	9	13	18	43	20(47%)	0.917	0.75(81%)	17	14(82%)
$CN_{44}^{9,14}$	9	14	44	47	40(85%)	0.932	0.75(80%)	14	10(71%)
$CN_{64}^{10,21}$	10	21	64	68	40(59%)	0.978	0.8(81%)	19	15(78%)
$CN_{18}^{11,21}$	11	21	18	68	20(29%)	0.985	0.82(83%)	13	10(76%)
$CN_{281}^{13,22}$	13	22	281	71	40(56%)	0.983	0.6(61%)	18	10(55%)
$CN_{36}^{16,30}$	16	30	36	96	40(42%)	0.995	0.70(75%)	14	10(71%)
$CN_{136}^{17,25}$	17	25	136	80	40(50%)	0.982	0.60(61%)	13	10(76%)
$CN_{281}^{18,27}$	18	27	281	87	40(46%)	0.973	0.70(72%)	16	10(62%)
$CN_{780}^{20,30}$	20	30	780	96	40(42%)	0.981	0.60(61%)	20	15(75%)
$CN_{44}^{21,26}$	21	26	44	84	40(48%)	0.861	0.60(69%)	16	12(75%)

Table 2.5: Grid topologies information used for the simulations

Original Topology						**Cost(*CN*)**	**Rel_{all}(*CN*)**	**Rel_2(*CN*)**
CN	**Grid**	**|V|**	**|E|**	***n***	***m***			
$CN^{36,57}_{\geq n\ (or\ m)}$	$Grid_{3x12}$	36	57	$\geq 3.557^{18}$	1262816	57	0.9130	0.9756
$CN^{36,60}_{\geq n\ (or\ m)}$	$Grid_{6x6}$	36	60	$\geq 3.557^{18}$	538020	60	0.9173	0.9613
$CN^{40,58}_{\geq n\ (or\ m)}$	$Grid_{2x20}$	40	58	2^{20}	524288	58	0.74530	0.7844
$CN^{48,77}_{\geq n\ (or\ m)}$	$Grid_{3x16}$	48	77	$\geq 2.5^{24}$	64019921	77	0.7218	0.9562
$CN^{200,298}_{\geq n\ (or\ m)}$	$Grid_{2x100}$	200	298	1.899^{102}	2^{99}	298	0.25107	0.3042

2.6.2 Input Orders

As later described in Chapter 3 to 5, the performance of each algorithm proposed in this thesis is significantly affected by different way of input (*i.e.*, spanning trees and (*s*, *t*) paths) orders. Further, as later shown in Chapter 3 and 4, generating optimal input order is an NP complete problem. Therefore, this thesis proposes several possible input

orders that, while not optimal, help each algorithm in producing near optimal results. Table 2.6 summarizes 10 Order Criteria (OC) that are used to generate 10 different input orders for both spanning trees and (*s*, *t*) paths.

Table 2.6: The function and uses of proposed order criteria

Order Criteria	Function	Used by	
		Algorithm	Chapter
OC1	$w_j=c_j/r_j$	DPR/C-1, DPR/C-2, DPC/R-1, DPC/R-2, DPCR/B-1, DPCR/B-2, DPCR/B-P	Chapter 3, 4 and 5
OC2	$w_j=c_j$	DPR/C-1, DPR/C-2, DPC/R-1, DPC/R-2	Chapter 3, 4
OC3	$w_j=(-\log r_j)$	DPR/C-1, DPR/C-2, DPC/R-1, DPC/R-2, DPCB/R-1, DPCB/R-2, DPCB/R-P	Chapter 3, 4 and 5
OC4	$w_j=(-\log r_j)+c_j$	DPR/C-1, DPR/C-2, DPC/R-1, DPC/R-2	Chapter 3, 4
OC5	$w_j=(-\log r_j)\times c_j$	DPR/C-1, DPR/C-2, DPC/R-1, DPC/R-2	Chapter 3, 4
OC6	$w_j=c_j+\eta_w\times(-\log r_j)$	DPC/R-1, DPC/R-2	Appendix B
OC7	$w_j=c_j\times r_j+\eta_w\times b_j$	DPCR/B-1, DPCR/B-2, DPCR/B-P	Chapter 5
OC8	$w_j=c_j\times b_j+\eta_w\times r_j$	DPCB/R-1, DPCB/R-2, DPCB/R-P	Chapter 5
OC9	$w_j=b_j$	DPCR/B-1, DPCR/B-2, DPCR/B-P	Chapter 5
OC10	$w_j=c_j/b_j$	DPCB/R-1, DPCB/R-2, DPCB/R-P	Chapter 5

Each order uses a function to generate a weight w_j from the cost, reliability and/or bandwidth value of each link $e_j \in E$, *i.e.*, c_j, r_j, and b_j respectively; see column Function. Note that the parameter η_w in OC6, OC7 and OC8 is a Lagrange multiplier whose value determines the optimality of the conversion; Section 5.3.1, Section 5.3.2, and Appendix B describe how to compute the multiplier in detail. The input is then sorted in increasing order of weight of the spanning trees or (*s*, *t*) paths. Notice that the spanning tree and (*s*, *t*) path weight is calculated as the summation of the weight of each link in the spanning tree and (*s*, *t*) path, respectively. The order criteria will be

described in more details when they are used in their corresponding algorithms and chapters; see columns Algorithm and Chapter.

2.6.3 Platform

All proposed algorithms were written in C and all simulations were run on an Intel Core i5 (2 cores) with 2.53 GHz and 4 GB of RAM, running Linux (Ubuntu Core 11.10).

2.7 Chapter Summary

This chapter has presented a CN model, notations and definitions that are used throughout this thesis. It has also described methods to compute network reliability and bandwidth. Further, it has presented concepts of DP and POS, and a review of the literature that is relevant to this thesis presented. The previous related solutions are categorized according to the number of objectives, *i.e.*, single or multiple, addressed by the solutions, which correspond to the problems addressed in Chapters 3, 4 and 5. The differences between the existing solutions from the algorithms proposed in this thesis have been discussed. Finally, the chapter has presented a description of the simulations environment used to evaluate the algorithms in this thesis.

Chapter 3

Network Topology Design with Maximum Reliability and Cost Constraint

This chapter addresses NTD-R/C, an O_1/C_1 optimization problem, to construct the network topology with maximum reliability while satisfying a given cost constraint. The solution of this problem is important for critical/sensitive applications like emergency response in hospitals. As discussed in Section 2.5.1, we found only a few papers, *i.e.*, [16]-[19], [37], that address this important NTD-R/C problem.

The chapter proposes a DP algorithm called DPR/C, to solve the NTD-R/C problem by generating the topology using a sequence of spanning trees to maximize Rel_{all}. The first version of DPR/C, called DPR/C-1, requires all n spanning trees to maximize Rel_{all}. The chapter proves that DPR/C-1 produces optimal NT given as input an optimal sequence of spanning trees. The chapter shows that generating optimal input order is NP-complete, and describes five order criteria to heuristically generate the best sequence of spanning trees that allow DPR/C-1 to produce near optimal results. Further, the chapter describes the second version of DPR/C, called DPR/C-2, which uses only $k \leq n$ spanning trees to improve the time efficiency of DPR/C-1 while producing similar results. Note that Appendix A shows how to use both versions of DPR/C for Rel_2 metric.

The work in this chapter has been published in [44] and [45]. Specifically, reference [44] addresses NTD-R/C with Rel_{all} measure and describes both versions of DPR/C, while reference [45] considers NTD-R/C for Rel_2 measure. Note that the DPR/C-1 and

DPR/C-2 approaches presented in this chapter were, respectively, called DPA-1 and DPA-2 in [44], and DPR/C-1 was called DPA-P in [45].

The layout of this chapter is as follows. Section 3.1 formulates the NTD-R/C problem, and Section 3.2 shows its DP formulation. Section 3.3 describes our proposed DPR/C algorithm to solve NTD-R/C. The section discusses DPR/C-1 with an illustrating example and its analysis. Further, it proposes five order criteria, and DPR/C-2. Section 3.4 presents the simulation results. Finally, Section 3.5 concludes this chapter.

3.1 Problem Statement

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-R/C problem.

$$\text{Maximize } Rel_{all}(G'=(V, E')) \tag{3.1}$$

$$\text{Subject to } \sum_{j=1}^{|E|} c_j Y_j \le C_{max} \tag{3.2}$$

Equation (3.1) calculates $Rel_{all}(G')$ of a network topology G' that contains links $E'=E-\{e_j \mid Y_j=0\}$; *i.e.*, E' is a set of selected links in Eq. (3.2) that form G', and Eq. (3.2) is Cost(G'). One may solve the NTD-R/C problem by generating each possible set of links in Eq. (3.2) that form G'. Then, for each set that has a cost of at most C_{max}, calculate its $Rel_{all}(G')$, and use Eq. (3.1) to select G' with the maximum $Rel_{all}(G')$ as an optimal topology G_{opt}. Unfortunately, this *Brute Force* (BF) solution requires generating $2^{|E|}$ possible link selections. Further, the reliability calculation in Eq. (3.1) for each G' requires exponential time. The solution, called BF-1, is feasible only for designing small sized topology. We have used BF-1 to generate the benchmark networks in Section 3.4.2.

As an alternative, let X_i be a decision variable {0, 1} that indicates if spanning tree ST_i in G=(V, E) is selected (X_i=1) or not selected (X_i=0). The following equations describe the NTD-R/C problem.

$$\text{Maximize} \quad \text{Rel}_{all}\left(\bigcup_{i=1}^{|ST_G|} ST_i \; X_i\right) \tag{3.3}$$

$$\text{Subject to Cost}\left(\bigcup_{i=1}^{|ST_G|} ST_i \; X_i\right) \leq C_{max} \tag{3.4}$$

Equation (3.3) calculates the maximum Rel_{all} using only the selected spanning trees ST_i that meet constraint Eq. (3.4). One may generate all 2^n possible spanning tree set combinations in Eq. (3.4). Then, for each combination that has a total link cost of at most C_{max}, calculate its reliability, and use Eq. (3.3) to select a combination with the maximum Rel_{all} as its G_{opt}. This solution, henceforth called BF-2, is also prohibitive for use in large networks because in general a network contains $n=O(|V|^{|V|})$ spanning trees [20].

To illustrate the NTD-R/C problem, consider the CN in Fig. 2.1. For C_{max}=18, Fig. 3.1 shows the optimal topology, G_{opt}, whose links form a set of spanning trees {{2,4,5}, {1,4,5}, {1,2,4}, {1,2,5}} with $\text{Rel}_{all}(G_{opt})$=0.874, and $\text{Cost}(G_{opt})$=18; G_{opt} does not contain spanning trees {1,3,4}, {1,3,5}, {2,3,5}, and {2,3,4} because link 3 is not selected. Notice that Cost(G)=20, and therefore if we set $C_{max}\geq$20, Eq. (3.2) would have Y_j=1 for each e_j, and Eq. (3.3) considers all spanning trees in G. Thus, for this case, Eq. (3.1) and Eq. (3.3) produce G_{opt}=G with $\text{Rel}_{all}(G_{opt})$=0.927. In contrast, Eq. (3.1) produces a reliability of 0 if the selected links in Eq. (3.2) do not form any spanning tree. Further, if C_{max} is set smaller than the minimum of $\text{Cost}(ST_i)$ for each $ST_i \in ST_G$, then no spanning tree can be selected in Eq. (3.3), and thus this case also produces a reliability of 0. Thus, this chapter sets C_{max} to a value no less than the minimum among $\text{Cost}(ST_i)$ to guarantee a feasible solution.

We observe that NTD-R/C problem is similar to the the 0-1 knapsack problem [43] in which there are n items (*i.e.*, spanning trees), where each item i has capacity x_i (*i.e.*, Cost(ST_i)), and value v_i (*i.e.*, $Rel_{all}(ST_i)$), and its goal is to select a set of items that have the maximum total value (*i.e.*, Eq. (3.3)) while having a total capacity of no more than a given capacity constraint X_{max} (*i.e.*, Eq. (3.4)). The 0-1 knapsack problem is solvable using a dynamic programming (DP) approach in a pseudo-polynomial time $O(n \times X_{max})$ [43]. In Section 3.2, we propose a similar DP approach to solve Eq. (3.3) and Eq. (3.4).

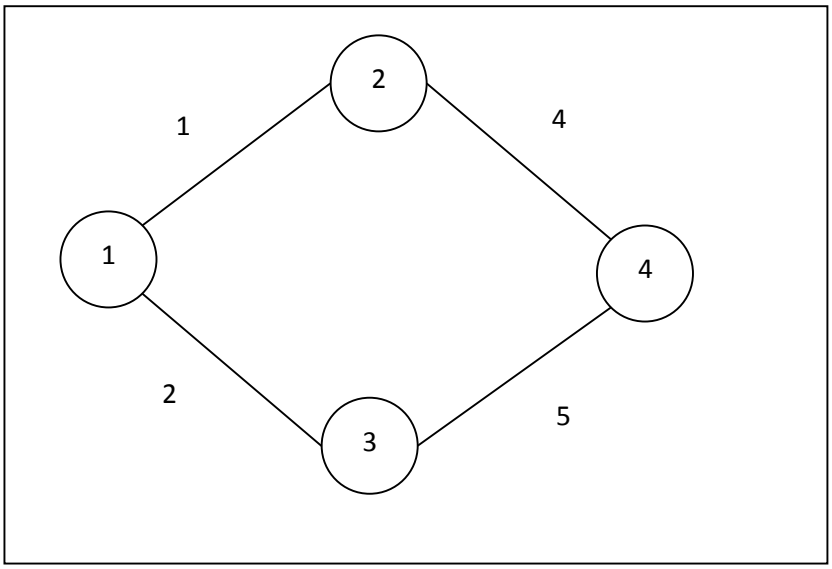


Fig. 3.1: Optimal solution of NTD-R/C with C_{max}=18 for CN in Fig. 2.1

3.2 Dynamic Programming Formulation for NTD-R/C

Let STX_i, for i=1, 2, …, n-1, n, be a sequence of spanning trees selected from i spanning trees in {ST_1, ST_2, …, ST_i}, and G_i=(V, $E_i \subseteq E$) be an induced graph whose links comprise all links in STX_i; thus STX_i can be generated from its G_i, and vice versa. Note that $0 \leq |STX_i| \leq i$, and there are 2^n different STX_i. We aim to select STX_n

with a cost of at most C_{max}, and the maximum reliability, *i.e.*, $\text{Cost}(\text{STX}_n) \leq C_{max}$ with maximum $\text{Rel}_{all}(\text{STX}_n)$.

Let DP[1.. n, 0.. C_{max}] be a 2-dimensional dynamic programming table. Each element DP[i, c], for i=1, 2, …, n, c=0, 1, 2, …, C_{max}, stores five pieces of information: a reliability $0 \leq R[i, c] \leq 1.0$, a sequence of spanning trees $X[i, c] \subseteq ST_G$, a cost $0 < C[i, c] \leq c$, a set of links $L[i, c] \subseteq E$, and an integer index $0 \leq J[i, c] \leq C_{max}$. Let $R[i, c] = \text{Rel}_{all}(\text{STX}_i)$, for c=0, 1, …, C_{max}, be the maximum Rel_{all} of STX_i subject to $\text{Cost}(\text{STX}_i) \leq c$, *i.e.*, $R[i, c] = \text{Max}\{\text{Rel}_{all}(\bigcup_{j=1}^{i} ST_j X_j)\}$ subject to $\text{Cost}(\text{STX}_i) \leq c$. In other words, R[$i$, c] is the Rel_{all} of the best subset STX_i that contains one or more spanning trees in $\{\text{ST}_1, \text{ST}_2, \ldots, \text{ST}_i\}$ with total cost at most c, and maximum Rel_{all}. Note that R[n, C_{max}] stores the Rel_{all} of STX_n subject to $\text{Cost}(\text{STX}_n) \leq C_{max}$, and NTD-R/C aims to find the best topology STX_n, *i.e.*, G_{opt}.

Let X[i, c] be the STX_i that produces R[i, c], C[i, c]=Cost(X[i, c]), and L[i, c] store links that are used in X[i, c]. For each range of columns $c1 \leq c \leq c2$ in row i that contain the same cost value, we set each $J[i, c] = c1$. Thus, index $J[i, c]$=0, 1, 2, …, C_{max} marks the starting column of a range of columns that have the same cost. For example, as later shown in Table 3.1, we store $J[1, c]$=13 at columns c=13 to c=20. Note that $\text{Cost}(\text{ST}_i - \text{X}[i\text{-}1, c])$ computes the cost of links in ST_i that are not in X[i-1, c].

Each R[i, c] is calculated using a dynamic programming (DP) formulation defined in the following four equations:

$R[i, c] = 0$, for i=1 with $\text{Cost}(\text{ST}_1) > c$ (3.5)

$R[i, c] = \text{Rel}_{all}(\text{ST}_1)$, for i=1 with $\text{Cost}(\text{ST}_1) \leq c$ (3.6)

$R[i, c] = R[i\text{-}1, c]$, for each i>1 and j=0, …, c with $\text{Cost}(\text{X}[i\text{-}1, j] \cup \text{ST}_i) > c$ (3.7)

$R[i,c]=\text{Max}(R[i\text{-}1,c],\ \text{Rel}_{all}(X[i\text{-}1,\ j] \cup ST_i))$, for each $i>1$ and $j=0, \ldots, c$ with

$$c \leq \text{Cost}(X[i\text{-}1,j] \cup ST_i) \leq C_{max} \quad (3.8)$$

Our DP formulation in Eq. (3.5) - Eq. (3.8) is similar to the DP formulation for the 0-1 knapsack problem [43]; we will describe the differences in subsection 3.3.3.1. We explain the DP formulation in Eq. (3.5) - Eq. (3.8) using a similar explanation for the 0-1 knapsack problem [43]. 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. In Eq. (3.5), because $\text{Cost}(ST_1)>c$, *i.e.*, we are over budget, ST_1 cannot be selected; thus $R[1, c]=0$. In contrast, the cost of ST_1 in Eq. (3.6) is within budget c, and thus it is selected, giving $R[1, c]=\text{Rel}_{all}(ST_1)$.

Equations Eq. (3.7) and Eq. (3.8) consider either selecting or not selecting each ST_i, for $i>1$, together with some previously selected spanning trees STX_{i-1} such that the total cost of the spanning trees is at most c, *i.e.*, $\text{Cost}(X[i\text{-}1, j] \cup ST_i) \leq c$, for each possible $j=J[i\text{-}1, c]=0, 1, \ldots, C_{max}$, and their induced topology has the maximum Rel_{all}. The best STX_i (*i.e.*, the set of spanning trees stored in $X[i, c]$ that has the highest $\text{Rel}_{all}(STX_i)$ with a total cost at most c) is either (i) the best subset STX_{i-1} stored in $X[i\text{-}1, c]$ that has a total cost at most c, *or* (ii) the best subset STX_{i-1} stored in $X[i\text{-}1, j]$ that has a total cost at most j *plus* spanning tree ST_i such that $\text{Cost}(X[i\text{-}1, j] \cup ST_i)$ is at most c, where $j=0, 1, \ldots, c$. Each generated STX_i is the best subset because it is obtained from either including spanning tree ST_i or not; either option should lead to the best subset.

Equation (3.7) and Equation (3.8) compute a sub-problem STX_i that involves two parameters, i and c, and they aim to generate each sub-problem that forms a sub-topology with maximum Rel_{all} within a cost constraint $c=1, 2, \ldots, C_{max}$. In Eq. (3.7),

when Cost(X[i-1, j] ∪ ST_i)>c, ST_i cannot be selected, and thus R[i, c]=R[i-1, c]; *i.e.*, the maximum Rel_{all} possible using all or a subset of spanning trees in ST_1, ST_2, …, ST_{i-1}, ST_i is obtained from the maximum reliability possible using all or a subset of the spanning trees in ST_1, ST_2, …, ST_{i-1}. In Eq. (3.8), selecting ST_i is possible because c≤Cost(X[i-1, j] ∪ ST_i)≤C_{max}. If ST_i is not selected, the potential maximum Rel_{all} would come from selecting spanning trees ST_1, ST_2, …, ST_{i-1} with unchanged budget c; *i.e.*, R[i, c]=R[i-1, c]. If ST_i is selected, the remaining allowable cost for selecting spanning trees ST_1, ST_2, …, ST_{i-1}, ST_i would be reduced from c to Cost(X[i-1, j] ∪ ST_i), and the resulting reliability would be Rel_{all}(X[i-1, j] ∪ ST_i). Thus, Eq. (3.8) sets R[i, c] to the maximum between the two potential Rel_{all} values. When the two options produce the same Rel_{all}, our implementation selects the one with lower cost. Note, for each i>1, Eq. (3.8) considers an option of selecting only ST_i when j=0.

3.3 DPR/C Algorithm

Section 3.3.1 describes the first version of DPR/C, called DPR/C-1, which requires all spanning trees to maximize Rel_{all}. Section 3.3.2 shows an illustrating example for DPR/C-1, and Section 3.3.3 discusses the DPR/C-1 analysis; specifically, subsection 3.3.3.1 compares the similarity and difference between NTD-R/C and 0/1 knapsack and subsection 3.3.3.2 shows that DPR/C-1 is optimal if the spanning trees are optimally ordered. Section 3.3.4 proposes five heuristic order criteria, OC1, OC2, OC3, OC4, and OC5 and describes a method to generate the spanning trees in increasing lexicographic order to improve the efficiency of DPR/C-1. Finally, Section 3.3.5 describes the second version of DPR/C, called DPR/C-2. Note that Appendix A shows how to use DPR/C for NTD-R/C with Rel_2 measure. For this case, DPR/C uses (s, t) simple paths of the network as input. The appendix also presents an illustrating

example, and shows how the five OCs make DPR/C-2 produce near optimal results using only $k \leq m$ paths.

3.3.1 DPR/C Version 1

The first version of DPR/C algorithm, called DPR/C-1, uses bottom-up dynamic programming and directly applies the DP formulation in Eq. (3.5) - Eq. (3.8). For a G=(V, E) that contains n spanning trees with cost constraint C_{max}, DPR/C-1 *implicitly* constructs a DP table of size $n \times C_{max}$. However, DPR/C-1 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, DPR/C-1 computes DP[2, c] in *row*2 using the information in DP[1, c] in *row*1, for all relevant columns c. After copying the contents of *row*2 to *row*1, it repeats the step until all spanning trees have been considered. Line 1 implements Eq. (3.5) while Line 2 to 8 are based on Eq. (3.6). The remainder of the code is used to implement Eq. (3.7) and Eq. (3.8). Specifically, Eq. (3.7) is solved in Line 9, and Eq. (3.8) is solved in Lines 10 to 28. Line 27 copies the contents of *row*2 to *row*1.

Function Cost(•) in the DPR/C-1 algorithm computes the total cost of the union of links in spanning tree set $\bullet = \{X[i\text{-}1, c] \cup ST_i\}$, and function $Rel_{all}(\bullet)$ calculates the Rel_{all} of the network using only spanning trees in set •. The Cost(•) function requires all unique links used in spanning tree set •. For each c, Cost(•) returns the sum of C[i-1, c] and the cost of links in ST_i that are not in L[i-1, c]. Using the bit implementation [22], one requires only 1-bit OR and 1-bit XOR operation to obtain the links in ST_i that are not in L[i-1, c]; and thus, for any •, Cost(•) can be computed in $O(|E|)$. DPR/C-1 uses the function only for each different j in each row i, at most once for every table

entry, and therefore the worst case time complexity for using the function is $O(n\times|E|\times C_{max})$.

For $Rel_{all}(\bullet)$ function, we propose using CAREL [22], a sum of disjoint products (SDP) technique, to compute the exact value of $Rel_{all}(\bullet)$; see Section 2.2.1 for more explanation about how to use CAREL. We use CAREL to compute the reliability contribution of each selected ST_i, *i.e.*, $\Re_i$. Specifically, for each ST_i, CAREL computes $\Re_i$, and we calculate $Rel_{all}(X[i\text{-}1, c] \cup ST_i)=R[i\text{-}1, c]+\Re_i$. Section 3.3.2 gives some examples of using CAREL to compute $Rel_{all}(X[i\text{-}1, c] \cup ST_i)$.

To illustrate the benefits of using CAREL to compute $Rel_{all}(\bullet)$, consider $Grid_{3x12}$ in Table 3.9 (discussed later Section 3.4.3). Computing upper bounds of $Rel_{all}(\bullet)$ using the MCS [10] (with sample size 10^6) produces topologies with maximum reliabilities of 0.492, 0.611, and 0.725 for C_{max} of 30, 45, and 54, which are lower than those produced using $Rel_{all}(\bullet)$ computed by CAREL, *i.e.*, 0.5371, 0.6402, and 0.7819, respectively. On average, computing $Rel_{all}(\bullet)$ using MCS produces topologies with 6.68% lower reliabilities as compared to using CAREL in NTD-R/C problem.

As described in [22], the complexity to compute each reliability $Rel_{all}(\bullet=X[i\text{-}1, c] \cup ST_i)$ depends on the total number of disjoint terms, $|DT_i|$, generated for ST_i, *i.e.*, $O(|E|\times|DT_i|)$. However, $Rel_{all}(\bullet)$ is used only if $Cost(\bullet)\leq C_{max}$. Let Ω be the total number of cases where $Cost(\bullet)\leq C_{max;}$ and $\Omega\leq n\times C_{max}$. Therefore the time complexity of using $Rel_{all}(\bullet)$ is $O(\Omega\times|E|\times T_{max})$, for $T_{max}=\max\{|DT_i|\}$. Thus, in the worst case, DPR/C-1 requires $O(\Omega\times|E|\times T_{max}+n\times|E|\times C_{max})$. Note that we used symbol ψ in (Elshqeirat *et al.* 2014a) instead of Ω.

We consider a small value of C_{max}, *e.g.*, 1000. For a network design that requires a larger cost constraint, *e.g.*, C_{max}=10000, one can scale every x columns in the larger

table, *e.g.*, x=10000/1000=10, as one column in the smaller table. In other words, for the example, columns 1, and 50 in the smaller table 2×1000 refer to cost constraints 1 to 10, and 500 to 510 in the larger table 2×10000, respectively. However such conversion will add additional computation time.

DPR/C-1 Algorithm:

1. Initialize X[1, c]=(), R[1, c]=0, L[1, c]=( ), C[1, c]=0 and J[1, c]=0 when c<Cost(ST_1) // Eq. (3.5)
2. **for** ($c \leftarrow$ Cost(ST_1) to C_{max}) **do** // Eq. (3.6)
3. X[1, c] $\leftarrow$ ST_1
4. R[1, c] $\leftarrow$ $Rel_{all}(ST_1)$ //Use CAREL
5. L[1, c] $\leftarrow$ L_1
6. C[1, c] $\leftarrow$ Cost(ST_1)
7. J[1, c] $\leftarrow$ Cost(ST_1)
8. **end for**
9. Copy *row*1 to *row*2 // Eq. (3.7)
10. **for** ($i \leftarrow$ 2 to n) **do** // Eq. (3.8)
11. **for** ($Y \leftarrow C_{max}$ downto 0) **do**
12. **while** (J[1, Y]$\neq J$[1, Y-1] || Y=0)
13. j=J[1, Y]
14. **if** (Cost(X[1, j] $\cup$ ST_i)$\leq C_{max}$)
15. **for** ($c \leftarrow$ Cost(X[1, j]$\cup ST_i$) to C_{max}) **do**
16. **if** (R[2, c]<Rel_{all}(X[1, j] $\cup$ ST_i) **then** //Use CAREL
17. X[2, c] $\leftarrow$ X[1, j]+ST_i
18. R[2, c] $\leftarrow$ Rel_{all}(X[1, j] $\cup$ ST_i)// Use CAREL
19. L[2, c] $\leftarrow$ Link(X[1, j] $\cup$ ST_i)
20. C[2, c] $\leftarrow$ Cost(X[1, j] $\cup$ ST_i)
21. J[2, c] $\leftarrow$ Cost(X[1, j] $\cup$ ST_i)
22. **end if**
23. **end for**
24. **end if**
25. **end while**
26. **end for**
27. Copy *row*2 to *row*1
28. **end for**

3.3.2 Illustrating Example

To illustrate DPR/C-1, consider the CN in Fig. 2.1 and C_{max}=18. Table 2.1 shows the network's link reliability and cost, spanning trees, and network reliability and cost. Our DPR/C-1 constructs the DP table shown in Table 3.1, and obtains the optimal topology in Fig. 3.1. For convenience, we show all n=8 rows although our

implementation creates only two rows. Each row of the table considers a ST_i for possible selection, and each column shows the budget cost $c \leq C_{max}$. Because the minimum cost of any spanning tree is 9, and C_{max}=18, DPR/C-1 requires 18-9+1=10 columns. Each element in the table shows the set of spanning trees selected for Eq. (3.3), its reliability, and index j; L[i, c] and C[i, c] are not shown to save space. Following Eq. (3.5) and line 1, the two values at each element in columns 0 to 8 are initialized to {} and 0, respectively. Because Cost(ST_1)=13, lines 2 to 8 initialize the first row for c=9, …, 12 with X[1, c]=(), R[1, c]=0, and J[1, c]=0; and for c=13, …, 18 with X[1, c]=(ST_1), R[1, c]=0.486, and J[1, c]=13. Note that CAREL [22] computes R[1, c] directly from the reliability of each link in the spanning tree, *i.e.*, R[1, c]=$\Re_1$=$Rel_{all}(ST_1)$=$r_2 \times r_4 \times r_5$=0.6×0.9×0.9=0.486.

Table 3.1: DP table for CN in Fig. 2.1 with C_{max}=18

c / ST_i	9	10	11	12	13	14	15	16	17	18
1	X[1,9]=() R[1, 9]=0 J[1, 9]=0	() 0 0	( ) 0 0	( ) 0 0	(1) 0.486 13	(1) 0.486 13	(1) 0.486 13	(1) 0.486 13	(1) 0.486 13	(1) 0.486 13
2	( ) 0 0	( ) 0 0	( ) 0 0	( ) 0 0	(1) 0.486 13	(1) 0.486 13	(2) 0.729 15	(2) 0.729 15	(2) 0.729 15	(1,2) 0.777 18
3	() 0 0	() 0 0	() 0 0	(3) 0.486 12	(3) 0.486 12	(3) 0.486 12	(2) 0.729 15	(2) 0.729 15	(2) 0.729 15	(1,2,3) 0.826 18
4	( ) 0 0	( ) 0 0	(4) 0.378 11	(3) 0.486 12	(3) 0.486 12	(3) 0.486 12	(2) 0.729 15	(2) 0.729 15	(2) 0.729 15	(1,2,3) 0.826 18
5	( ) 0 0	( ) 0 0	(4) 0.378 11	(3) 0.486 12	(3) 0.486 12	(3) 0.486 12	(2) 0.729 15	(2) 0.729 15	(2) 0.729 15	(1,2,3,5) 0.874 18
6	(6) 0.378 9	(6) 0.378 9	(6) 0.378 9	(3) 0.486 12	(3) 0.486 12	(3,6) 0.523 14	(2) 0.729 15	(2) 0.729 15	(2) 0.729 15	(1,2,3,5) 0.874 18
7	(6) 0.378 9	(6) 0.378 9	(7) 0.567 11	(7) 0.567 11	(7) 0.567 11	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(1,2,3,5) 0.874 18
8	(6) 0.378 9	(6) 0.378 9	(7) 0.567 11	(7) 0.567 11	(7) 0.567 11	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(3,6,7) 0.839 14	(1,2,3,5) 0.874 18

Next, for ST_2, we copy *row*1 to *row*2 using line 9, and then use Eq. (3.8) to update *row*2 to improve the reliability in some columns when possible. For j=J[1, 13]=13,

and Cost((X[i-1, j=13]=ST_1) ∪ ST_2)={1, 2, 4, 5}=18≤C_{max}, selecting ST_2 would generate Rel_{all}(X[1, 13] ∪ ST_2)=0.777 as follows. CAREL generates DT_2={$e_1e_4e_5\,\overline{e_2}$ } for $ST_2=e_1e_4e_5$. Note that the Boolean expression in DT_2 is mutually disjoint from $ST_1=e_2e_4e_5$ because $\overline{e_2}$ represents link e_2 being off. Thus, the reliability contribution of ST_2 is $\Re_2=r_1\times r_4\times r_5\times(1-r_2)=0.9\times0.9\times0.9\times0.4=0.291$, and Rel_{all}(X[1, 13] ∪ ST_2)=R[1, 13]+$\Re_2$=0.486+0.291=0.777. On the other hand, if ST_2 is not selected, the maximum achievable reliability is R[1, 18]=0.486; hence ST_2 is selected with ST_1 for c=18. Thus, for this case, DPR/C-1 set X[2, c]=(ST_1, ST_2), R[2, c]=0.777, and J[2, c]=18 at column c=18. When j=J[1, 0]=0, Cost(X[i-1, 0]={} ∪ ST_2)=15<C_{max}. Because Rel_{all}(X[1, 0]={} ∪ ST_2)=0.729>R[1, c]=Rel_{all}(ST_1)=0.486, selecting ST_2 at each column c=15, 16 , …, 17 is better; thus DPR/C-1 keeps columns c=0, 1, …,14 unchanged, and sets X[2, c]=(ST_2), R[2, c]=0.729, and J[2, c]=15 at columns c=15, 16, 17.

As another example, Cost(ST_3)=12 and thus ST_3 can be selected only for columns c≥12. ST_3 is selected at c=12 because Rel_{all}(ST_3)=0.486>R[i-1, 12]; further, ST_3 is also selected at c=13, 14 because its cost is less than Cost(X[i-1, c])=13, while having equal reliability. However, ST_3 is not used at c=15 because Rel_{all}(X[i-1, j=12]=( ) ∪ ST_3)=0.482<R[i-1, 15]=0.729. It also cannot be selected at c=16 or 17 because it's cost together with ST_1 or ST_2 is 18. For c=18, we consider different j values. When j=J[2, c]=18, selecting ST_3 would generate Rel_{all}(X[2, 18] ∪ ST_3)=0.826. Specifically, CAREL generates DT_3={$e_1e_2e_4\,\overline{e_5}$ } that is mutually disjoint from $ST_1=e_2e_4e_5$ and $ST_2=e_1e_4e_5$, and computes the reliability contribution of ST_3 as $\Re_3=r_1\times r_2\times r_4\times(1-r_5)=0.9\times0.6\times0.9\times0.1=0.0486$. Thus, Rel_{all}(X[2, 18] ∪ ST_3)=0.777+0.0486=0.826. On

the other hand, selecting ST_3 with j=0, 13, 15 would generate $Rel_{all}(X[2, 0] \cup ST_3)$=0.482, $Rel_{all}(X[2, 13] \cup ST_3)$=0.534, or $Rel_{all}(X[2, 15] \cup ST_3)$=0.777, respectively; thus, the best solution is to select ST_3 with X[2, 18], *i.e.*, for $j=J[2, c]$=18. Similarly, using Eq. (3.8), lines 10 to 28 set $X[5, 18]=(X[4, 18] \cup ST_5)$, and R[5, 18]=0.874, *i.e.*, $R[5, 18]=Rel_{all}(X[4, 18] \cup ST_5)=0.826+r_1 \times r_2 \times r_5 \times (1-r_4)=0.826+0.9 \times 0.6 \times 0.9 \times 0.10=0.826+0.0486=0.874$. Repeating the steps for ST_6 to ST_8, DPR/C-1 sets $X[8, 18]=(ST_1, ST_2, ST_3, ST_5)$ as its output with R[8, 18]=0.874. The optimal topology G_{opt}, shown in Fig. 3.1, is obtained by selecting all links in the spanning trees (ST_1, ST_2, ST_3, ST_5).

3.3.3 DPR/C-1 Analysis

3.3.3.1 NTD-R/C Versus 0-1 Knapsack

The DP formulation in Eq. (3.5) - Eq. (3.8) is similar to the DP solution for the 0-1 knapsack problem [43]. Let X[i, c] be the maximum value achievable using a knapsack of capacity x and items 1, 2, …, i. For each item i with capacity $x_i \leq x$, the DP for the 0-1 knapsack calculates $X[i, x]=\max\{X[i-1, x], X[i-1, c-x_i]+\text{item } i\}$, which is similar to Eq. (3.8). The first option does not select item i (*i.e.*, ST_i), and thus X[i, x] (*i.e.*, X[i, c]) is set to the maximum value (reliability) using the same x (*i.e.*, c) but from items 1, …, i-1 (*i.e.*, ST_1, …, ST_{i-1}). The second option is to take item i (*i.e.*, ST_i), and the value (reliability) for the option is $X[i, x]=X[i-1, x-x_i]+\text{item } i$ (*i.e.*, $R[i, c]=Rel_{all}(X[i-1, j] \cup ST_i)$).

Our DP solution for NTD-R/C is different from the DP for the 0-1 knapsack problem in two ways. First, for the 0-1 knapsack problem, the total cost of two items is the sum of each item's cost, while for NTD-R/C, $Cost(ST_i)+Cost(ST_j) \geq Cost(ST_i \cup ST_j)$

because spanning trees ST_i and ST_j may contain common links. Therefore, Eq. (3.8) must consider all possible values of j, in contrast to the 0-1 knapsack problem that uses only one value stored in the previous row. Second, for the 0-1 knapsack problem, X[i, x] is optimal when both X[i-1, x] and X[i-1, x-x_i] are optimal; because the value for each capacity x is computed sequentially for each item, the sub problem is always optimal, and therefore, its DP produces an optimal value at X[n, X_{max}], where the set of the items (*i.e.*, ST_{i-1}, …, ST_n) has a capacity of at most X_{max}. However, Eq. (3.8) does not guarantee to produce an optimal R[i, c] because, unlike for the knapsack problem, where the total value of two items is the sum of each item's value, reliability $Rel_{all}(ST_i)+Rel_{all}(ST_j) \neq Rel_{all}(ST_i \cup ST_j)$. Further, even when $Rel_{all}(ST_i) > Rel_{all}(ST_j)$, $Rel_{all}(ST_h \cup ST_i)$ is not always larger than $Rel_{all}(ST_h \cup ST_j)$, for any ST_h. Therefore, for our case, R[i, c] is not necessarily maximum, even when its two sub problems are optimal.

3.3.3.2 Optimality of DPR/C-1

This section establishes Theorem 3.1 and Theorem 3.2. The first theorem shows that DPR/C-1 produces the optimal topology for a given set of optimally ordered spanning trees, called STX_{opt}, defined in this Section. However, as stated in Theorem 3.2, generating STX_{opt} is NP-complete.

Definition 3.1. A sequence of spanning trees $STX_n \subseteq (ST_1, ST_2, \ldots, ST_n)$ in graph G is called STX_{max} or *cost maximal* if (i) $Cost(STX_n) \leq C_{max}$, and (ii) $Cost(STX_n \cup ST_j) > C_{max}$ for any $ST_j \in ST_G$ and $ST_j \notin STX_n$.

Definition 3.2. A sequence of cost maximal spanning trees $STX_{max} \subseteq ST_G$ is called STX_{opt} or *cost optimal* if $Rel_{all}(STX_{max})$ is the maximum among all other possible cost

maximal sets of spanning trees. Thus, the optimal topology G_{opt} is the induced graph of STX_{opt}.

For the CN in Fig. 2.1 with n=8 and C_{max}=18, there are five different STX_{max}, one of which is STX_{opt}=(ST_1, ST_2, ST_3, ST_5) with a reliability of 0.874 that is the highest among other STX_{max}. Note that the reliability of each other four STX_{max} is Rel_{all}((ST_1, ST_4, ST_6))=0.567, Rel_{all}((ST_2, ST_7, ST_8))=0.841, Rel_{all}((ST_3, ST_6, ST_7))=0.749, and Rel_{all}((ST_4, ST_5, ST_8))=0.749, each of which is less than 0.874. Further, sequence (ST_1, ST_2) that contains the same links as (ST_1, ST_2, ST_3, ST_5), and thus forms the same topology, is not an STX_{max}. In addition, Rel_{all}((ST_1, ST_2))=0.777<Rel_{all}((ST_1, ST_2, ST_3, ST_5)) because Rel_{all}((ST_1, ST_2, ST_3, ST_5))=Rel_{all}((ST_1, ST_2))+$\Re_3$+$\Re_5$=0.777+0.0486+0.0486=0.874.

In general, one may obtain more than one possible STX_{opt} from a set of possible STX_{max}. Further, DPR/C-1 will generate a G_{opt} when it is given as input any STX_{opt}. The following theorem states this fact.

Theorem 3.1. The DPR/C-1 produces an optimal network topology G_{opt} for a given STX_{opt}.

Proof. Without loss of generality, let STX_{opt}=(ST_1, ST_2, …, ST_β), for β=|STX_{opt}|≤n. By definition, we claim that DPR/C-1 will always generate a topology G at DP[β, C_{max}] with X[β, C_{max}]=STX_{opt} and R[β, C_{max}]=Rel_{all}(STX_{opt}), and thus G=G_{opt}. Specifically, DPR/C-1 at column c=C_{max} first uses Eq. (3.5) to generate X[1, C_{max}]={ST_1} and R[1, C_{max}]=Rel_{all}(ST_1). Then, for that column, at row 2, it uses Eq. (3.8) to generate X[2, C_{max}]=(ST_1, ST_2) and R[2, C_{max}]=Rel_{all}((ST_1, ST_2)) because Rel_{all}(ST_1, ST_2)≥Rel_{all}(ST_1) and Cost(ST_1, ST_2)≤Cost(STX_{opt})≤C_{max}, and therefore ST_2 will always be included. Note that Rel_{all}((ST_1, ST_2))=R[1, C_{max}]+$\Re_2$, where $\Re_2$>0.

Similarly, DPR/C-1 will generate X[3, C_{max}]=(ST_1, ST_2, ST_3) and R[3, C_{max}]=Rel_{all}((ST_1, ST_2, ST_3)), …, X[β, C_{max}]=(ST_1, ST_2, ST_3, …, ST_β) and R[β, C_{max}]=Rel_{all}((ST_1, ST_2, ST_3, …, ST_β)). Notice that Eq. (3.8) computes R[i, C_{max}]=Rel_{all}(X[i-1, C_{max}] $\cup$ ST_i) because R[i-1, C_{max}]>Rel_{all}(X[i-1, j]) for any $j<C_{max}$, and Cost(X[i-1, C_{max}] $\cup$ ST_i)$\leq C_{max}$. In other words, X[i, C_{max}] is always generated from X[i-1, C_{max}] by including ST_i, and R[i, C_{max}] is computed from R[i-1, C_{max}] plus the reliability contribution of including ST_i in the topology, *i.e.*, R[i, C_{max}]=R[i-1, C_{max}]+$\Re_i$. Thus, DPR/C-1 will always generate an optimal topology G_{opt} with X[β, C_{max}]=(ST_1, ST_2, ST_3, …, ST_β) and R[β, C_{max}]=Rel_{all}((ST_1, ST_2, ST_3, …, ST_β)). **Q.E.D.**

Theorem 3.2. Generating STX_{max} and STX_{opt} of a general graph G is NP-Complete.

Proof. For STX_{max}, we prove the theorem by reduction from the subset-sum problem [26]. We consider a graph in which, for each pair of spanning trees, $ST_i \cap ST_j=\varnothing$ such that Cost(ST_i, ST_j)=Cost(ST_i) $\cup$ Cost(ST_j). Then, for the reduced problem, we set the target sum to C_{max}, and each integer item to Cost(ST_i). Because each STX_{opt} is a STX_{max}, one can directly conclude that generating STX_{opt} is also NP complete. **Q.E.D**.

In Section 3.3.4, we propose a method to generate the spanning trees in increasing weight and lexicographic order to optimize the performance of our dynamic programming approach. Further, in Section 3.3.5, we describe a more efficient algorithm, called DPR/C-2, which uses only $k\leq n$ of the ordered spanning trees.

3.3.4 Improving the Efficiency of DPR/C-1

Preprocessing spanning trees can improve the performance of our DPA in two ways. First, as stated in Theorem 1, sorting them such that they form STX_{opt} would make the approach able to generate optimal topology. This section describes five of the ten order criteria in Section 2.6.2 to heuristically generate effective ordered spanning trees. For each order criterion, OC1, OC2, OC3, OC4, or OC5, we compute link weight w_j for each $e_j \in E$, calculated as: $w_j=c_j/r_j$, $w_j=c_j$, $w_j=-(\log r_j)$, $w_j=-(\log r_j)+c_j$ and $w_j=-(\log r_j)\times c_j$, respectively. Then, one can use Prim's algorithm [26] to generate all spanning trees in increasing weight order. Recall that the weight of a spanning tree is calculated as the summation of the weight of each link in the spanning tree. Using the OCs, we obtain the following orders for the spanning trees in Table 2.1; OC1: (ST_7,ST_2,ST_8,ST_6,ST_3,ST_1,ST_5,ST_4), OC2: (ST_6,ST_7,ST_4,ST_3,ST_8,ST_1,ST_5,ST_2), OC3: (ST_6,ST_4,ST_3,ST_1,ST_5,ST_7,ST_8,ST_2), OC4: (ST_6,ST_7,ST_3,ST_3,ST_2,ST_1,ST_5,ST_8) and OC5: (ST_2,ST_7,ST_8,ST_3,ST_6,ST_1,ST_5,ST_4). Note that the five order criteria can also be used to sort (*s*, *t*) paths; see Appendix A.

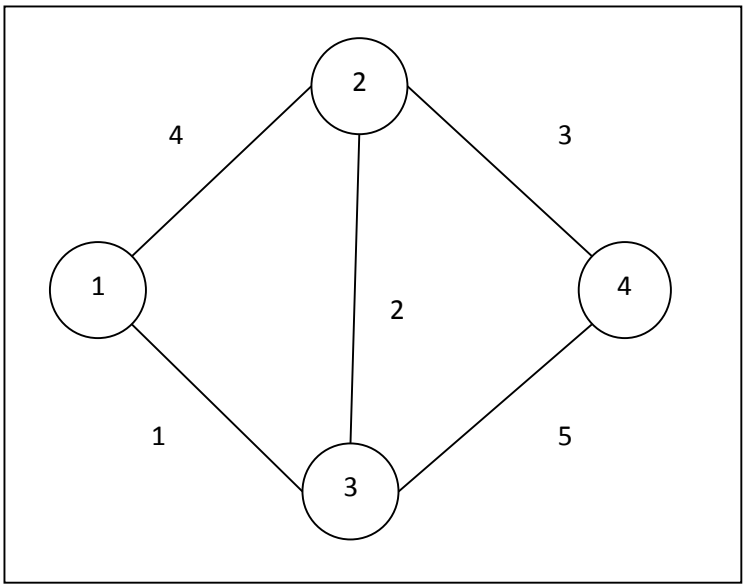


Fig. 3.2: A CN of Fig. 2.1 with new labels for OC2

Table 3.2: Link weight and spanning tree set for CN in Fig. 3.2

i	ST_G			Link Weight		
	ST_i	$Rel_{all}(ST_i)$	$Cost(ST_i)$	e_j	c_j	r_j
1	{1, 2, 3}	0.378	9	1	2	0.7
2	{1, 2, 5}	0.378	11	2	3	0.6
3	{1, 3, 4}	0.567	11	3	4	0.9
4	{1, 4, 5}	0.567	13	4	5	0.9
5	{2, 3, 4}	0.486	13	5	6	0.9
6	{2, 3, 5}	0.486	13			
7	{2, 4, 5}	0.486	14			
8	{3, 4, 5}	0.729	15			

Second, sorting the spanning trees in increasing lexicographic order allows the SDP technique used to calculate reliability, *e.g.*, CAREL [22], generate only one equivalent disjoint term for each spanning tree, and hence minimize the technique's computational complexity [25]. We propose using the following two steps to generate the spanning trees of a network G in increasing weight and lexicographic order. First, we label the link with the least cost with the smallest lexicographic alphabet (1, 2, …, |E|), *i.e.*, 1, then the second least cost with the second alphabet, *etc.*; links with the same costs are labeled with their consecutive alphabets randomly. For example, for OC2, this step converts the CN in Fig. 2.1 into Fig. 3.2. Then, we use a modified Prim's algorithm [26] to generate all spanning trees sorted in their increasing weights. Table 3.2 shows the generated spanning trees in increasing order of lexicographic as well as weight for OC2. Because a modified Prim's algorithm requires a time complexity of $O(|E|\times\log|V|)$, the DPR/C-1 algorithm requires an extra $O(n\times(|E|\times\log|V|))$ time complexity for the improvement, *i.e.*, $O(\Omega\times|E|\times T_{max}+n\times|E|\times C_{max}+n\times(|E|\times\log|V|))$.

3.3.5 DPR/C Version 2

Like the algorithms in [16], [17], [19], our DPR/C-1 requires all spanning trees of the CN to generate the best topology, and is not feasible for CN that contains a large number of spanning trees. In this section, we propose the second version of DPR/C

algorithm, called DPR/C-2, that uses only the first k least weight spanning trees that can be generated using the method described in Section 3.3.4.

Our DPR/C-2 sets the value of k dynamically as follows. Consider that DPR/C-2 has generated and used spanning trees ST_1, ST_2, …, ST_{i-1}, ST_i to obtain a network topology STX_i with reliability R_i and cost $C_i \leq C_{max}$, and it generates ST_{i+1} to obtain a STX_{i+1} with reliability R_{i+1} and cost C_{i+1}. DPR/C-2 considers two cases: (i) $C_{max} \geq C_{i+1} > C_i$, thus $STX_{i+1} \neq STX_i$ and $R_{i+1} > R_i$; and (ii) $C_{i+1} = C_i$, $STX_{i+1} = STX_i$, and $R_{i+1} > R_i$, *i.e.*, links in ST_{i+1} have already been included in G_i. If each generated ST_{i+1} improves the reliability while changing the topology within budget, *i.e.*, case (i), DPR/C-2 keeps generating the subsequent ST_{i+1}. However, for case (ii), DPR/C-2 would check if the reliability improvement is significant; in this chapter, we set $R_{i+1}/R_i \geq 1.005$ as the criterion, *i.e.*, 0.5% improvement. If it is significant, DPR/C-2 continues with the next spanning tree. Otherwise, if each of the next 10 consecutive spanning trees does not significantly improve reliability, DPR/C-2 stops considering the subsequence spanning trees, *i.e.*, ST_{i+1}, ST_{i+2}, …, ST_n, and returns with the topology formed from $STX_{max} \subseteq (ST_1, ST_2, \ldots, ST_{i=k})$. Notice that setting a larger value for k does not necessarily make DPR/C-2 produce better result due to the heuristic nature of the spanning tree order, unless (ST_1, ST_2, …, ST_k) is a STX_{opt}.

DPR/C-2 reduces the time complexity of DPR/C-1 because it generates and uses only $k \leq n$ spanning trees; *e.g.*, DPR/C-2 needs to generate and use only spanning trees ST_1, ST_2, ST_3, and ST_5 in Table 2.1. Further, because the k spanning trees are generated in increasing lexicographic order, the SDP approach, *i.e.*, CAREL [22], which is used in DPR/C-2, would generate only k equivalent disjoint terms. Therefore DPR/C-2 reduces the time complexity of DPR/C-1 to $O(\Omega \times |E| \times k$

$+k\times|E|\times C_{max}+k\times(|E|\times\log|V|))=O(k\times|E|\times C_{max})$; since Ω is the total number of cases where $Cost(\bullet)\leq C_{max}$, when $\Omega\leq C_{max}$. Our simulation in Section 3.4.3 shows that DPR/C-2 requires only a small *k* when generating topologies for networks with up to 1.899^{102} spanning trees, and therefore we make a conjecture that DPR/C-2 has a pseudo polynomial time complexity when the *k* spanning trees are sorted in increasing lexicographic order.

3.4 Evaluation

First, in Section 3.4.1, we use DPR/C-1 on the 20 networks, described in Section 2.6.1, to observe the effects of using the five order criteria, described in Section 3.3.4, and lexicographic order on its effectiveness and efficiency respectively. Second, in Section 3.4.2, we use both DPR/C-1 and DPR/C-2, to generate the network topology with known optimal topology to gauge DPR/C's effectiveness. Finally, we use five grid networks in Section 2.6.1 that contain up to 1.899^{102} spanning trees and 2^{99} (*s*, *t*) paths to show the efficiency and effectiveness of DPR/C-2. Note that we do not compare the performances of DPR/C with the existing approaches in [16], [17], [19] because they are feasible only for small networks.

3.4.1 The Effect of Input Orders on DPR/C Performance

This section describes the effects of using five order criteria, OC1 to OC5, and lexicographic order on the effectiveness and efficiency of DPR/C respectively. Section 3.4.1.1 shows the effects when using DPR/C with Rel_{all} measure while Section 3.4.1.2 describes the results for Rel_2 metric.

3.4.1.1 All-Terminal Reliability Measure

For each of the 20 network topologies in Table 2.3, we use a modified Prim's algorithm [26] to generate spanning trees sorted based on OC1 to OC5 order criteria. Table 3.3 shows that DPR/C-1 using each order criterion produces at least 2 of the 20 best results (in bold) as compared to none using random order. Thus, the orders improve the effectiveness of DPR/C-1. The table shows that OC2 is the best performer, producing 13 of 20 best results, followed by OC1 and OC5 with 9 and 8 respectively; OC4 and OC3 are the worst with 5 and 2 respectively. Note that OC2 produces the best results because it orders spanning trees in increasing cost weight, matching the constraint parameter of NTD-R/C, which enable DPR/C-1 to use more spanning trees when generating topology, thus heuristically increase the $Rel_{all}(G)$.

Table 3.3: The effects of spanning tree orders on the performance of DPR/C-1

Input			**$Rel_{all}(G_n)$ for each OC criterion**				
CN	***C_{max}***	**Random**	**OC1**	**OC2**	**OC3**	**OC4**	**OC5**
$CN_8^{4,5}$	18	0.3492	0.6048	**0.7603**	0.6048	0.6264	0.6632
$CN_{21}^{5,8}$	24	0.5621	0.8034	0.8237	0.7231	**0.8841**	0.8034
$CN_{21}^{6,8}$	20	0.3599	0.5645	0.5339	**0.5686**	0.4037	**0.5686**
$CN_{55}^{6,9}$	25	0.4729	**0.7388**	0.7232	0.5784	**0.7388**	0.5784
$CN_{368}^{7,12}$	33	0.576	**0.9080**	**0.9080**	0.6284	0.6432	**0.9080**
$CN_{1033}^{7,15}$	38	0.7281	**0.9229**	**0.9229**	0.7998	0.8985	0.7998
$CN_{247}^{8,12}$	35	0.2978	**0.7113**	**0.7113**	0.4141	0.5617	0.4141
$CN_{256}^{8,12}$	35	0.3511	0.5972	**0.7591**	0.5972	0.5648	**0.7591**
$CN_{576}^{8,13}$	35	0.5757	0.4994	**0.8362**	0.4994	0.5085	0.4994
$CN_{171}^{9,12}$	33	0.1667	0.4073	**0.5845**	0.4073	0.5533	**0.5845**
$CN_{327}^{9,13}$	35	0.2707	**0.6167**	0.5820	0.4477	**0.6167**	0.4477
$CN_{647}^{9,14}$	35	0.4530	0.7889	0.5207	0.7889	**0.8662**	**0.8662**
$CN_{2112}^{10,21}$	55	0.259	**0.5723**	**0.5723**	0.311	0.5438	**0.5723**
$CN_{1598}^{11,21}$	56	0.3738	0.3197	0.5761	0.3197	**0.5961**	0.3197
$CN_{3666}^{13,22}$	60	0.2272	0.3112	**0.4205**	0.3112	0.4173	**0.4205**
$CN_{7683}^{16,30}$	80	0.4714	**0.5872**	**0.5872**	0.3686	0.3507	0.3686
$CN_{9471}^{17,25}$	65	0.6154	0.5129	**0.8725**	0.5129	0.5201	0.6154
$CN_{21456}^{18,27}$	70	0.4847	**0.7274**	**0.7274**	0.4131	0.4096	0.4131
$CN_{24173}^{20,30}$	80	0.2891	0.2132	**0.4714**	0.2132	0.2946	0.2132
$CN_{18257}^{21,26}$	70	0.7204	**0.901**	0.8497	**0.901**	0.508	**0.901**
# of best results		0	9	**13**	2	5	8

To see the effect of using lexicographic order on the efficiency of DPR/C, we use OC2 and the relabelling method described in Section 3.3.4 to generate spanning trees in increasing order of lexicographic. Table 3.4 shows that DPR/C-1 using the order produces topologies with Rel_{all} between 1.18 to 3.5 times of those produced using random ordered trees. The table also shows that the SDP approach [22] used in DPR/C-1 generates fewer disjoint terms (#DT) as compared to the random order.

Table 3.4: The effects of OC2 and lexicographic order on DPR/C-1 performance

Input		**Random**			**OC2 and lexicographic order**		
CN	C_{max}	**$Rel_{all}(STX_n)$**	**#DT**	**CPU time (seconds)**	**$Rel_{all}(STX_n)$**	**#DT**	**CPU time (seconds)**
$CN_{8}^{4,5}$	18	0.3492	10	0.01	0.7603	6	0.01
$CN_{21}^{5,8}$	24	0.5621	17	1.10	0.8237	8	0.99
$CN_{21}^{6,8}$	20	0.3599	14	0.93	0.5339	9	0.72
$CN_{55}^{6,9}$	25	0.4729	29	1.08	0.7232	16	1.07
$CN_{368}^{7,12}$	33	0.576	71	7.01	0.908	24	6.74
$CN_{1033}^{7,15}$	38	0.7281	178	17.83	0.9229	66	14.01
$CN_{247}^{8,12}$	35	0.2978	47	5.51	0.7113	31	4.40
$CN_{256}^{8,12}$	35	0.3511	69	5.58	0.7591	33	5.31
$CN_{576}^{8,13}$	35	0.5757	86	13.19	0.8362	37	12.72
$CN_{171}^{9,12}$	33	0.1667	43	4.26	0.5845	22	3.19
$CN_{327}^{9,13}$	35	0.2707	94	6.44	0.582	29	5.26
$CN_{647}^{9,14}$	35	0.453	148	15.95	0.5207	56	14.61
$CN_{2112}^{10,21}$	55	0.259	278	28.08	0.5723	41	22.04
$CN_{1598}^{11,21}$	56	0.3738	204	21.16	0.5761	54	20.02
$CN_{3666}^{13,22}$	60	0.2272	126	43.37	0.4205	71	42.20
$CN_{7683}^{16,30}$	80	0.4714	213	56.48	0.5872	49	51.39
$CN_{9471}^{17,25}$	65	0.6154	381	78.94	0.8725	107	73.98
$CN_{21456}^{18,27}$	70	0.4847	406	182.27	0.7274	187	151.40
$CN_{24173}^{20,30}$	80	0.2891	541	194.54	0.4714	151	185.75
$CN_{18257}^{21,26}$	70	0.7204	351	148.49	0.8497	142	143.73

Specifically, DPR/C-1 generates, for each spanning tree, exactly one DT using the ordered trees as compared to an average of three DTs using random ordered trees, and hence the order speeds up its running time, *i.e.*, 16.93% faster as compared to using random order. These results show the benefits of our OC2 and lexicographic order method. Note that each #DT in Table 3.4 shows the number of DTs generated for the

selected spanning trees in each X[n, C_{max}]; *e.g.*, for $CN_{327}^{9,13}$, X[n, 35] contains 33 and 29 spanning trees for random and pre-ordered that result in 94 DTs and 29 DTs, respectively.

3.4.1.2 Two-Terminal Reliability Measure

We use Yen's algorithm [46] to generate all (s, t) paths sorted following each of the five order criteria, *i.e.*, OC1 to OC5, and use them as inputs of DPR/C-1.

Table 3.5: The effects of path orders on the performance of DPR/C-1

Input		**$Rel_2(G_m)$ for each each OC criterion**					
CN	***C_{max}***	**Random**	**OC1**	**OC2**	**OC3**	**OC4**	**OC5**
$CN_4^{4,5}$	18	0.866	**0.912**	0.866	0.866	0.866	**0.912**
$CN_9^{5,8}$	20	**0.914**	**0.914**	**0.914**	**0.914**	**0.914**	**0.914**
$CN_7^{6,8}$	20	0.823	**0.833**	**0.833**	**0.833**	**0.833**	**0.833**
$CN_{13}^{6,9}$	20	0.781	**0.793**	0.781	**0.793**	**0.793**	**0.793**
$CN_{25}^{7,12}$	20	**0.939**	**0.939**	**0.939**	**0.939**	**0.939**	**0.939**
$CN_{14}^{7,15}$	20	**0.947**	**0.947**	0.941	**0.947**	**0.947**	**0.947**
$CN_{20}^{8,12}$	20	**0.853**	**0.853**	0.843	**0.853**	**0.853**	**0.853**
$CN_{24}^{8,12}$	20	0.686	0.702	0.656	0.702	**0.778**	0.702
$CN_{29}^{8,13}$	20	**0.777**	**0.777**	0.761	**0.777**	**0.777**	**0.777**
$CN_{13}^{9,12}$	20	**0.591**	**0.591**	**0.591**	**0.591**	**0.591**	**0.591**
$CN_{18}^{9,13}$	20	**0.691**	**0.691**	**0.691**	**0.691**	**0.691**	**0.691**
$CN_{44}^{9,14}$	40	0.901	**0.908**	0.906	0.905	0.907	0.907
$CN_{64}^{10,21}$	40	0.963	**0.964**	0.961	0.962	**0.964**	0.963
$CN_{18}^{11,21}$	20	**0.921**	**0.921**	**0.921**	**0.921**	**0.921**	**0.921**
$CN_{281}^{13,22}$	40	0.961	0.957	**0.966**	0.957	0.957	0.957
$CN_{36}^{16,30}$	20	**0.882**	**0.882**	**0.882**	**0.882**	**0.882**	**0.882**
$CN_{136}^{17,25}$	40	0.937	0.942	**0.947**	0.942	0.942	0.942
$CN_{281}^{18,27}$	40	0.879	**0.902**	0.892	0.892	0.889	**0.902**
$CN_{780}^{20,30}$	40	0.838	0.842	0.881	**0.901**	0.842	0.88
$CN_{44}^{21,26}$	40	0.682	0.678	**0.688**	0.678	0.678	0.678
# of best results		9	**15**	10	12	13	13

As shown in Table 3.5, DPR/C-1 using each order criterion produces between 10 and 15 of 20 the best results (in bold) as compared to only 9 using the random order. Thus, the order criteria help DPR/C-1 in producing better results. The table shows that OC1

is the best performer, followed by OC4 and OC5 with 15, 13, and 13 best results respectively; OC2 and OC3 are the worst.

To see the effect of lexicographic order on the computation time of DPR/C-1, we use OC1 and the relabelling method described in Section 3.3.4 to generate (*s, t*) paths in increasing lexicographic order. As shown in Table 3.6, DPR/C-1 generates fewer disjoint terms (#DT) using the order as compared to the random order, and hence the order speeds up its running time. Note that each #DT in Table 3.6 shows the number of DTs generated for the selected paths in each X[m, C_{max}]; *e.g.*, for $CN_{780}^{20,30}$, X[m, 40] contains 69 and 52 paths for random and pre-ordered that result in 148 DTs and 81 DTs, respectively. DPR/C-1 produces results up to 13.84% faster as compared to using random order. Further, the criterion allows DPR/C-1 to produce topologies with reliability up to 2.62% higher as compared to using the random order; see the results for $CN_{281}^{13,27}$. In the rest of the thesis, we will use the lexicographic order for each algorithm that utilizes SDP approach. The results in Table 3.3 and 3.5 show that OC2 helps DPR/C-1 produce the most best results for $Rel_{all}(G)$ but the least for $Rel_2(G)$.

The results are due to the heuristic nature of the criterion. On the other hand, OC1 that generates the most best results for $Rel_2(G)$ also produces the second most best results for $Rel_{all}(G)$. Notice that OC1 generates each link weight from the ratio between link reliability and its cost, and thus each link weight provides a good balance between both factors, using which DPR/C is able to maximize the reliability objective that meets the cost constraint.

Table 3.6: The effects of OC1 and lexicographic order on DPR/C-1 performance

Input		Random			OC1 and lexicographic order		
CN	C_{max}	$Rel_2(PX_m)$	**#DT**	**CPU time (seconds)**	$Rel_2(PX_m)$	**#DT**	**CPU time (seconds)**
$CN_4^{4,5}$	18	0.866	3	0.021	0.912	2	0.018
$CN_9^{5,8}$	20	0.914	11	0.037	0.914	7	0.030
$CN_7^{6,8}$	20	0.823	8	0.032	0.833	5	0.028
$CN_{13}^{6,9}$	20	0.781	12	0.041	0.793	9	0.033
$CN_{25}^{7,12}$	20	0.939	16	0.052	0.939	13	0.046
$CN_{14}^{7,15}$	20	0.947	13	0.043	0.947	10	0.034
$CN_{20}^{8,12}$	20	0.853	17	0.066	0.853	12	0.043
$CN_{24}^{8,12}$	20	0.686	14	0.058	0.702	10	0.033
$CN_{29}^{8,13}$	20	0.777	22	0.077	0.777	15	0.058
$CN_{13}^{9,12}$	20	0.591	10	0.035	0.591	6	0.030
$CN_{18}^{9,13}$	20	0.691	14	0.053	0.691	11	0.038
$CN_{44}^{9,14}$	40	0.901	27	0.109	0.908	14	0.091
$CN_{64}^{10,21}$	40	0.963	56	0.337	0.964	31	0.312
$CN_{18}^{11,21}$	20	0.921	17	0.057	0.921	12	0.036
$CN_{281}^{13,22}$	40	0.961	91	1.653	0.957	42	1.487
$CN_{36}^{16,30}$	20	0.882	28	0.082	0.882	16	0.064
$CN_{136}^{17,25}$	40	0.937	85	1.360	0.942	38	1.274
$CN_{281}^{18,27}$	40	0.879	103	2.095	0.902	67	1.854
$CN_{780}^{20,30}$	40	0.838	148	2.825	0.842	81	2.434
$CN_{44}^{21,26}$	40	0.682	32	0.097	0.678	18	0.076

3.4.2 Performance on Benchmark Networks

Since we are unable to find any benchmark network with known optimal topology for NTD-R/C problem, we generate 100 networks from the 20 topologies to benchmark the optimality of DPR/C with Rel_{all} measure as follows.

Consider a graph G=(V, E) with its cost constraint C_{max}. The BF-1, described in Section 3.1, requires generating $\binom{|E|}{1} + \binom{|E|}{2} + \ldots \binom{|E|}{\alpha} + \ldots + \binom{|E|}{|E|-1}$ possible subgraphs G_i from G each of which contains STX_i, and among those that have $Cost(STX_i) \leq C_{max}$, BF-1 selects one with the maximum $Rel_{all}(STX_i)$ as the optimal topology. Therefore, for networks with large |E|, it is impractical to use BF-1 to generate optimal topologies. As an alternative, we have used BF-1 to generate optimal

topology only among $\binom{|E|}{\alpha}$ possible G_i's, each of which is generated by removing a different possible α links from G=(V, E). In particular, for α=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, we first generated $\binom{|E|}{1}$ numbers of G_i. For each G_i, we then calculate $Rel_{all}(STX_i)$, and select G_i with the maximum reliability as the optimal G_{opt} with cost $Cost(STX_{opt})$. Consequently, an optimal solution for G=(V, E) with cost constraint C_{max}=$Cost(STX_{opt})$ would obtain G_{opt} as its output. Note that the C_{max} used for each generated benchmark is the tightest possible constraint, and thus each benchmark evaluates the worst possible performance for DPR/C algorithm. However, for networks with large |E|, $\binom{|E|}{\alpha}$ is also prohibitively large, and thus selecting the maximum reliability value might be infeasible for α>5. For example, for |E|=30 and α=6, there are 593775 different G_i and thus finding G_{opt} among them require significant amount of time since computing the reliability value of each G_i takes exponential time. Therefore, we consider only α=1, 2, 3, 4, 5 for each of the 20 networks in Table 2.3, generating in total 100 benchmark networks.

We use the same method to generate another set of 100 networks from the 20 topologies to benchmark the optimality of DPR/C with Rel_2 measure.

3.4.2.1 Performance for All-Terminal Reliability

We use both versions of DPR/C to generate topologies from the 100 benchmark networks. For each network, we sort the spanning trees following OC2 criterion. Our simulations show that both methods produce the same results, which are optimal 85% of the time; Table 3.7 shows only their 15 non-optimal results, to save space. While performing as well as DPR/C-1, DPR/C-2 requires significantly fewer spanning trees

(column *k*) to produce its results, especially for larger networks (*e.g.*, 671 versus 24173). Table 3.7 shows that DPR/C-2 uses between 2.1% to 39.8% of the number of spanning trees used in DPR/C-1. Note that, as described in Section 3.3.5, DPR/C-2 stops its computation after generating at least 10 consecutive insignificant improvements; each of the reported values of *k* excludes the unnecessary additional spanning trees. The CPU time column shows the running time of DPR/C-2.

Table 3.7: The non-optimal results produced by DPR/C using OC2 order

Input			***k***	**DPR/C-1 and DPR/C-2 results**		
CN	α	C_{max}		**Rel$_{all}$(STX$_k$)**	**Cost(STX$_k$)**	**CPU time (seconds)**
$CN_{55}^{6,9}$	2	23	12(21%)	0.7109(-0.92%)	22(-4.35%)	0.05(4.7%)
$CN_{247}^{8,12}$	3	30	93(37.6%)	0.5049(-0.2%)	30(0.00%)	2.27(36.67%)
$CN_{256}^{8,12}$	5	26	102(39.8%)	0.5285(-2.52%)	25(-3.85%)	2.31(52.98%)
$CN_{576}^{8,13}$	1	39	141(24.4%)	0.7050(-2.76%)	36(-3.80%)	3.49(25.95%)
$CN_{327}^{9,13}$	3	31	121(37%)	0.3966(-1.34%)	28(-9.68%)	2.64(41.71%)
$CN_{327}^{9,13}$	5	23	112(34.2%)	0.3287(-0.60%)	23(0.00%)	2.38(38.57%)
$CN_{647}^{9,14}$	3	57	147(22.7%)	0.7220(-3.46%)	55(-3.51%)	3.94(27.52%)
$CN_{2112}^{10,21}$	4	72	481(15.4%)	0.5970(-3.48%)	70(-2.78%)	13.68(47.06%)
$CN_{1598}^{11,21}$	4	53	483(30.2%)	0.5940(-2.73%)	51(-3.77%)	12.94(58.66%)
$CN_{1598}^{11,21}$	5	51	483(30.2%)	0.5015(-2.96%)	51(0.00%)	12.93(59.18%)
$CN_{3666}^{13,22}$	3	84	567(34%)	0.3452(-3.4%)	81(-3.57%)	14.21(30.07%)
$CN_{9471}^{17,25}$	5	63	622(6.56%)	0.4570(-1.08%)	62(-1.59%)	15.16 (20.33%)
$CN_{21456}^{18,27}$	2	63	652(3.03%)	0.4205(-1.73%)	60(-4.76%)	17.66(10.8%)
$CN_{24173}^{20,30}$	5	77	671(2.77%)	0.2480(-2.81%)	77(0.00%)	17.49(9.14%)
$CN_{18257}^{21,26}$	3	37	385(2.1%)	0.5206(-3.14%)	37(0.00%)	8.08(6.02%)

As shown in the table, DPR/C-2 requires no more than 17.66 CPU seconds to produce the topologies. In general, DPR/C-2 is faster than DPR/C-1; see the % value in brackets that shows the percentage of DPR/C-2's running time with respect to DPR/C-1's time. These results show the benefits of using DPR/C-2 than DPR/C-1, especially for use in large networks. DPR/C-2 and DPR/C-1 produces NT with Rel$_{all}$ no worse than 3.48% off optimal, respectively; see the % value in bracket next to each reliability value.

Interestingly, most of the non-optimal results have up to 9.68% lower cost as compared to those for optimal; see the % value in bracket for each cost.

3.4.2.2 Performance for Two-Terminal Reliability

We sort the (*s*, *t*) paths of the 100 benchmark networks following OC1, OC4 and OC5 criteria; we didn't consider OC2 and OC3 because, as shown in Table 3.5, they are the worst performers. Then, we use both DPR/C-1 and DPR/C-2 with each of the 300 sets of sorted (*s*, *t*) paths as input. Our simulations show that DPR/C-1 and DPR/C-2 with OC1, OC4 and OC5 produce 89%, 88%, 87% optimal results respectively. Interestingly, of the 11 non-optimal results for OC1, DPR/C-1 and DPR/C-2 produce network with reliability no worse than 1.43% off from optimal; see the % value in bracket next to each value in $Rel_2(PX_k)$ column in Table 3.8. Further, most of the non-optimal results have up to 4.1% lower cost as compared to that of optimal; see the % value in bracket next to each cost in $Cost(PX_k)$ column. Notice that OC4 and/or OC5 produce optimal results for 3 of the 11 non-optimal results for OC1, and therefore DPR/C-1 and DPR/C-2 could produce up to 92% optimal results with only 8 non optimal results if they used all OC1, OC4 and OC5 and select the best among their results; Table 3.8 shows their 8 non-optimal results. Note that DPR/C-2 uses between 47 and 85 of 780 paths to generate the results, *i.e.*, only 6% to 11% of the total paths and up to 1.26 CPU seconds while producing the same results as DPR/C-2; their non-optimal results are only up to 1.43% off from optimal.

Table 3.8: The non-optimal results produced by DPR/C with OC1, OC4 & OC5

Input			*k*	DPR/C-1 and DPR/C-2 results		
CN	α	C_{max}		**Rel$_2$(PX$_k$)**	**Cost(PX$_k$)**	**CPU time (seconds)**
$CN_{25}^{7,12}$	1	37	12	0.9759 (-0.24%)	34 (-4.1%)	0.12(2.36%)
$CN_{64}^{10,21}$	3	59	15	0.9757(-0.13%)	57(-3.4%)	0.06(3.54%)
$CN_{64}^{10,21}$	4	56	15	0.9745(-0.16%)	54(-3.5%)	0.04(12.87%)
$CN_{281}^{13,22}$	1	68	37	0.9809 (-0.21%)	68(0%)	0.22(10.08%)
$CN_{281}^{13,22}$	5	55	30	0.935 (-0.29%)	54(-1.8%)	0.15(23.79%)
$CN_{136}^{17,25}$	2	75	43	0.9788 (-0.25%)	73(-2.7%)	0.41(17.63%)
$CN_{780}^{20,30}$	3	86	47	0.9730 (-0.60%)	85(-1.2%)	0.65(18.32%)
$CN_{780}^{20,30}$	4	82	85	0.9713 (-1.43%)	80(-2.4%)	1.26(35.76%)

3.4.2 DPR/C-2 on Grid Networks

This section evaluates the performance of DPR/C-2 on five grid networks, *i.e.*, $G_{6\times6}$, $G_{3\times12}$, $G_{3\times16}$, $G_{2\times20}$, $G_{2\times100}$, described in Table 2.5. Recall that the grid networks contain 36 to 200 nodes, 57 to 298 links, 3.557^{18} to 1.899^{102} spanning trees and 538020 to 2^{99} (*s*, *t*) paths; see Section 2.6.1. As shown in Table 3.9 (see k_{max}), DPR/C-2 with OC2 uses only $1368/1.899^{102}=5.3^{-26}$ to $742/2^{20}=7.07^{-4}$ fractions of the spanning trees contained in the networks. Similarly, Table 3.10 shows that DPR/C-2 with OC1, OC4 and OC5 produces each topology using only $216/524288=4.1\times10^{-4}$ to $1000/2^{99}=1.5^{-27}$ fractions of the (*s*, *t*) paths contained in the networks. DPR/C-2 is also very efficient in producing the results for the networks with the other C_{max} values, which show the practicality of our technique; see the *k* and CPU time columns. Specifically, DPR/C-2 requires no more than 26.32 (22.64) CPU seconds to produce results for grid networks that contain up to 1.899^{102} spanning trees (2^{99} (*s*, *t*) paths); see Table 3.9 (Table 3.10). Note that for each network, we consider four different C_{max} values, *i.e.*, 50%, 75%, 90%, and 99% of the total cost of the network, *e.g.*, 29, 43, 51, and 56 for $CN_{\geq3.557^{18}}^{36,57}$ in Table 3.9, respectively.

Table 3.9: Performance of DPR/C-2 with spanning trees for Grids

CN	C_{max}	***k***	$Rel_{all}(G_k)$	**CPU time (seconds)**
$CN^{36,57}_{\geq 3.557^{18}}$ **$Grid_{3x12}$ k_{max}=247 Rel_{max}=0.9130**	29	38	0.5248	4.27
	43	147	0.6109	4.45
	51	189	0.7869	4.76
	56	247	**0.9121 (-0.4%)**	4.86
$CN^{36,60}_{\geq 3.557^{18}}$ **$Grid_{6x6}$ k_{max}=306 Rel_{max}=0.9173**	30	123	0.5371	6.41
	45	209	0.6402	6.53
	54	286	0.7819	6.86
	59	306	**0.8922 (-0.2%)**	7.05
$CN^{40,58}_{2^{20}}$ **$Grid_{2x20}$ k_{max}=742 Rel_{max}=0.74530**	29	312	0.6362	11.92
	44	486	0.6867	12.28
	52	502	0.7089	12.40
	57	512	**0.7351 (-2.6%)**	12.35
$CN^{48,77}_{\geq 2.5^{24}}$ **$Grid_{3x16}$ k_{max}=512 Rel_{max}=0.7218**	39	549	0.6183	17.18
	58	587	0.6821	17.45
	69	712	0.7037	18.21
	76	742	**0.7216 (-0.9%)**	18.66
$CN^{200,298}_{1.899^{102}}$ **$Grid_{2x100}$ k_{max}=1368 Rel_{max}=0.25107**	149	879	0.0951	22.03
	224	944	0.1742	25.44
	268	1015	0.1963	26.07
	295	1368	**0.2273 (-0.8%)**	26.32

Table 3.9 and Table 3.10 also show the reliability value of each generated topology. Because DPR/C-2 does not calculate the exact reliability values of its generated topologies, we have used MCS [10] to compute estimated $Rel_{all}(G)$ and $Rel_2(G)$ for each generated topology; we used a sample size of 10^6 in the simulation. However, we are unable to gauge the optimality of the generated topologies for such large networks, except for C_{max}=0.99×C_{all}. In this case, for each network and Rel_{all} measure, we deleted one link from the network, and used MCS to generate its $Rel_{all}(G)$; we repeated the step |E|-1 times, and selected the maximum $Rel_{all}(G)$ as the best solution. Note that the solution may not be optimal because MCS does not always generate exact reliability. We used the same method for Rel_2 measure.

Table 3.10: Performance of DPR/C-2 with (*s*, *t*) paths for Grids

CN	C_{max}	*k*	$Rel_2(G_k)$	**CPU time (seconds)**
$CN_{1262818}^{36,57}$ $Grid_{3x12}$ k_{max}=358, Rel_{max}=0.9753	29	41	0.9401	3.38
	43	335	0.9717	5.16
	51	344	0.9736	5.32
	56	358	**0.9748(-0.05%)**	**5.77**
$CN_{538020}^{36,60}$ $Grid_{6x6}$ k_{max}=323, Rel_{max}=0.9613	30	184	0.6592	4.68
	45	278	0.8933	5.65
	54	280	0.9386	5.71
	59	329	**0.9585 (-0.3%)**	**6.54**
$CN_{524288}^{40,58}$ $Grid_{2x20}$ k_{max}=216, Rel_{max}=0.7841	29	58	0.2164	3.72
	44	207	0.4714	5.21
	54	210	0.7053	5.66
	57	215	**0.7661(-2.29%)**	**5.74**
$CN_{64019921}^{48,77}$ $Grid_{3x16}$ k_{max}=247, Rel_{max}=0.9562	39	127	0.8058	4.14
	58	148	0.8720	4.65
	69	178	0.9297	4.88
	76	243	**0.9552(-0.08%)**	**5.69**
$CN_{2^{99}}^{200,298}$ $Grid_{2x100}$ k_{max}=1000, Rel_{max}=0.3042	149	482	0.2320	10.33
	224	724	0.2640	14.78
	268	811	0.2874	19.02
	297	987	**0.3020(-0.65%)**	**22.64**

As shown in Table 3.9 and Table 3.10 (the numbers in bold in column $Rel_{all}(G_k)$ and $Rel_2(G_k)$, respectively), DPR/C-2 is able to generate a topology with Rel_{all} only up to 2.6% off from the best topology and Rel_2 up to 2.29% off from the best solution, respectively for C_{max}=0.99×C_{all}. Note that the reliability of each generated topology cannot be larger than its Rel_{max} calculated from the original networks.

3.5 Chapter Summary

We have addressed an NP-hard problem, called NTD-R/C, to design a network topology with maximum reliability subject to a cost constraint, 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. We have proposed a heuristic dynamic programming method, DPR/C, to solve NTD-R/C. The first version of DPR/C, called DPR/C-1, requires all spanning trees to maximize Rel_{all}. The chapter has shown that DPR/C-1 is optimal if the spanning trees are optimally ordered. It also

shows that generating optimal order of spanning trees is NP-complete, and therefore proposes five heuristic order criteria, OC1, OC2, OC3, OC4, and OC5. Further, it has shown that generating the spanning trees and (*s*, *t*) paths in increasing lexicographic order helps reducing the running time of SDP method [22] used to compute $Rel_{all}(G)$ and $Rel_2(G)$ respectively. The chapter has also described the second version of DPR/C, called DPR/C-2, that uses only *k* of the sorted spanning trees to improve the time efficiency of DPR/C-1 while producing similar results. Extensive simulations using hundreds of networks show the merits of using the sorting methods, and the effectiveness of DPR/C-1 and DPR/C-2. For DPR/C with Rel_{all} and Rel_2 measures, the simulations show that OC2 and OC1 are respectively the best criteria. Further, when the input is also ordered in increasing lexicographic order, DPR/C produces results up to 16.93% faster as compared to using random order.

The chapter also shows that DPR/C-2 is able to generate 85% optimal results, while using only a small number of *k* (2.1% through 39.8%) spanning trees (*e.g.*, 671 versus 24173), and up to 17.66 CPU seconds. Furthermore, the non-optimal results are only up to 3.4% off from optimal for the simulated examples. Further, DPR/C-2 is able to generate 92% optimal results, while using only a small number of *k* (*s*, *t*) paths (only 6% to 11% of all (*s*, *t*) paths in the network), and up to 1.26 CPU seconds, where the non-optimal results are only up to 1.43% off from optimal.

The following Chapter 4 will describe a closely related problem to NTD-R/C, called NTD-C/R, and propose DPC/R, another DPA, to solve the problem.

Chapter 4

Network Topology Design with Minimal Cost and Reliability Constraint

This chapter discusses NTD-C/R, an O_1/C_1 optimization problem, to construct network topology with minimum cost subject to a given reliability constraint. The solution to the problem is important in designing topology used for critical applications, *e.g.*, emergency response, rescue and military operations.

The chapter proposes a DP solution, called DPC/R with two versions to solve NTD-C/R given as input a sequence of spanning trees of the network. Version 1, called DPC/R-1, requires all n spanning trees of the network, while version 2, called DPC/R-2, uses only $k \leq n$ spanning trees, and thus reduces the first version's time complexity while producing similar results. For each version, the chapter considers Rel_{all}; Appendix B presents NTD-C/R with Rel_2 measure and its solution. The chapter shows that DPC/R always produces a feasible solution for the problem. Further, it shows that DPC/R will produce an optimal topology given a sequence of optimally ordered spanning trees. This chapter shows that generating optimally ordered spanning trees is NP-complete, and proposes five heuristics approaches to sort the spanning trees.

Extensive simulations using hundreds of benchmark networks that contain up to 200 nodes, 298 links, 2^{99} (s, t) paths and 1.899^{102} spanning trees show the merits of using the sorting methods, and the effectiveness of our algorithms. Further, the chapter shows the effectiveness of DPC/R-2 vis-à-vis to four existing state-of-the-art meta-heuristic techniques [12]-[15].

The preliminary version of DPC/R-1 with Rel_{all} measure has been presented in [47], while its full version has been published in [48]. Our work on DPC/R-1 with Rel_2 has been published in [4]. Note that DPC/R-1 approach was called DPCR-ST in [47], [48] and DPCR-P in [4].

The layout of this chapter is as follows. Section 4.1 formulates the NTD-C/R problem. Section 4.2 gives the DP formulation for NTD-C/R. Section 4.3 consists of five subsections; it describes DPC/R-1 in Section 4.3.1, its illustrating example in Section 4.3.2, its theoretical analysis in Section 4.3.3, five order criteria in Section 4.3.4, and DPC/R-2 that uses only $k \leq n$ spanning trees in Section 4.3.5. Section 4.4 provides simulation results to show the effectiveness and efficiency of our approaches. Finally, Section 4.5 concludes the chapter.

4.1 Problem Statement

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 not selected (Y_j=0). The following two equations describe the NTD-C/R problem.

$$\text{Minimize} \sum_{j=1}^{|E|} c_j \, Y_j \tag{4.1}$$

$$\text{Subject to Rel } (G_i=(V, E_i)) \geq R_{min} \tag{4.2}$$

Equation (4.1) calculates the minimum cost of a network topology $G_i=(V, E_i)$ that contains links $E_i=E-\{e_j \mid Y_j=0\}$; *i.e.*, E_i is a set of selected links in Eq. (4.2) that form G_i that has a reliability of at least R_{min}. One may solve the NTD-C/R problem by generating each possible set of links in Eq. (4.2) that form G_i. Then, calculate Cost(G_i) for each G_i that has $Rel_{all}(G_i) \geq R_{min}$, and select a G_i with the minimum cost as G_{min}. Unfortunately, this brute force solution, called BF-3 requires generating $2^{|E|}$ possible

link selections, *i.e.*, the G_i. Further, the $Rel_{all}(G_i)$ in Eq. (4.2) for each G_i requires exponential time; thus BF-3 is feasible only for designing small topologies.

As an alternative, let X_i be a decision variable {0, 1} that indicates if spanning tree ST_i in G=(V, E) is selected (X_i=1), or not selected (X_i=0). The following equations describe the NTD-C/R problem.

$$\text{Minimize Cost}\left(\bigcup_{i=1}^{|ST_G|} ST_i \; X_i\right) \tag{4.3}$$

$$\text{Subject to } Rel_{all}\left(\bigcup_{i=1}^{|ST_G|} ST_i \; X_i\right) \geq R_{min} \tag{4.4}$$

Equation (4.3) calculates the minimum cost of the network containing only the selected spanning trees ST_i from Eq. (4.4). One may generate all 2^n possible combinations of spanning trees that meet the constraint in Eq. (4.4). Then, for each combination that has $Rel_{all}(G_i)$ of at least R_{min}, use Eq. (4.3) to calculate its cost, and select a combination with the minimum cost as its G_{min}. This solution, henceforth called BF-4, is also prohibitive for use in large networks because in general a network contains $n=O(|V|^{|V|})$ spanning trees [20]. In Section 4.2, we propose a DP approach to solve Eq. (4.3) and Eq. (4.4).

To illustrate the NTD-C/R problem, consider the network in Fig. 2.1. For R_{min}=0.82, Fig. 4.1 shows the optimal network, G_{min}, whose links form a set of spanning trees {{1, 3, 4},{1, 3, 5},{1, 4, 5}} with $Rel_{all}(G_{min})$=0.841, and $Cost(G_{min})$=17; G_{min} does not contain spanning trees {1, 2, 4}, {1, 2, 5},{2, 4, 5}, {2, 3, 5}, and {2, 3, 4} because link 2 is not selected. Notice that $Rel_{all}(G)$=0.927; and therefore, if we set R_{min}=0.927, Eq. (4.1) would have Y_j=1 for each e_j, and Eq. (4.3) considers all spanning trees in G. Thus, for this case, Eq. (4.1) and Eq. (4.3) produce G_{min}=G with a reliability of 0.927. In contrast, Eq. (4.2) produces a reliability of 0 if the selected links in Eq. (4.1) do not form any spanning tree.

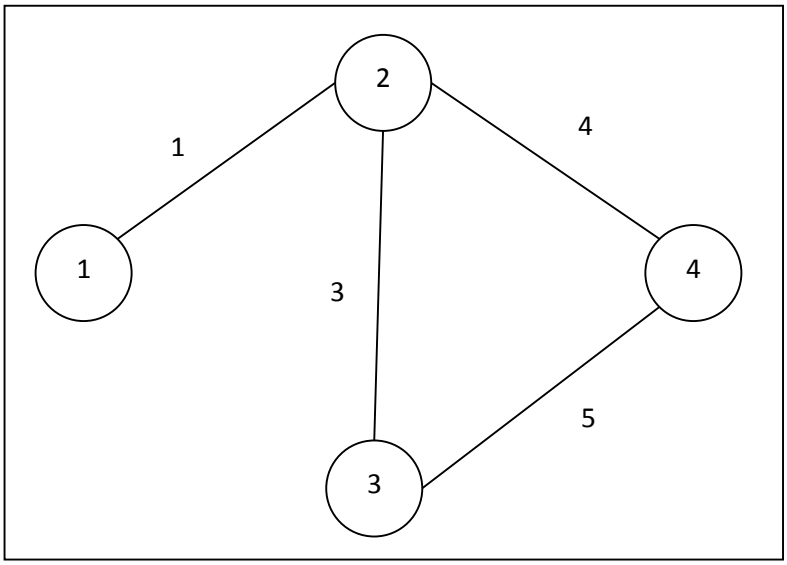


Fig. 4.1: Optimal solution of NTD-C/R with R_{min}=0.82 for CN in Fig. 2.1

4.2 Dynamic Programming Formulation for NTD-C/R

Let STX_i, for i=1, 2, …, n-1, n, be a sequence of spanning trees selected from i spanning trees in {ST_1, ST_2, …, ST_i}, and G_i=(V, $E_i \subseteq E$) be an induced graph whose links comprise of all links in STX_i. Note that $0 \leq |STX_i| \leq i$, and there are 2^n different STX_i; we aim to select STX_n with $Rel_{all}(G_n)$ of at least R_{min}, and the minimum cost, *i.e.*, $Rel_{all}(G_n) \geq R_{min}$ and minimum $Cost(G_n)$. An STX_i forms a *feasible* solution or topology if its reliability $Rel_{all}(STX_i) \geq R_{min}$; otherwise, it is a *non-feasible* solution. For Fig. 2.1 with R_{min}=0.82, STX_8=(ST_2, ST_7, ST_8) is a feasible solution because $Rel_{all}(STX_8)$=0.841$\geq R_{min}$. Two sets of spanning trees STX_i and STX_j are *equivalent* if they contain the same links that form the same topology, and thus they contain the same set of spanning trees, and have the same Rel_{all} and cost. For Fig. 2.1, STX_2=(ST_1, ST_2) and STX_3=(ST_1, ST_3) are equivalent because both form the same topology.

Let DP[1 .. n, 0 .. $\check{R}_{min}$] be a 2-dimensional DP table, where $\check{R}_{min}$=*round*($\sigma \times R_{min}$), for a positive integer multiplier σ, and a function *round*(α) that returns the closest integer

value of α. For example, the function returns $\check{R}_{min}$=92 ($\check{R}_{min}$=93) when we set σ=100, and R_{min}=0.9216 (R_{min}=0.9261).

Each element DP[i, $\check{r}$], for i=1, 2, …, n, $\check{r}$=0, 1, 2, …, $\check{R}_{min}$, stores five pieces of information: a cost C[i, $\check{r}$]>0, a reliability 0≤R[i, $\check{r}$]≤1.0, a sequence of spanning trees X[i, $\check{r}$]⊆ST_G, a set of links L[i, $\check{r}$]⊆E, and an integer index 0≤J[i, $\check{r}$]≤ σ. In essence, the columns of the DP table partition the reliability constraint R_{min} into σ consecutive reliability constraints, *i.e.*, R_{min}/σ, $(2\times R_{min})/\sigma$, …, $(\sigma\times R_{min})/\sigma=R_{min}$. Specifically, each column index $\check{r}$=0, 1, …, $\check{R}_{min}$, corresponds to a reliability constraint r=0, 1/σ, …, $(\check{R}_{min}/\sigma)\approx R_{min}$, *i.e.*, $r=\check{r}/\sigma$ and $\check{r}=round(\sigma\times r)$, and each DP[$i$, $\check{r}$] is used to store four pieces of information for each selected topology G_i that has $Rel_{all}(G_i)\geq r$. Specifically, for each $Rel_{all}(G_i)\geq r$, we set C[i, $\check{r}$]=Cost(G_i), R[i, $\check{r}$]=$Rel_{all}(G_i)$, X[i, $\check{r}$]=STX_i, and L[i, $\check{r}$]=E_i. For $Rel_{all}(G_i)<r$, we set C[i, $\check{r}$]=∞, R[i, $\check{r}$]=0, X[i, $\check{r}$]=( ), and L[i, $\check{r}$]={}. Note that C[i, $\check{r}$]=0 is not possible because each link is assumed to have a non-zero cost. As C[n, $\check{R}_{min}$] is the cost of G_n=(V, E_n⊆E) with $Rel_{all}(G_n)\geq R_{min}$, NTD-C/R aims to generate DP[n, $\check{R}_{min}$] that contains the minimum C[n, $\check{R}_{min}$], which represent the G_{min}.

For each range of columns $\check{r}_1\leq\check{r}\leq\check{r}_2$ in row i that contain the same reliability value, we set each j=J[i, $\check{r}$]=$\check{r}_2$. Thus, index j=J[i, $\check{r}$]=0, 1, 2, …, $\check{R}_{min}$ marks the ending column of a range of columns that have the same reliability. Notice that $\check{r}_\Delta$ has been used instead of j in [48]. For example, as later shown in Table 4.1, we store j=J[1, $\check{r}$]=49 at columns $\check{r}$=0 to $\check{r}$=49 because R[1, 0]=R[1, 1]= … =R[1, 49]. Note that we set j=J[i, $\check{r}$]=$\check{r}$ when $\check{r}_1=\check{r}_2$, *i.e.*, when the length of the range is one. Our DP approach computes each C[i, $\check{r}$] using the following four equations.

C[i, $\check{r}$]=Cost(ST_i) for i=1 with $Rel_{all}(ST_i)\geq r$ (4.5)

C[i, $\check{r}$]=∞ for i=1 with $Rel_{all}(ST_i)<r$ (4.6)

$C[i, \check{r}]=\text{Min}(C[i\text{-}1, \check{r}], \text{Cost}(ST_i))$ for $i>1$, and $\text{Rel}_{all}(ST_i)\geq r$ (4.7)

$C[i, \check{r}]=\text{Min}(C[i\text{-}1, \check{r}], \text{Cost}(L[i\text{-}1, j] \cup L_i))$ for $i>1$, $j\leq\check{r}$,

and $\text{Rel}_{all}(L[i\text{-}1, j] \cup L_i)\geq r$ (4.8)

We explain the DP formulation in Eq. (4.5) - Eq. (4.8) as follows; 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. In Eq. (4.5), when the first spanning tree has $\text{Rel}_{all}(ST_1)$ of at least r, it should be selected, giving $C[1, \check{r}]=\text{Cost}(ST_1)$. In contrast, when $\text{Rel}_{all}(ST_1)<r$, ST_1 is not selected because it does not meet the reliability constraint r; thus Eq. (4.6) sets $C[1, \check{r}]=\infty$ to denote that no spanning tree is selected.

Equations (4.7) and (4.8) are used for each remaining ST_i, for $i=2, 3, \ldots, n$. Eq. (4.7) considers two options, selecting or not selecting ST_i, when $\text{Rel}_{all}(ST_i)\geq r$, and selects the option that produces the minimum cost. Specifically, when ST_i is selected (not selected), its cost is $\text{Cost}(ST_i)$ ($C[i\text{-}1, \check{r}]$), and the equation selects the minimum between the two because both options satisfy the Rel_{all} requirement r. Note that the Rel_{all} value in the element would be changed to $\text{Rel}_{all}(ST_i)$ if ST_i is selected. Further, Eq. (4.7) considers a situation when no trees have been selected for column $\check{r}$, *i.e.*, $C[i\text{-}1, \check{r}]=\infty$, and $R[i\text{-}1, \check{r}]=0$, in which case it will select ST_i.

Equation (4.8) considers the case when selecting ST_i together with some previous sequence of selected trees that satisfies the required all-terminal reliability r, *i.e.*, $\text{Rel}_{all}(L[i\text{-}1, j] \cup L_i)\geq r$, for each possible $j=J[i\text{-}1, \check{r}]=0, 1, \ldots, \check{R}_{min}$. Like Eq. (4.7), Eq. (4.8) also considers the minimum cost between either selecting or not selecting ST_i; the former produces $\text{Cost}(L[i\text{-}1, j] \cup L_i)$, and the latter produces $C[i\text{-}1, \check{r}]$. Specifically, when ST_i is selected (not selected), the cost is calculated from the selected spanning

trees STX_i, (STX_{i-1}). Note that the Rel_{all} value in the column would be changed to $Rel_{all}(L[i\text{-}1, j] \cup L_i)$ if ST_i is selected. Further, Eq. (4.8) also considers a situation when no trees have been selected for column $\check{r}$, *i.e.*, $C[i\text{-}1, \check{r}]=\infty$, and $R[i\text{-}1, \check{r}]=0$, in which case it will select ST_i.

Like NTD-R/C of Section 3.2.3, the DP formulation for NTD-C/R in Eq. (4.5) - Eq. (4.8) are similar to DP solution for 0-1 knapsack problem [43]. In the 0-1 knapsack problem [43], there are n items (spanning trees in NTD-C/R), where each item i has weight x_i ($Rel_{all}(ST_i)$ in NTD-C/R) and value v_i ($Cost(ST_i)$ in NTD-C/R), and its goal is to select a set of items that have the *maximum* total value (*minimum* cost in NTD-C/R as stated in Eq. (4.3)) while having a total weight of no more than a given weight constraint X_{max} (R_{min} in NTD-C/R as stated in Eq. (4.4)). However, recall that in the 0-1 knapsack problem, the total cost of two items is the sum of each item's cost, while in NTD-C/R, $Cost(ST_i)+Cost(ST_p) \geq Cost(ST_i \cup ST_p)$ because ST_i and ST_p may contain common links. Therefore Eq. (4.8) must consider all possible values of j, *i.e.*, $J[i, \check{r}]$. Further, while the total capacity of two items in the 0-1 Knapsack problem equals the sum of each item's capacity, in NTD-C/R, $Rel_{all}(ST_i)+Rel_{all}(ST_p) \neq Rel_{all}(ST_i \cup ST_p)$, and $Rel_{all}(ST_i) > Rel_{all}(ST_p)$ does not always mean $Rel_{all}(ST_h \cup ST_i) > Rel_{all}(ST_h \cup ST_p)$, for any ST_h. Therefore, each $C[i, \check{r}]$ is not necessarily minimum, even when it is computed from two optimal sub problems. In Section 4.3.3, we will show that our heuristic DP solution generates an optimal topology when it uses an optimal order of spanning trees, defined later, as its input.

The solution for NTD-C/R, *i.e.*, DPC/R (described in Section 4.3) is similar to the solution to NTD-R/C, *i.e.*, DPR/C of Chapter 3. However, DPR/C cannot be used to solve the NTD-C/R for two main reasons. First, the DP formulations for NTD-C/R and

NTD-R/C are different; see Eq. (4.5) to Eq. (4.8) and Eq. (3.5) to Eq. (3.8). Second, to maximize the reliability in NTD-R/C, DPR/C needs to compute an exact reliability value for each entry of its DP table, a very time consuming step. On the other hand, NTD-C/R aims to minimize network cost; thus it is sufficient for DPC/R to generate only an approximated reliability, which can be solved using a significantly faster heuristic technique such as MCS [10]. Therefore, although the two problems are closely related, the solutions for NTD-C/R cannot be effectively used to solve NTD-R/C as they are either too expensive computationally or lack the necessary precision to generate an acceptable solution.

4.3 DPC/R Algorithm

Section 4.3.1 presents the first version of DPC/R, namely DPC/R-1, that requires all spanning trees of the network as its input to solve NTD-CR. Section 4.3.2 shows an illustrating example for DPC/R-1. Section 4.3.3 provides theoretical analysis of DPC/R-1. Specifically, subsection 4.3.3.1 shows that DPC/R-1 always produces feasible solutions, subsection 4.3.3.2 proves that the approach generates optimal topologies, given optimal order of spanning trees, and subsection 4.3.3.3 computes its time complexity. Section 4.3.4 proposes five greedy heuristics, *i.e.*, OC1 to OC5 to improve the effectiveness of DPC/R-1. Finally, Section 4.3.5 describes the second version of DPC/R, called DPC/R-2.

Note that Appendix B shows how to use DPC/R for Rel_2 measure. For this case, the algorithm requires a sequence of (s, t) paths. Further, the appendix introduces another order criterion, *i.e.*, OC6.

4.3.1 DPC/R Version 1

The first version of DPC/R algorithm, called DPC/R-1, directly applies Eq. (4.5) - Eq. (4.8). For a G=(V, E) that contains n spanning trees with reliability constraint R_{min}, DPC/R-1 *implicitly* constructs a DP table of size $n \times \check{R}_{min}$. However, DPC/R-1 keeps only two consecutive rows, called *row*1 and *row*2, and therefore it requires only a table of size $2 \times \check{R}_{min}$. Specifically, DPC/R-1 computes C[2, j] and R[2, j] in *row*2 using the information in C[1, $\check{r}$] and R[1, $\check{r}$] in *row*1, for all relevant columns $\check{r}$ and j.

DPC/R-1 Algorithm

1. **Initialize** C[1, $\check{r}$]=∞, R[1, $\check{r}$]=0, X[1, $\check{r}$]=( ), L[1, $\check{r}$]={}, J[1, $\check{r}$]=$\check{R}_{min}$, for $Rel_{all}(ST_1)<r$ // Eq. (4.6)
2. **for** ($\check{r} \leftarrow$ 0 to *round*($\sigma \times Rel_{all}(ST_1)$) **do** // Eq. (4.5)
3. C[1, $\check{r}$] ← $Cost(ST_1)$
4. R[1, $\check{r}$] ← $Rel_{all}(ST_1)$
5. X[1, $\check{r}$] ← ST_1
6. L[1, $\check{r}$] ← L_1
7. J[1, $\check{r}$] ← *round*($\sigma \times$R[1, $\check{r}$])
8. **end for** $\check{r}$
9. *Copy row*1 to *row*2
10. **for** ($i \leftarrow 2$ to n) **do** // Eq. (4.7) - Eq. (4.8)
11. **for** ($\check{r} \leftarrow$ 0 to *round*($\sigma \times Rel_{all}(ST_i)$) **do** // Eq. (4.7)
12. C[2, $\check{r}$] ← Min(C[1, $\check{r}$], Cost (ST_i))
13. **if** C[2, $\check{r}$]<$Cost(ST_i)$
14. X[2, $\check{r}$] ← X[1, $\check{r}$]
15. L[2, $\check{r}$] ← L[1, $\check{r}$]
16. **else**
17. X[2, $\check{r}$] ← ST_i
18. L[2, $\check{r}$] ← L_i
19. **end if**
20. R[2, $\check{r}$] ← Rel_{all}(L[2, $\check{r}$])
21. J[2, $\check{r}$] ← *round*($\sigma \times$R[2, $\check{r}$])
22. **end for** $\check{r}$
23. **for** ($y \leftarrow 0$ **to** $\check{R}_{min}$) **do** // Eq. (4.8)
24. **if** ($J[1, y] \neq J[1, y\text{-}1]$)
25. $j=J[1, y]$
26. **if** $Rel_{all}(L[1, j] \cup L_i) \geq \check{r}$
27. C[2, $\check{r}$] ← Min(C[1, $\check{r}$], Cost (L[1, j] ∪ L_i))
28. **if** C[2, $\check{r}$]<$Cost(L[1, j] \cup L_i)$
29. X[2, $\check{r}$] ← X[1, $\check{r}$]
30. L[2, $\check{r}$] ← L[1, $\check{r}$]
31. **else**
32. X[2, $\check{r}$] ← X[1, j] ∪ ST_i

33. $L[2, \check{r}] \leftarrow L[1, j] \cup L_i$
34. **end if**
35. $R[2, \check{r}] \leftarrow Rel_{all}(L[2, \check{r}])$
36. $J[2, \check{r}] \leftarrow round(\sigma \times R[2, \check{r}])$
37. **end if**
38. **end if**
39. **end for** ***y***
40. *Copy row*2 to *row*1
41. **end for** ***i***

After copying the contents of *row*1 to *row*2, it repeats the step until all spanning trees are considered. In this chapter, we set the integer multiplier σ, described in Section 4.2, to 100, and thus the DP table contains no more than 101 columns. Line 1 of DPC/R-1 Algorithm implements Eq. (4.6), while Lines 2 through 8 are based on Eq. (4.5). The remainder of the code is used to implement Eq. (4.7) and Eq. (4.8). Specifically, Eq. (4.7) is solved in Lines 11 through 22, Eq. (4.8) in Lines 23 through 39, and Line 40 copies the contents of *row*1 to *row*2.

4.3.2 Illustrating Example

To illustrate the DPC/R-1 algorithm, consider the network in Fig. 2.1, and R_{min}=0.82. Table 2.1 shows the network's link reliability and cost, and spanning tree reliability and cost. Our DPC/R-1 algorithm constructs the DP table shown in Table 4.1, and obtains the optimal topology in Fig. 4.1. For convenience, we show all eight rows, although our implementation creates only two rows. Each row of the table considers a ST_i for possible selection, and its columns are labeled by reliability values from 0 to $\check{R}_{min}$. Due to space limitation, Table 4.1 shows only a range of the reliability values $\check{r}$. Because $Rel_{all}(ST_1)$=0.486, and thus $\check{r}$=49, lines 2 through 8 in the DPC/R-1 algorithm set $C[1, \check{r}]$=13, $R[1, \check{r}]$=0.49, $X[1, \check{r}]=(ST_1)$, and $L[1, \check{r}]=\{2, 4, 5\}$ for $\check{r}$=0, …, 49 with $j=J[1, \check{r}]$=49; Fig. 4.2(a) shows the graph representation for $X[1, \check{r}]=(ST_1)$. Further, Eq.

(4.6) initializes the first row with C[1, $\check{r}$]=∞, R[1, $\check{r}$]=0, X[1, $\check{r}$]={}, and L[1, $\check{r}$]={} for $\check{r}$=50, …, $\check{R}_{min}$; and thus J[1, $\check{r}$]=$\check{R}_{min}$=82.

Next, for ST_2 with $Rel_{all}(ST_2)$=0.729, and thus $\check{r}$=73, Eq. (4.7) produces C[2, $\check{r}$]=C[1, $\check{r}$], R[2, $\check{r}$]=R[1, $\check{r}$], X[2, $\check{r}$]=X[1, $\check{r}$], and L[2, $\check{r}$]=L[1, $\check{r}$], because C[1, $\check{r}$]=13<Cost(ST_2)=15; thus this step keeps the non-feasible topology shown in Fig. 4.2 (a) for each column $\check{r}$=0, …, 49 in the row. In contrast, for $\check{r}$=50, …, 73, because C[1, $\check{r}$]=∞, Eq. (4.7) sets C[2, $\check{r}$]=15, R[2, $\check{r}$]=0.73, X[2, $\check{r}$]=(ST_2), L[2, $\check{r}$]={1, 4, 5}, and J[2, $\check{r}$]=73, Fig. 4.2 (b) shows the graph representation for X[2, $\check{r}$]=(ST_2). For $\check{r}$=74, …, $\check{R}_{min}$, selecting ST_2 for each $\check{r}$ in the range is feasible. For this case, we consider j=J[1, $\check{r}$] in *row* 1 that has two possible values, *i.e.*, j=49, and j=82 for $\check{r}$=0, …, 49, and $\check{r}$=50, …, $\check{R}_{min}$=82, respectively. For j=49, $Rel_{all}((L[i\text{-}1,\ j=49]=L_1) \cup L_2)$=0.88; thus Eq. (4.8) selects both ST_2 and ST_1 at column $\check{r}$=74, ..., $\check{R}_{min}$, and sets C[2, $\check{r}$]=18, R[2, $\check{r}$]=0.88, X[2, $\check{r}$]=(ST_1, ST_2), L[2, $\check{r}$]={1, 2, 4, 5}, and j=82. For this case, Eq. (4.8) produces a feasible topology for each column in the range; Fig. 4.2 (c) shows its graph representation. Note that Eq. (4.8) does not select both ST_1 and ST_2 at column $\check{r}$=0, ..., 73, because Cost($L_1 \cup L_2$)=18>C[1, $\check{r}$]=13. For j=82, we obtain Rel_{all}(L[i-1, j]={} ∪ ST_2={1, 4, 5})=0.729, and thus $\check{r}$=73, which is less than each $\check{r}$=74, ..., $\check{R}_{min}$ under consideration. Therefore, Eq. (4.8) produces exactly the same results at columns $\check{r}$=50, ..., 73 as previously obtained using Eq. (4.7).

As another example, for ST_3 with $Rel_{all}(ST_3)$=0.486, and thus $\check{r}$=49, for $\check{r}$=0, ..., 49, Eq. (4.7) selects ST_3, and sets C[3, $\check{r}$]=12, R[3, $\check{r}$]=0.49, X[3, $\check{r}$]=(ST_3), L[3, $\check{r}$]={1, 2, 4}, and J[3, $\check{r}$]=49, because Cost(ST_3)=12<C[2, $\check{r}$]=13; Fig. 4.2 (d) shows the graph representation for X[3, $\check{r}$]=(ST_3). For Eq. (4.8), there are three possible values for j=J[2, $\check{r}$], *i.e.*, j=49, j=73, and j=82 for $\check{r}$=0, …, 49, $\check{r}$=50, …, 73, and $\check{r}$=74, …, 82,

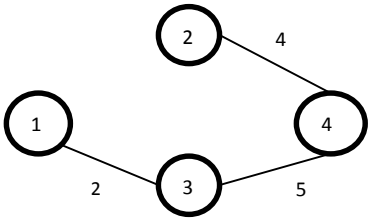
respectively. For j=49, $Rel_{all}((L[i\text{-}1, j=49]=L_1) \cup L_3)$=0.88, and thus Eq. (4.8) is used for $\check{r}$=0, …, 82 at row 3. Because Cost($L_1 \cup L_3$)=18>C[2, $\check{r}$]=13 for $\check{r}$=0, …, 49, and Cost($L_1 \cup L_3$)=18>C[2, $\check{r}$]=15 for $\check{r}$=50, …, 73, Eq. (4.8) does not update the DP table for $\check{r}$=0, …, 73. On the other hand, Cost($L_1 \cup L_3$)=C[2, $\check{r}$]=18 for $\check{r}$=74, ..., 82, and therefore the equation includes ST_3 to the selected trees in the previous row, *i.e.*, (ST_1), (ST_2), and (ST_1, ST_2) for columns $\check{r}$=0, ..., 49, $\check{r}$=50, ..., 73, and $\check{r}$=74, ..., 82 respectively. For this case, Eq. (4.8) produces (ST_1, ST_3), (ST_2, ST_3), and (ST_1, ST_2, ST_3) for columns $\check{r}$=0, ..., 49, $\check{r}$=50, ..., 73, and $\check{r}$=74, ..., 82, respectively.

Notice that the results, *i.e.*, (ST_1, ST_3), (ST_2, ST_3), (ST_1, ST_2, ST_3), and (ST_1, ST_2) are equivalent feasible solutions with a reliability of 0.88, and cost of 18; the table shows one of the solutions, *i.e.*, C[3, $\check{r}$]=18, R[3, $\check{r}$]=0.88, X[3, $\check{r}$]=(ST_1, ST_3), L[3, $\check{r}$]={1, 2, 4, 5}, and J[3, $\check{r}$]=88 as illustrated in Fig. 4.2 (c).

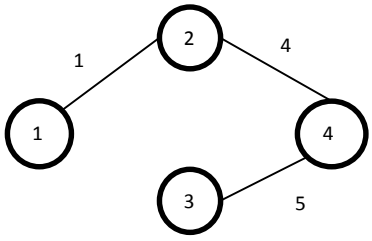
Repeating steps ST_4 through ST_8 to find the best solution, note that Eq. (4.8) is used at row 8 and does update the DP table for $\check{r}$=76, …, 82; because $Rel_{all}((L[7, j=57]=L_7) \cup L_8)$=0.841>$R_{min}$ and Cost($L_7 \cup L_8$)=17<C[7, $\check{r}$]=18 for $\check{r}$=76, …, 82. DPC/R-1 obtains X[8, $\check{R}_{min}$]=(ST_7, ST_8) with cost C[8, $\check{R}_{min}$]=17 and reliability R[8, $\check{R}_{min}$]=0.841.The optimal topology G_{min}, shown in Fig. 4.1.

Table 4.1: DP Table for CN in Fig. 2.1 with R_{min}=0.82

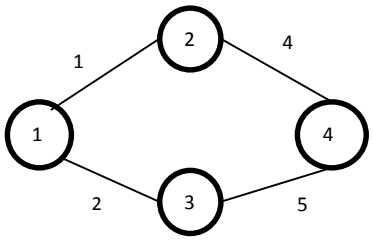
$\check{r}$ / ST_i	$\check{r}$=0, …, 38	$\check{r}$=39, …, 49	$\check{r}$=50, …, 57	$\check{r}$=58, …, 73	$\check{r}$=74, 75	$\check{r}$=76, …, 82
ST_1	C[1, $\check{r}$]=13 R[1, $\check{r}$]=0.49 X[1, $\check{r}$]=(1) L[1,$\check{r}$]={2,4,5} J[1, $\check{r}$]=49	13 0.49 (1) {2,4,5} 49	∞ 0 {} {} 87	∞ 0 {} {} 87	∞ 0 {} {} 87	∞ 0 {} {} 87
ST_2	13 0.49 (1) {2,4,5} 49	13 0.49 (1) {2,4,5} 49	15 0.73 (2) {1,4,5} 73	15 0.73 (2) {1,4,5} 73	18 0.88 (1,2) {1,2,4,5} 87	18 0.88 (1,2) {1,2,4,5} 87
ST_3	12 0.49 (3) {1,2,4} 49	12 0.49 (3) {1,2,4} 49	15 0.73 (2) {1,4,5} 73	15 0.73 (2) {1,4,5} 73	18 0.88 (1,3) {1,2,4,5} 87	18 0.88 (1,3) {1,2,4,5} 87
ST_4	11 0.38 (4) {2,3,5} 38	12 0.49 (3) {1,2,4} 49	15 0.73 (2) {1,4,5} 73	15 0.73 (2) {1,4,5} 73	18 0.88 (1,3) {1,2,4,5} 87	18 0.88 (1,3) {1,2,4,5} 87
ST_5	11 0.38 (4) {2,3,5} 38	12 0.49 (3) {1,4,5} 49	15 0.73 (2) {1,4,5} 73	15 0.73 (2) {1,4,5} 73	16 0.75 (4,5) {1,2,3,5} 75	18 0.88 (1,3) {1,2,4,5} 87
ST_6	9 0.38 (6) {2,3,4} 38	12 0.49 (3) {1,2,4} 49	14 0.75 (3,6) {1,2,3,4} 75	14 0.75 (3,6) {1,2,3,4} 75	14 0.75 (3,6) {1,2,3,4} 75	18 0.88 (1,3) {1,2,4,5} 87
ST_7	9 0.38 (6) {2,3,4} 38	11 0.57 (7) {1,3,4} 57	11 0.57 (7) {1,3,4} 57	14 0.75 (6,7) {1,2,3,4} 75	14 0.75 (6,7) {1,2,3,4} 75	18 0.88 (1,3) {1,2,4,5} 87
ST_8	9 0.38 (6) {2,3,4} 38	11 0.57 (7) {1,3,4} 57	11 0.57 (7) {1,3,4} 57	14 0.75 (6,7) {1,2,3,4} 75	14 0.75 (6,7) {1,2,3,4} 75	17 0.84 (7,8) {1,3,4,5} 85



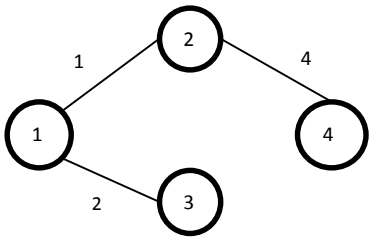
(a) X[1, [0 .. 49]] with $Rel_{all}(ST_1)$=0.49



(b) X[2, [50 .. 73]] with $Rel_{all}(ST_2)$=0.73



(c) X[2, [74 .. 82]]=X[3, [74 .. 82]] with $Rel_{all}(STX_3)=0.88$



(d) X[3, [0 .. 49]] with $Rel_{all}(ST_3)=0.49$

Fig. 4.2: Examples of graph representations for X[i=[1 .. n=8], $\check{r}$=[0 .. 82]].

4.3.3 DPC/R-1 Analysis

4.3.3.1 Feasible Solutions

In this subsection, we will use Lemma 4.1 and Theorem 4.1 to show that the DPC/R-1 algorithm will always produce feasible solutions for the NTD-C/R problem. We first present a definition that is used in the lemma and theorem.

Definition 4.1. A row i=1, 2, …, n in the DP table is a *non-feasible row* if none of its elements contain a feasible solution, *i.e.*, each R[i, $\check{r}$]<R_{min}, for $\check{r}$=1, 2, …, $\check{R}_{min}$.

Note that R[i, $\check{R}_{min}$]=$Rel_{all}(STX_i)\neq 0$ for each feasible solution STX_i, while a non-feasible row i has R[i, $\check{R}_{min}$]=0. For example, row i=1 in Table 4.1 is a non-feasible row, while the remaining rows are feasible rows.

Lemma 4.1. A non-feasible row i in DP table must have at least one column $\check{r}$ that contains X[i, $\check{r}$]=(ST_1, ST_2, …, ST_i), for i=1, 2, …, n, and $\check{r}$=1, 2, …, ($\check{R}_{min}$-1).

Proof. An X[i, l]=(ST_1, ST_2, …, ST_i) must be generated only from X[i-1, h]=(ST_1, ST_2, …, ST_{i-1}), for any column $\check{R}_{min}>l\geq h$, because the DPC/R-1 algorithm processes the spanning trees in sequence from ST_1, ST_2, …, ST_{i-1}, ST_i.

For row i=1, and $Rel_{all}(ST_1)<R_{min}$, Eq. (4.6) sets R[1, $\check{r}$]=0 and X[1, $\check{r}$]={} for all $r>Rel_{all}(ST_1)$, and thus R[1, $\check{R}_{min}$]=0. Further, Eq. (4.5) sets R[1, $\check{r}$]=$Rel_{all}(ST_1)$ and

X[1, $\check{r}$]=(ST_1) in each column $r \leq Rel_{all}(ST_1)$, and thus row 1 is a non-feasible row, and Lemma 4.1 is true for $i=1$.

For $i=z$, where $z=2, 3, \ldots, n$, let us assume that there *is* a non-feasible row z which contains X[z, h]=(ST_1, ST_2, …, ST_{z-1}, ST_z) at column $h<\check{R}_{min}$. We want to show that, if $z+1$ is a non-feasible row, then it must have at least one column $(\check{R}_{min}-1) \geq l \geq h$ that contains X[$z+1$, l]=(ST_1, ST_2, …, ST_{z-1}, ST_z, ST_{z+1}). Note that, when $Rel_{all}(ST_{z+1}) \geq R_{min}$, row $z+1$ is a feasible row as Eq. (4.7) sets X[$z+1$, $\check{R}_{min}$]=(ST_{z+1}) because $Cost(ST_{z+1})<C[z, \check{R}_{min}]=\infty$. Further, row $z+1$ is also a feasible row if $Rel_{all}(ST_1 \cup ST_2 \cup \ldots \cup ST_{z-1} \cup ST_z \cup ST_{z+1}) \geq R_{min}$, even when $Rel_{all}(ST_{z+1})<R_{min}$. For this case, $Cost(ST_1 \cup ST_2 \cup \ldots \cup ST_{z-1} \cup ST_z \cup ST_{z+1})<C[z, \check{R}_{min}]=\infty$, and thus Eq. (4.8) sets X[$z+1$, $\check{R}_{min}$]=(ST_1, ST_2, …, ST_{z-1}, ST_z, ST_{z+1}), and R[$z+1$, $\check{R}_{min}$]=$Rel_{all}(ST_1 \cup ST_2 \cup \ldots \cup ST_{z-1} \cup ST_z \cup ST_{z+1}) \geq R_{min}$. Thus, row $z+1$ is a non-feasible row only when $Rel_{all}(ST_1 \cup ST_2 \cup \ldots \cup ST_{z-1} \cup ST_z \cup ST_{z+1})<R_{min}$. For this case, we consider the largest column $h<\check{R}_{min}$ with X[z, h]=(ST_1, ST_2, …, ST_{z-1}, ST_z), *i.e.*, R[z, h]=h. Consequently, each column $l>h$ will contain X[z, l]={}, R[z, l]=0, and C[z, l]=∞ because it is not possible to have any subset of spanning trees from (ST_1, ST_2, …, ST_{z-1}, ST_z) with reliability larger than $Rel_{all}(ST_1 \cup ST_2 \cup \ldots \cup ST_{z-1} \cup ST_z)$. For this case, Eq. (4.8) will set X[$z+1$, l]=(ST_1, ST_2, …, ST_{z-1}, ST_z, ST_{z+1}), C[$z+1$, l]=Cost(ST_1, ST_2, …, ST_{z-1}, ST_z, ST_{z+1}), R[$z+1$, l]=Rel_{all}(ST_1, ST_2, …, ST_{z-1}, ST_z, ST_{z+1}), and L[$z+1$, l]=$L_1 \cup L_2 \cup \ldots \cup L_{z-1} \cup L_z \cup L_{z+1}$ at columns $l \leq R[z+1, l]$. **Q.E.D.**

Theorem 4.1. For a network G with $Rel_{all}(G) \geq R_{min}$, DPC/R-1 always generates a feasible solution for G.

Proof. We want to show that in the worst case DPC/R-1 generates G at DP[n, $\check{R}_{min}$] as its feasible solution. Following Lemma 4.1, each non feasible row $z=1, 2, \ldots, n$ must

contain at least one column $\check{r}$=1, 2, …, ($\check{R}_{min}$-1) with X[z, $\check{r}$]=(ST_1, ST_2, …, ST_{z-1}, ST_z). Thus, in the worst case, there is at least one column r in a non-feasible row n-1 that contains X[n-1, $\check{r}$]=(ST_1, ST_2, …, ST_{n-1}). For this case, Eq. (4.8) will set X[n, $\check{r}$]=(ST_1, ST_2, …, ST_{n-1}, ST_n), C[n, $\check{r}$]=Cost(ST_1, ST_2, …, ST_{n-1}, ST_n), R[n, $\check{r}$]=Rel_{all}(ST_1, ST_2, …, ST_{n-1}, ST_n), and L[n, $\check{r}$]=$L_1 \cup L_2 \cup \ldots \cup L_{n-1} \cup L_n$ at columns $\check{r}$≤round(R[n, $\check{r}$]). For Rel_{all}(G)≥R_{min}, R[n, $\check{r}$]≥R_{min}. Thus, in the worst case, DP[n, $\check{R}_{min}$] will contain a feasible solution. **Q.E.D**.

4.3.3.2 Optimality of DPC/R-1

This subsection establishes Theorem 4.2 and Theorem 4.3. Theorem 4.2 states that the DPC/R-1 algorithm produces an optimal topology if it uses a given sequence of optimally ordered spanning trees, called STX_{opt}; while Theorem 4.3 shows that generating STX_{opt} is NP-complete. We first describe two definitions that are used in the theorems.

Definition 4.2. A sequence of spanning trees STX_n⊆(ST_1, ST_2, …, ST_n) in graph G is called STX_{min} or *reliability minimal* if (i) $Rel_{all}(STX_n)$≥R_{min}, and (ii) $Rel_{all}(STX_n\text{-}(ST_j))$<$R_{min}$ for any ST_j∈STX_n.

Definition 4.3. An STX_{min} is called STX_{opt} or *reliability optimal* if Cost(STX_{min}) is the minimum among all possible STX_{min}.

For example, if R_{min}=0.87, there are six STX_{min}:(ST_1, ST_2), (ST_1, ST_3), (ST_1, ST_5), (ST_2, ST_3), (ST_2, ST_5), and (ST_3, ST_5); and six equivalent STX_{opt}:(ST_1, ST_2), (ST_1, ST_3),(ST_1, ST_5), (ST_2, ST_3), (ST_2, ST_5), and (ST_3, ST_5). The six STX_{opt} contain the same set of links, and thus form the same G_{min}. In general, however, a graph G may contain several different G_{min}.

Theorem 4.2. The DPC/R-1 algorithm produces an optimal network topology for a given STX_{opt}.

Proof. Without loss of generality, consider the DPC/R-1 algorithm is given $STX_{opt}=(ST_1, ST_2, \ldots, ST_\beta)$, for $\beta=|STX_{opt}|\leq n$. By definition, the feasible solution that DPC/R-1 produces from the given STX_{opt} includes all spanning trees in the set. Theorem 4.1 guarantees that DPC/R-1 in the worst case produces $X[n, \check{R}_{min}]=(ST_1, ST_2, \ldots, ST_n)$ that includes all the spanning trees. Because DPC/R-1 adds spanning trees in order, $X[\beta, \check{R}_{min}]=(ST_1, ST_2, \ldots, ST_\beta)$. Further, because by definition $STX_{opt}=(ST_1, ST_2, \ldots, ST_\beta)$ is the optimal topology, $Cost(STX_{opt})$ is the minimal among all feasible topologies. Therefore, Eq. (4.7) and Eq. (4. 8) will keep $X[i, \check{R}_{min}]=X[\beta, \check{R}_{min}]$, for $i=\beta+1, \beta+2, \ldots, n$. Therefore, $X[n, \check{R}_{min}]=X[\beta, \check{R}_{min}]=(ST_1, ST_2, \ldots, ST_\beta)$ is an optimal solution. **Q.E.D.**

Theorem 4.3. Generating STX_{min} and STX_{opt} of a general graph G is NP-Complete.

Proof. For STX_{min}, we prove the theorem by reduction from the subset-sum problem [26]. For all links e_j in G, let $\rho(ST_i)$ be the probability that each link $e_j \in ST_i$ is operational while each $e_j \notin ST_i$ fails, *i.e.*, $\rho(ST_i)=\prod_{e_j\in ST_i} r_j \prod_{e_j\notin ST_i}(1-r_j)$. Because all failures are assumed to be statistically independent, $Rel_{all}(G)=\rho(ST_1)+\rho(ST_2)+ \ldots + \rho(ST_n)$. Given a sequence $(ST_1, ST_2, \ldots, ST_n)$, each $\rho(ST_i)$ can be computed in polynomial time in the order of |E|, and the problem to generate STX_{min} is to find a subset of values from a set $\{\rho(ST_1), \rho(ST_2), \ldots, \rho(ST_n)\}$ such that their sum meets the required R_{min}. In other words, we set the target sum to R_{min}, and each value item to $\rho(ST_i)$. Because each STX_{opt} is STX_{min}, one can directly conclude that generating STX_{opt} is also NP complete. **Q.E.D**.

4.3.3.3 Time Complexity

The time complexity of DPC/R-1 can be computed as follows. The Cost(•) function used in the DPC/R-1 algorithm requires all unique links in the set of spanning trees •. For each $\check{r}$, Cost(•) returns the sum of C[i-1, $\check{r}$], and the cost of links in ST_i that are not in L[i-1, $\check{r}$]. Using the bit implementation [22], one requires only one bit OR, and one bit XOR operation to obtain the links in ST_i that are not in L[i-1, $\check{r}$]; and thus, for any •, Cost(•) can be computed in $O(|E|)$. DPC/R-1 uses the function at most once for every table entry, and therefore the worst case time complexity for using the function is $O(n \times |E| \times \check{R}_{min})$.

The Rel_{all}(•) function used in the DPC/R-1 algorithm can be implemented using any exact reliability calculation [22], heuristic technique [10], or approximation (bounding) method [25]. Note that unlike the solution for NTD-R/C, described in Chapter 3, which needs to compute exact Rel_{all} values using CAREL [22] to search for the topology with the maximum reliability, NTD-C/R aims to minimize network cost, and thus it is sufficient for its solution to generate only an approximated reliability, which can be solved using a significantly faster heuristic technique such as MCS [10]. Specifically, the simulation methods for reliability calculation that reduce the time complexity but produce only estimated reliability values, are acceptable for the heuristic solutions to NTD-C/R because the values are used only to test for the feasibility of the resulting topology. Note that each estimated value is used only to see if a solution is feasible.

In this chapter, we use a MCS [10] with time complexity $O(nr \times |V|^4)$ to estimate the Rel_{all}(•) of each candidate network; nr is the number of replication. Notice that Rel_{all}(•) is used only for each different j in each row i. Hence, in total, the time complexity of

using $\mathrm{Rel}_{all}(\bullet)$ is $O(\Omega \times nr \times |V|^4)$, where Ω is the total number of different j in the table; $\Omega \leq n \times \check{R}_{min}$. Thus, in the worst case, DPC/R-1 requires $O(\Omega \times nr \times |V|^4 + n \times |E| \times \check{R}_{min}) = O(\Omega \times n \times (|V|^4))$, since nr is constant value and $\check{R}_{min} \leq 100$. Notice that $\psi_{\check{r}\Delta}$ and b have been used instead of Ω and nr, respectively, in [48].

4.3.4 Improving the Efficiency of DPC/R-1

Our DPC/R-1 algorithm requires all n spanning trees of the network, which is not feasible for a network that contains a large number of spanning trees. Further, Theorem 4.2 states that DPC/R-1 generates an optimal solution only if it is given an optimal sequence of spanning trees, STX_{opt}. Unfortunately, as shown in Theorem 4.3, generating STX_{opt} is NP-complete. Therefore, to improve the effectiveness and time complexity of DPC/R-1, we use five different heuristic techniques, OC1, OC2, OC3, OC4 and OC5, described in Section 2.6.2 and 3.3.4, each of which sequentially generates only $0 \leq k \leq n$ spanning trees for its input. Note that each heuristic aims to generate k spanning trees that are expected to contain the trees in STX_{opt}. Note that Eq. (4.5) - Eq. (4.8) consider spanning trees starting from ST_1, and thus DPC/R-1 sets ST_1 as the least weighted spanning tree, ST_2 as the second least weighted, *etc*. Because Prim's algorithm requires a time complexity of $O(|E| \times \log|V|)$, the DPC/R-1 algorithm requires an extra $O(n \times (|E| \times \log|V|))$ time complexity for the improvement, *i.e*., $O(\Omega \times nr \times |V|^4 + n \times |E| \times \check{R}_{min} + n \times (|E| \times \log|V|)) = O(\Omega \times n \times (|V|^4))$.

4.3.5 DPC/R Version 2

Our improved version of DPC/R-1 algorithm, called DPC/R-2, generates only the first *k* least weight spanning trees, and sets the value of *k* dynamically as follows. For each of the five heuristics, consider that DPC/R-1 has generated and used a sequence of spanning trees ST_1, ST_2, …, ST_i to obtain a network topology G_i with cost C_i, and reliability $R_i \geq R_{min}$, for $i \leq n$. In other words, the DPC/R-2 algorithm obtains a feasible, but not necessarily optimal, solution G_i. Then, our DPC/R-2 algorithm generates the next least weight ST_{i+1}; and if it obtains a feasible solution G_{i+1} with reduced cost as compared to G_i, *i.e.*, $Cost(G_i) > Cost(G_{i+1})$, it keeps generating the subsequent spanning trees. The DPC/R-2 algorithm stops when it generates 10 consecutive spanning trees, none of which can further reduce topology cost. Specifically, consider the DPC/R-2 algorithm has generated a feasible solution G_k from sequence (ST_1, ST_2, …, ST_k). DPC/R-2 will terminate and return G_k as its best solution if G_{k+1}, G_{k+2}, …, G_{k+10}, generated respectively from (ST_1, ST_2, …, ST_{k-1}, ST_k), …, (ST_1, ST_2, …, ST_k, ST_{k+1}, …, ST_{k+10}) have $Cost(G_k)=Cost(G_{k+1})=Cost(G_{k+2})= \ldots =Cost(G_{k+10})$. Notice that using more than *k* spanning trees does not necessarily make DPC/R-2 generate better results due to the heuristic nature of the spanning tree order, except when (ST_1, ST_2, …, ST_k, ST_{k+1}, …, ST_n) is a STX_{opt}. However, this improvement does not require all spanning trees a priori. Thus the DPC/R-2 algorithm's time complexity becomes $O(\Omega \times nr \times |V|^4 + k \times |E| \times \check{R}_{min} + k \times (|E| \times \log|V|))$; reducing its running time for smaller values of *k*. As an example, as discussed in Section 4.4.4 for a grid network with 200 nodes and 298 links that contain 1.899^{102} spanning trees, the improvement enables the DPC/R-2 algorithm to generate results using only 1214 spanning trees.

4.4 Evaluation

First, in Section 4.4.1, we use DPC/R-1 on the 20 networks, described in Section 2.6.1, to observe the effects of different input orders on the performance of DPC/R-1 for both Rel_{all} and Rel_2 measures. Second, in Section 4.4.2, we compare the effectiveness of DPC/R for Rel_{all} measure against the state-of-the-art approaches ACO-SA [12], NGA [13], LS-NGA [14], and BDD [15]. For this case, we use DPC/R-2 with OC1, OC2 and OC3, and run all methods on the 76 fully connected networks described in Section 2.6.1. Because there is no known optimal results for NTD-C/R with Rel_2, in Section 4.4.3, we generate 100 benchmark networks to gauge the effectiveness of DPC/R with the reliability measure. The section uses both DPC/R-1 and DPC/R-2 with OC1 and OC6. Finally, in Section 4.4.4, we use the five grid networks in Section 2.6.1 that contain up to 1.899^{102} spanning trees and 2^{99} (s, t) path to show the efficiency of DPC/R-2, in terms of its required number of spanning trees and (s, t) paths, k, and CPU time as well as its effectiveness.

4.4.1 The Effect of Input Orders on DPC/R Performance

For each of the 20 network topologies in Table 2.3 and Table 2.4, we use Prim's algorithm [26] to generate spanning trees sorted based on order criteria OC1 to OC5. Then we run DPC/R-1 to generate the topology for each of the 100 sequence of generated spanning trees. As shown in Table 4.2, DPC/R-1 using each order criterion produces between 18 and 8 of the best results (in bold) as compared to only 3 using the random order. Thus, the order criteria help DPC/R-1 in producing better results. The table shows that OC1 is the best performer, followed by OC2 and OC3 with 18, 14, and 15 best results; OC4 and OC5 are the worst with 8 and 10 best results respectively.

To see the effects of pre-ordering (*s*, *t*) paths on the performance of DPC/R-1, we use OC1 and OC6, described in Section 3.3.4 and Appendix B respectively, to obtain weight w_j for each link e_j. Then, we use Yen's algorithm [46] to sort the path set in increasing weight. Note that we select OC1 because it has been shown effective for NTD-R/C in Section 3.4.1 as well as for NTD-C/R in Table 4.2. To see the effects of using different values of δ for OC6, we set δ=1, δ=2 and δ=10; see Eq. (C.7) in Appendix B for more details.

Table 4.2: The effects of spanning tree orders on the performance of DPC/R-1

CN	***R_{min}***	**Random**	**Cost(G_n) for each order criterion**				
			OC1	**OC2**	**OC3**	**OC4**	**OC5**
$CN_8^{4,5}$	0.82	18	**17**	**17**	**17**	18	18
$CN_{21}^{5,8}$	0.80	28	**26**	**26**	28	27	**26**
$CN_{21}^{6,8}$	0.80	28	**23**	25	28	27	25
$CN_{55}^{6,9}$	0.80	38	**32**	**32**	**32**	37	37
$CN_{368}^{7,12}$	0.80	35	**28**	33	**28**	**28**	**28**
$CN_{1033}^{7,15}$	0.80	88	**75**	**75**	**75**	84	82
$CN_{247}^{8,12}$	0.80	38	34	**33**	**33**	**33**	**33**
$CN_{256}^{8,12}$	0.80	34	**29**	**29**	**29**	**29**	32
$CN_{576}^{8,13}$	0.80	36	**30**	**30**	**30**	34	**30**
$CN_{171}^{9,12}$	0.80	32	**31**	32	**31**	35	32
$CN_{327}^{9,13}$	0.80	**47**	**47**	**47**	51	49	**47**
$CN_{647}^{9,14}$	0.80	**26**	**26**	**26**	**26**	**26**	29
$CN_{2112}^{10,21}$	0.75	**33**	**33**	**33**	**33**	**33**	**33**
$CN_{1598}^{11,21}$	0.80	71	**63**	**63**	**63**	**63**	**63**
$CN_{3666}^{13,22}$	0.70	55	52	**50**	**50**	**50**	**50**
$CN_{7683}^{16,30}$	0.80	75	**67**	69	**67**	70	72
$CN_{9471}^{17,25}$	0.80	49	**47**	**47**	**47**	**47**	**47**
$CN_{21456}^{18,27}$	0.80	60	**46**	54	48	54	58
$CN_{24173}^{20,30}$	0.80	86	**81**	86	**81**	86	83
$CN_{18257}^{21,26}$	0.80	67	**58**	**58**	62	65	**58**
Total		3	18	14	15	8	10

As shown in Table 4.3, DPC/R-1 using each order criterion produces between 13 and 18 of the best results (in bold) as compared to only 4 using a random order. Thus, the

order criteria help DPC/R-1 in producing better results. The table shows that OC6 with δ=1 is the best performer, followed by OC1, producing 18 and 17 of the best results respectively. However, with δ=2 and δ=10, OC6 produces only 15 and 13 best results, respectively, and the results are the subset of those produced with δ=1. The results show that OC6 performs better when δ=1, validating our analysis in Appendix B. Notice that both OC1 and OC6 produce exactly the same results except for five networks, in which OC6 (OC1) produce 3 (2) better results as compared to OC1 (OC6). Thus, we suggest running DPC/R-1 for the (*s*, *t*) paths input using OC1 and OC6, and select the better results to improve its performance.

Table 4.3: The effects of path orders on the performance of DPC/R-1

Input			**$Rel_2(G_m)$ for each order criterion**			
CN	***R_{min}***	**Random**	**OC1**	**OC6**		
				δ=1	δ=2	δ=10
$CN_4^{4,5}$	0.85	17	**18**	**18**	**18**	17
$CN_9^{5,8}$	0.80	27	**21**	25	25	25
$CN_7^{6,8}$	0.70	**14**	**14**	**14**	**14**	**14**
$CN_{13}^{6,9}$	0.75	21	**19**	**19**	**19**	**19**
$CN_{25}^{7,12}$	0.80	28	**26**	**26**	**26**	**26**
$CN_{14}^{7,15}$	0.80	**31**	**31**	**31**	**31**	**31**
$CN_{20}^{8,12}$	0.80	32	38	**31**	38	38
$CN_{24}^{8,12}$	0.60	33	**31**	**31**	**31**	**31**
$CN_{29}^{8,13}$	0.50	24	**20**	**20**	**20**	**20**
$CN_{13}^{9,12}$	0.60	**27**	**27**	**27**	**27**	**27**
$CN_{18}^{9,13}$	0.75	30	25	**22**	25	25
$CN_{44}^{9,14}$	0.75	29	**27**	**27**	**27**	**27**
$CN_{64}^{10,21}$	0.80	35	**33**	**33**	**33**	**33**
$CN_{18}^{11,21}$	0.82	**18**	**18**	**18**	21	21
$CN_{281}^{13,22}$	0.60	44	**29**	**29**	**29**	**29**
$CN_{36}^{16,30}$	0.70	35	31	**30**	**30**	**30**
$CN_{136}^{17,25}$	0.60	28	**27**	30	30	30
$CN_{205}^{18,21}$	0.70	58	**44**	**44**	**44**	**44**
$CN_{780}^{20,30}$	0.60	48	**31**	**31**	**31**	**31**
$CN_{44}^{21,26}$	0.60	25	**21**	**21**	**21**	26
Total		**4**	**17**	**18**	**15**	**13**

4.4.2 Performance of DPC/R-2

To evaluate the effectiveness of DPC/R-2, we first compare, in Section 4.4.2.1, its performance against known best results on the 76 fully connected networks that are reported in [12]. The subsection also analyse the merits of using order criteria OC1, OC2, and OC3. Then, in Section 4.4.2.2, we compare the results with those generated by the state-of-the-art approaches ACO-SA [12], NGA [13], LS-NGA [14], and BDD [15].

4.4.2.1 Effectiveness of DPC/R-2

For each of the 76 fully connected network topologies in [12], we first generate its spanning trees using four different orders: OC1, OC2, OC3, and random. We use Prim's algorithm [26] to generate random order spanning trees, and modify the algorithm to produce spanning trees in order following each of the three OCs. Then, we run DPC/R-2 on each of the four set of spanning trees.

Table 4.4 compares the effects of using OC1, OC2, OC3, and random order on the performance of DPC/R-2. Note that each $G_{V,C_{total}}^{r_j,R_{min}}$ in the first column denotes a fully connected network G with |V| nodes, equal link reliability r_j, reliability constraint R_{min}, and total link cost C_{total}; see Appendix C for each network's cost matrix. The second column of the table, C_{min}, is the minimum cost of each topology with reliability of at least R_{min} as reported in [12]. As stated in [13], the reliability of each topology with cost C_{min} was estimated using a Monte Carlo method that produces result within 1% of R_{min}.

Table 4.4: The experimental results of DPC/R-2 on the 76 networks

CN	C_{min}	DPC/R-2					BDD	
		C_{best}	Rel_{all}	k	*Order*	CPU sec.	C_{best}	Rel_{all}
$G_{6,709}^{0.9,0.9}$	231	231	0.93	62	1,2,3,r	9.08	*N/A*	*N/A*
$G_{6,851}^{0.9,0.9}$	239	239	0.95	70	1,2,3,r	7.74	*N/A*	*N/A*
$G_{6,835}^{0.9,0.9}$	227	227	0.94	15	1,2,3,r	1.15	*N/A*	*N/A*
$G_{6,773}^{0.9,0.9}$	212	212	0.92	99	2	14.1	*N/A*	*N/A*
$G_{6,705}^{0.9,0.9}$	184	184	0.94	99	1,3,r	9.72	*N/A*	*N/A*
$G_{6,709}^{0.9,0.95}$	254	254	0.95	341	1,2,3,r	20.89	*N/A*	*N/A*
$G_{6,851}^{0.9,0.95}$	286	**239**	0.95	89	1,2,3	7.8	*N/A*	*N/A*
$G_{6,715}^{0.9,0.95}$	275	**234**	0.95	95	1,2,3	8.13	*N/A*	*N/A*
$G_{6,773}^{0.9,0.95}$	255	**236**	0.95	104	1,2,3	10.3	*N/A*	*N/A*
$G_{6,705}^{0.9,0.95}$	198	**193**	0.95	76	2	7.03	*N/A*	*N/A*
$G_{6,709}^{0.95,0.95}$	227	**185**	0.95	80	1,3	4.47	*N/A*	*N/A*
$G_{6,851}^{0.95,0.95}$	213	213	0.97	94	1,3,r	7.69	*N/A*	*N/A*
$G_{6,835}^{0.95,0.95}$	190	190	0.97	10	1,2,r	2.6	*N/A*	*N/A*
$G_{6,773}^{0.95,0.95}$	200	200	0.95	99	2	11.6	*N/A*	*N/A*
$G_{6,705}^{0.95,0.95}$	179	**148**	0.95	99	1,2,3	5.94	*N/A*	*N/A*
$G_{7,803}^{0.9,0.9}$	189	189	0.91	61	1,2,3	13.7	*N/A*	*N/A*
$G_{7,1028}^{0.9,0.9}$	184	190	0.95	92	2	14.7	*N/A*	*N/A*
$G_{7,1101}^{0.9,0.9}$	243	**200**	0.93	88	1,3	9.79	*N/A*	*N/A*
$G_{7,2816}^{0.9,0.9}$	129	129	0.91	93	1,2,3,r	7.68	*N/A*	*N/A*
$G_{7,1007}^{0.9,0.9}$	124	124	0.93	66	1,2,3,r	8.11	*N/A*	*N/A*
$G_{7,803}^{0.9,0.95}$	205	205	0.96	61	1,2,3,r	16.13	*N/A*	*N/A*
$G_{7,1028}^{0.9,0.95}$	209	209	0.96	92	1,2,3,r	9.2	*N/A*	*N/A*
$G_{7,1101}^{0.9,0.95}$	268	**264**	0.95	403	1,2,3	18.48	*N/A*	*N/A*
$G_{7,816}^{0.9,0.95}$	143	**139**	0.95	566	1,2,3	28.39	*N/A*	*N/A*
$G_{7,1007}^{0.9,0.95}$	153	153	0.96	94	1,2,3,r	13.42	*N/A*	*N/A*
$G_{7,803}^{0.95,0.95}$	185	**180**	0.95	61	1,3	7.94	*N/A*	*N/A*
$G_{7,1028}^{0.95,0.95}$	182	182	0.95	92	1,2,3	13.32	*N/A*	*N/A*
$G_{7,1101}^{0.95,0.95}$	230	**228**	0.95	489	1,2,3	14.16	*N/A*	*N/A*
$G_{7,816}^{0.95,0.95}$	122	129	0.98	93	1,2,3,r	4.6	*N/A*	*N/A*
$G_{7,1007}^{0.95,0.95}$	124	124	0.99	66	1,2,3,r	17.88	*N/A*	*N/A*
$G_{8,1343}^{0.9,0.9}$	208	**184**	0.90	95	1,2,3,r	11.54	218	0.92
$G_{8,1351}^{0.9,0.9}$	203	203	0.92	48	1,2,3	10.46	213	0.92
$G_{8,1352}^{0.9,0.9}$	211	211	0.93	451	1,3	23.27	211	0.92
$G_{8,1452}^{0.9,0.9}$	291	299	0.90	182	1	19.56	300	0.91
$G_{8,1263}^{0.9,0.9}$	178	178	0.91	99	1,3,r	11.4	181	0.93
$G_{8,1343}^{0.9,0.95}$	247	**223**	0.95	95	1,2,3	12.14	259	0.96
$G_{8,1351}^{0.9,0.95}$	247	**233**	0.95	97	1,3	15.8	253	0.95
$G_{8,1352}^{0.9,0.95}$	245	**232**	0.95	489	1,2,3,r	29.5	245	0.95
$G_{8,1452}^{0.9,0.95}$	336	**332**	0.95	331	1	27.97	351	0.96

$G_{8,1263}^{0.9,0.95}$	202	**181**	0.95	89	1,3	22.66	202	0.95
$G_{8,1343}^{0.95,0.95}$	179	179	0.97	96	1,2,3,r	6.6	179	0.96
$G_{8,1351}^{0.95,0.95}$	194	194	0.97	337	1,3,r	20.07	196	0.96
$G_{8,1352}^{0.95,0.95}$	197	**192**	0.95	499	1,3	25.87	197	0.96
$G_{8,1452}^{0.95,0.95}$	276	282	0.96	341	1	13.36	280	0.96
$G_{8,1263}^{0.95,0.95}$	173	**150**	0.95	99	2	12.29	184	0.97
$G_{9,1859}^{0.9,0.9}$	239	242	0.90	79	1,3	9.10	244	0.93
$G_{9,1897}^{0.9,0.9}$	191	212	0.91	92	2	7.74	194	0.91
$G_{9,1828}^{0.9,0.9}$	257	**252**	0.90	92	3	11.15	273	0.90
$G_{9,1749}^{0.9,0.9}$	171	171	0.90	94	1,2,3,r	14.1	183	0.91
$G_{9,1678}^{0.9,0.9}$	198	**195**	0.91	98	1,3	9.72	198	0.91
$G_{9,1859}^{0.9,0.95}$	286	**279**	0.96	93	1,2,3	10.89	286	0.96
$G_{9,1897}^{0.9,0.95}$	220	236	0.96	99	1,2,3,r	7.8	237	0.95
$G_{9,1828}^{0.9,0.95}$	306	**296**	0.95	242	1,2,3	18.13	306	0.95
$G_{9,1749}^{0.9,0.95}$	219	**200**	0.95	89	1,2,3	20.3	219	0.95
$G_{9,1678}^{0.9,0.95}$	237	**212**	0.95	583	1	37.03	239	0.95
$G_{9,1859}^{0.95,0.95}$	209	214	0.97	79	2	4.47	209	0.97
$G_{9,1897}^{0.95,0.95}$	171	199	0.95	332	2	27.69	171	0.95
$G_{9,1828}^{0.95,0.95}$	233	233	0.97	227	1	20.6	249	0.96
$G_{9,1749}^{0.95,0.95}$	151	151	0.95	95	1,3	11.6	177	0.97
$G_{9,1678}^{0.95,0.95}$	185	**183**	0.98	290	1	15.94	206	0.95
$G_{10,1803}^{0.9,0.9}$	131	131	0.90	104	2	13.7	131	0.91
$G_{10,2155}^{0.9,0.9}$	154	154	0.92	102	1,2,3,r	14.7	154	0.91
$G_{10,1828}^{0.9,0.9}$	267	**250**	0.90	111	1,3	9.79	*N/A*	*N/A*
$G_{10,2546}^{0.9,0.9}$	263	263	0.94	410	1,3	27.68	263	0.94
$G_{10,2517}^{0.9,0.9}$	293	309	0.91	127	1,3,r	18.11	309	0.91
$G_{10,1803}^{0.9,0.95}$	153	153	0.95	102	2	16.13	164	0.95
$G_{10,2155}^{0.9,0.95}$	197	**195**	0.95	364	1,2,3	24.2	205	0.95
$G_{10,1828}^{0.9,0.95}$	311	321	0.95	111	1,2,3,r	18.48	*N/A*	*N/A*
$G_{10,2546}^{0.9,0.95}$	291	291	0.95	421	1,2,3,r	28.39	309	0.96
$G_{10,2517}^{0.9,0.95}$	358	360	0.95	150	1,4	13.42	366	0.96
$G_{10,1803}^{0.95,0.95}$	121	125	0.96	89	1,3,r	7.94	127	0.95
$G_{10,2155}^{0.95,0.95}$	136	136	0.98	222	2	23.32	144	0.90
$G_{10,1828}^{0.95,0.95}$	236	**233**	0.97	103	1,3	14.16	*N/A*	*N/A*
$G_{10,2546}^{0.95,0.95}$	245	**231**	0.97	386	1,2,3,r	24.6	256	0.96
$G_{10,2517}^{0.95,0.95}$	268	296	0.97	114	1	17.88	277	0.96
$G_{11,2609}^{0.9,0.9}$	246	246	0.91	215	2	26.34	*N/A*	*N/A*

Each of the 76 C_{best} in column 3 is the *minimum* among the costs of topologies generated using OC1, OC2, OC3 and random, and column Rel_{all} gives its reliability. Column CPU shows the CPU time of running our approach for each network. Our

DPC/R-2 algorithm produces topologies with $C_{best}<C_{min}$, and reliability within 0.5% of R_{min}; we used the MCS [10] to calculate the reliability. Note that we use the *round* function that will make each topology with cost C_{best} and reliability of 0.745 an acceptable solution for R_{min}=0.75 because *round* (0.745)=75. Column *Order* shows which order, 1=OC1, 2=OC2, 3=OC3, r=random, is used to produce each C_{best}; *e.g.*, *Order*={1,3,4} means the corresponding C_{best} is generated using either random, OC1, or OC3, but not OC2. Column *k* shows the maximum number of spanning trees among the four order criteria used by the DPC/R-2 algorithm to generate their topologies.

As shown in Table 4.4, DPC/R-2 using OC1, OC2, and OC3 produced 81.5% (62 of 76) best results (underlined), 30 topologies of which (bold) have lower cost than the C_{min} reported in [13]. Further, DPC/R-2 produced the topologies using only 10/1296=0.77% to 583/4782969=0.01% of the spanning trees contained in the networks, using CPU times ranging between 1.15 and 37.03 seconds, and thus our approach is very efficient. Consistent with its time complexity, described in Section 4.3.3.1, as shown in Table 4.4, DPC/R-2 requires a larger CPU time when it uses a larger number of spanning trees, *k*, to generate its results.

Table 4.4 also shows that DPC/R-2 with random ordered spanning trees generates C_{best} only in 28 of 76 networks (36.8%), which is the worst as compared to OC1 (82.8%), OC2 (63.1%), and OC3 (72.3%). Further, for each case in which the random order generates C_{best}, at least one of the other three orders was also able to produce the result. This result further emphasizes the merit of using ordered spanning trees for our DP approach.

To compare the performances of OC1, OC2, and OC3, we summarize their results from Table 4.4 in Tables 4.5 and 4.6. The tables show the total number of topologies

generated with cost C_{best} and their cost optimality with respect to C_{min}, *i.e.*, $C_{best}>C_{min}$, $C_{best}=C_{min}$, $C_{best}<C_{min}$. As shown in Table 4.5, OC1 is the best performer, producing C_{best} 82.8% of the time, followed by OC3 with 72.3%, and OC2 with 63.1%; see the column Total. For each order, the last column in the table shows the total number of topologies with cost C_{best} that can only be generated using its two alternative order criteria; *e.g.*, row 1 of the table shows that OC1 produces 13 topologies with cost worse than that produced using OC2 or OC3.

Table 4.6 shows the total number of C_{best} uniquely produced using one or more of the three different OCs. The table shows that there are in total 8, 12, and 1 topology with cost C_{best} uniquely generated by OC1, OC2, and OC3, respectively, and the three criteria produce the same topologies 35/76=46% of the time. Further, there are 1, and 19 topologies that can only be generated by either OC1 or OC2, and OC1 or OC3, respectively. The results show that it is important for DPC/R-2 to use the three order criteria, OC1, OC2, and OC3, and select the best among their results to generate topologies with lower costs. As shown in the table, such approach produces only 18.4% topologies with larger costs.

Table 4.5: Comparisons among OC1, OC2, and OC3

Cost / Order	Total number of topologies with cost C_{best}				Total number of topologies with cost C_{best} using the other two sorting criteria
	$C_{best}<C_{min}$	$C_{best}=C_{min}$	$C_{best}>C_{min}$	Total	
OC1	27 (35.5%)	26 (34.2%)	10 (13.1%)	63 (82.8%)	13 (17.2%)
OC2	17 (22.3%)	24 (31.5%)	7 (9.2%)	48 (63.1%)	28 (36.8 %)
OC3	25 (32.8%)	24 (31.5%)	6 (7.8%)	55 (72.3%)	21 (27.6%)

Table 4.6: The distribution of C_{best} generated using one or more sorting criteria

Cost / Order			
	$C_{best}<C_{min}$	$C_{best}=C_{min}$	$C_{best}>C_{min}$
OC1	3	1	4
OC2	2	6	4
OC3	1	0	0
OC1, OC2	0	1	0
OC1, OC3	9	7	3
OC2, OC3	0	0	0
OC1, OC2, OC3	15	17	3
Total	30 (39.4%)	32 (42.1%)	14 (18.4%)

4.4.2.2 DPC/R-2 Versus Existing Approaches

We aim to use Table 4.4 also to compare the performance of DPC/R-2 algorithm on the 76 fully connected networks against those of four state-of-the-art approaches: ACO-SA [12], NGA [13], LS-NGA [14], and BDD [15]. The table shows the C_{best} and Rel_{all} for the 45 networks generated using BDD as reported in [15]; each *N/A* denotes each of the 31 non-reported values. However, we cannot present the results reported in [12]-[14] on the same table because they are presented in a different format. Further, we can't compare their CPU time due to the difference in the CPU processors and simulation environment. As an alternative, we have summarized the performance of all the evaluated algorithms in Table 4.7 and Fig. 4.3.

As shown in Table 4.7, NGA, LS-NGA, and ACO-SA generate the best solutions for 16, 24, and 48 out of 76 instances, respectively, while BDD obtains the best solutions for 14 out of 45 instances. On the other hand, DPC/R-2 produces 62 out of the 76 best results (81.5%), see Fig. 4.3, and thus it has significantly higher effectiveness over the existing algorithms. Further, 30 of 62 C_{best} generated using the DPC/R-2 algorithm are better than C_{min}. These results show the superiority of our efficient DP approach as compared to the existing state-of-the-arts solutions [12]-[15].

Table 4.7: Comparison between DPC/R-2, NGA, LS-NGA, ACO-SA and BDD

	DPC/R-1	NGA	LS-NGA	ACO-SA	BDD
$C_{best}=C_{min}$	32 (42.1%)	16 (21%)	24 (31.5%)	48 (63.1%)	14 (33.33%)
$C_{best}<C_{min}$	30 (39.4%)	N/A	N/A	N/A	N/A

4.4.3 DPC/R Performance on Benchmark Networks

We generated 100 networks from the 20 topologies, described in Section 2.6.1, to benchmark the optimality of DPC/R with Rel_2. Let $\alpha>0$ be the number of links deleted from a network. For $\alpha=1$, each best topology $G_i=(V, E-\{e_j\})$, for any $e_j \in E$, is generated as follows. For each possible $e_j \in E$, we first generated G_i; there are $|E|$ different G_i. For each G_i, we calculated $Rel_2(G_i)$ using MCS [10], and selected G_i with maximum reliability as G_{min} with cost $Cost(G_{min})$. We repeat the steps for $\alpha=2$, 3, 4, 5. We did not generate benchmark networks for larger α because, as discussed in Section 2.6.1, it would be very time consuming. Note that the solution for each benchmark topology may not be optimal because MCS [10] does not always produce exact reliability. For each generated network G_i, we set $R_{min}=Rel_2(G_{min})$, the tightest possible constraint, and thus each benchmark evaluates the worst possible performance for our DPC/R. We have run DPC/R-1 twice, once using OC1 and another using OC6 with $\delta=1$, to generate topologies from the 100 benchmark networks, and take the better results between them. Note that we consider only OC1 and OC6 with $\delta=1$ because they have been shown the best OCs in the previous simulations. Our simulations in Table 4.8 show that DPC/R with OC1 and OC6 produces 91% best results. Interestingly, of the 9 non-optimal results, DPC/R produces network with reliability no worse than 10.97% off from optimal, and most of the non-optimal results has higher cost than that for optimal by

up to 0.13%. Further, DPC/R-2 uses only between 8.89% and 27.5% of each network's total paths to generate the results.

Table 4.8: Comparison between DPC/R (using OC1 and OC6) and optimal results

Input			*k*	DPC/R-1 and DPC/R-2 results		
CN	α	R_{min}		**$Rel_2(G_k)$**	**$Cost(G_k)$**	**CPU time Sec.**
$CN_{25}^{7,12}$	2	0.812	4(16%)	0.828(1.97%)	18(0.13%)	0.23(12.30%)
$CN_{29}^{8,13}$	2	0.816	6(20.6%)	0.816(0.00%)	22(0.05%)	0.62(7.84%)
$CN_{29}^{8,13}$	4	0.702	8(27.5%)	0.779(10.97%)	27(0.08%)	0.12(10.35%)
$CN_{44}^{9,14}$	1	0.881	12(27.2%)	0.883(0.23%)	35(0.03%)	0.27(9.67%)
$CN_{64}^{10,21}$	1	0.90	10(15.6%)	0.904(0.44%)	36(0.06%)	0.18(14.89%)
$CN_{64}^{10,21}$	5	0.702	12(18.7%)	0.764(8.83%)	22(0.10%)	0.13(21.72%)
$CN_{281}^{13,22}$	1	0.935	25(8.89%)	0.935 (0.00%)	39(0.05%)	0.49(19.06%)
$CN_{281}^{13,22}$	5	0.809	27(9.60%)	0.826(2.10%)	23(0.10%)	0.58(24.68%)
$CN_{136}^{17,25}$	1	0.976	20(14.7%)	0.978(0.20%)	34(0.00%)	0.25(27.43%)

4.4.4 DPC/R-2 on Grid Networks

To further evaluate the performance of DPC/R-2, we use it for the five grid networks described in Section 2.6.1. For each network, we consider five different values of R_{min}, *i.e.*, 50%, 60%, 70%, 80%, and 99% of the reliability of the original network, Rel_{max}; *e.g.*, for $G_{\geq 3.557^{18}}^{36,57}$ with Rel_{max}=0.9173, we set R_{min} to 0.46, 0.55, 0.64, 0.73, and 0.9, respectively. We used MCS [10] to compute Rel_{max} for all grid networks; we used a sample size of 10^6 in the simulation. Note that the Rel_{max} for G_{3x12}, G_{2x20}, and $Grid_{2x100}$ given in [50] are exactly the same as those that we generated using MCS.

To see how fast, in terms of *k*, the DPC/R-2 algorithm produces the results for each of the five grid networks, we set R_{min} to Rel_{max}; this is the worst-case scenario because the resulting topology includes all links in the original network. As shown in Table 4.9, the DPC/R-2 algorithm with OC1 produces each topology using only up to k_{max}=1214 spanning trees, and thus between $248/2^{20}=2.3^{-4}$ and $1214/1.899^{102}=4.7^{-26}$ fraction of the spanning trees in the networks. DPC/R-2 is also very fast in producing the results for the networks with the other R_{min} values; see column *k*. Note that DPC/R-

2 requires k_{max} spanning trees to produce Rel_{max}, and thus k_{max} is the upper bound value of k.

Table 4.9 also shows the reliability value of each generated topology. However, we are unable to gauge the optimality of the generated topologies for such large networks, except for Cost(G)-1. In this case, for each network, we deleted one link from the network, and used MCS to generate its $Rel_{all}(G)$; we repeated the step |E|-1 times, and selected the minimum cost with $Rel_{all}(G) \geq R_{min}$ as the best solution. As shown in Table 4.9 (the numbers in bold in column $Rel_{all}(G)$), the DPC/R-2 algorithm is able to generate a topology with a reliability only up to 5.05% off from the best solutions and requires only up to 27.11 CPU seconds.

To evaluate the time efficiency of DPC/R-2 with Rel_2 measure, we use OC1 and OC6 criteria, and run the algorithm on grid network $Grid_{3x12}$ that has 57 links and total cost of 57; see Fig. 4.3 (a). For R_{min}=0.7, DPC/R-2 produces the topology in Fig. 4.3 (b) with minimum cost of 52 and $Rel_2(G)$=0.7253 using only k=45 paths, *i.e.*, 45/1262818=3.5^{-5} fractions of the paths contained in the networks.

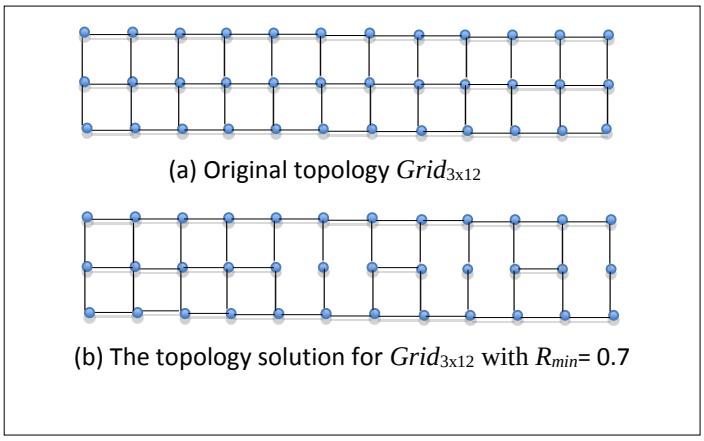


Fig. 4.3: An example grid network $Grid_{3x12}$ and topology solution

Table 4.9: Performance DPC/R-2 with spanning trees for Grid network results

CN	R_{min}	*k*	Cost	$Rel_{all}(G)$	CPU
$Grid_{3x12}$ $G^{36,57}_{\geq 3.557^{18}}$ k_{max}=145 Rel_{max}=0.9173	0.46	137	44	0.4913	4.34
	0.55	137	45	0.5703	4.11
	0.64	139	46	0.6590	4.57
	0.73	141	48	0.7343	4.89
$Grid_{6x6}$ $G^{36,60}_{\geq 3.557^{18}}$ k_{max}=194 Rel_{max}=0.9130	0.45	132	46	0.4670	4.51
	0.54	144	48	0.6016	4.67
	0.63	159	49	0.6863	4.72
	0.72	165	50	0.7273	4.83
	0.89	194	59	**0.9082(-0.53%)**	5.12
$Grid_{2x20}$ $G^{40,58}_{2^{20}}$ k_{max}=248 Rel_{max}=0.7452	0.37	188	28	0.3910	6.25
	0.44	209	34	0.4574	7.09
	0.52	214	37	0.5362	7.42
	0.59	236	39	0.6091	8.51
	0.73	248	57	**0.7405(-0.63%)**	8.76
$Grid_{3x16}$ $G^{48,77}_{\geq 2.5^{24}}$ k_{max}=187 Rel_{max}=0.7218	0.36	166	52	0.2712	4.45
	0.43	168	58	0.4394	4.46
	0.50	175	59	0.5021	4.52
	0.58	180	62	0.5877	5.66
	0.71	187	76	**0.7180(-0.53%)**	5.72
$Grid_{2x100}$ $G^{200,298}_{1.899^{102}}$ k_{max}=1214 Rel_{max}=0.251	0.12	742	184	0.1250	21.65
	0.14	876	195	0.1472	22.94
	0.17	898	224	0.1746	23.89
	0.19	1056	268	0.1951	25.90
	0.23	1214	297	**0.2383(-5.05%)**	27.11

Further, DPC/R-2 requires only between 127 and 1318 paths to generate results for the other four networks. These results further show the applicability of DPC/R-2 on networks containing large number of (*s*, *t*) paths.

4.5 Chapter Summary

We have formally defined a NP-hard NTD problem, called NTD-C/R, to generate a topology that has the minimum cost subject to reliability constraint R_{min}. The problem considers both Rel_{all} and Rel_2. We have proposed a heuristic DPA, called DPC/R, to solve NTD-C/R from a given set of spanning trees and (*s*, *t*) paths for Rel_{all} and Rel_2 measures respectively. The first version of the method, DPC/R-1, requires all spanning trees or (*s*, *t*) paths as its input, while the second version, DPC/R-2, incrementally

generates only a selected *k* spanning trees or (*s*, *t*) paths from the network, and thus is scalable on networks with large numbers of spanning trees or paths.

We have proposed to sort the spanning trees or (*s*, *t*) paths using six different order criteria, *i.e.*, OC1 to OC6, to optimize our method's effectiveness and efficiency. Our extensive simulations show that OC1 and OC6 are the best criteria for Rel_{all} and Rel_2 measures, respectively. Our simulations on various networks that contain up to 200 nodes, 298 links, 1.899^{102} spanning trees and 2^{99} (*s*, *t*) paths show the practicality of our techniques and the superiority of our efficient DP approach as compared to the existing four state-of-the-arts solutions. The experimental study shows that DPC/R is able to generate 81.5% and 91% best solutions for Rel_{all} and Rel_2.

Next, Chapter 5 will discuss two related bi-objective problems, NTD-CR/B and NTD-CB/R, and describe how to use Lagrange Relaxation techniques to convert each of the problems into its corresponding O_1/C_1 problem. The chapter will also describe two DPA, called DPCR/B and DPCB/R, to solve the problems.

Chapter 5

Network Topology Design for Bi-Objective and One Constraint

This chapter addresses two bi-objective network topology design (NTD) problems. The first NTD problem, NTD-CR/B, aims to minimize network cost (C) and maximize reliability (R) subject to satisfying a required operational bandwidth (B), B_{min}. The problem extends the NTD-R/C problem of Chapter 3 by including the bandwidth metric. In contrast, the second problem, NTD-CB/R, an extension of the NTD-C/R in Chapter 4, aims to minimize cost and maximize bandwidth while satisfying a required operational reliability, R_{min}. For both problems, this chapter considers Rel_2 reliability measure. Both NTD problems are NP-hard [8] since computing network reliability in general is NP-hard [9]. Therefore this chapter proposes two heuristic solutions to solve the problems, *i.e.*, DPCR/B for NTD-CR/B and DPCB/R for NTD-CB/R. Both solutions use the Lagrange Relaxation technique ([51], [52]) and weighted sum method to convert the bi-objectives in both problems into their corresponding single objective problems. Then, each solution uses a heuristic DPA to solve the problem. The chapter also describes the extensions of DPCR/B and DPCB/R, called DPCR/B-P and DPCB/R-P respectively that utilize the concept of Pareto Optimal Set (POS) of non-dominated solutions to improve their effectiveness.

The layout of this chapter is as follows. Section 5.1 formulates NTD-CR/B and NTD-CB/R problems. Section 5.2 presents the quality index that is used to rank the goodness of all feasible solutions of each of the problems. Section 5.3 describes Lagrange

Relaxation technique to convert BO into SO. Section 5.4 and Section 5.5 describe two approaches, called DPCR/B and DPCB/R, with their illustrating examples and time complexities to solve NTD-CR/B and NTD-CB/R respectively. Section 5.6 extends the DPCR/B and DPCB/R to DPCR/B-P and DPCB/R-P respectively to generate the POS of non-dominated solutions. Finally, Section 5.7 presents the simulation results and Section 5.8 concludes the chapter.

5.1 Problem Statement

NTD-CR/B problem extends NTD-R/C of Chapter 3 by including bandwidth constraint and cost as the second objective. In contrast, NTD-CB/R problem extends NTD-C/R of Chapter 4 by inserting bandwidth parameter as the second objective. Specifically, given a set of computer centres, their connecting links, link failure rate, bandwidth and installation cost, NTD-CR/B selects the most suitable set of links such that its resulting topology meets a required bandwidth constraint B_{min} while minimizing its cost and maximizing its reliability. On the other hand, NTD-CB/R selects the most suitable set of links such that the resulting topology meets a required reliability constraint R_{min} while minimizing its cost and maximizing its bandwidth. Both NTD problems are NP-hard since computing network reliability in general is NP-hard [9]. The following sections formally define both problems and propose heuristic solutions to solve them.

5.1.1 NTD-CR/B Problem

Let $X_i \in \{0, 1\}$ be a decision 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 Eq. (5.1) and Eq. (5.2) define

the Network Topology Design with Cost and Reliability objectives subject to a Bandwidth constraint (NTD-CR/B) problem:

$$\text{Min } (\text{Cost}(\bigcup_{i=1}^{|P_G|} P_i\ X_i)) \text{ and Max } (\text{Rel}_2(\bigcup_{i=1}^{|P_G|} P_i\ X_i)) \quad (5.1)$$

$$\text{Subject to } \text{BW}(\bigcup_{i=1}^{|P_G|} P_i\ X_i) \geq B_{min} \quad (5.2)$$

Equation (5.1) aims to minimize the cost and maximize the reliability of the topology induced by the (*s*, *t*) paths selected in Eq. (5.2) 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. (5.1) and Eq. (5.2). Note that we call each topology that satisfies Eq. (5.2) a *feasible* solution.

5.1.2 NTD-CB/R Problem

Let $Y_i \in \{0, 1\}$ be a decision variable that indicates if path P_i in G=(V, E) is selected (Y_i=1) or is not selected (Y_i=0). The following equations (5.3) and (5.4) state the NTD-CB/R problem.

$$\text{Min } (\text{Cost}(\bigcup_{i=1}^{|P_G|} P_i\ Y_i)) \text{ and Max } (\text{BW}(\bigcup_{i=1}^{|P_G|} P_i\ Y_i)) \quad (5.3)$$

$$\text{Subject to } \text{Rel}_2(\bigcup_{i=1}^{|P_G|} P_i\ Y_i) \geq R_{min} \quad (5.4)$$

Eq. (5.3) aims to minimize the cost and maximize the bandwidth of the topology formed by the paths selected in Eq. (5.4) while its reliability is at least R_{min}. Specifically, the goal of NTD-CB/R is to remove some links in G such that the remaining topology G_{best} satisfies Eq. (5.4). Note that we call each topology that satisfies Eq. (5.4) a *feasible* solution.

5.2 Quality Index for Feasible Solutions

The NTD-CR/B and NTD-CB/R are bi-objective optimization problems that aim to minimize one objective while maximizing the other; this thesis calls the problems Min-Max bi-objective. However, in general, a feasible solution for such problem may only minimize one objective ***or*** maximize the other. For example, consider NTD-CR/B and the CN in Fig. 2.2 with B_{min}=8. We obtain two feasible solutions G_1 and G_2 with {Cost(G_1)=25, $Rel_2(G_1)$=0.878, BW(G_1)=11}, and {Cost(G_2)=20, $Rel_2(G_2)$=0.833, and BW(G_2)=8}, respectively. Notice that neither feasible solution satisfies the BO in Eq. (5.1) because Cost(G_2)<Cost(G_1) but $Rel_2(G_2)<Rel_2(G_1)$. Consequently, one needs to define a measure or quality index to determine the *best* among the feasible solutions for the Min-Max bi-objective optimization problems.

For NTD-CR/B, we define a quality index $rc_r=Rel_2(G_r)/Cost(G_r)$, and consider a feasible topology G_i *better* than G_j if (i) $rc_i>rc_j$, or (ii) $rc_i=rc_j$ and $BW(G_i)\geq BW(G_j)$. For the previous example, G_2 is better than G_1 because $rc_2=Rel_2(G_2)/Cost(G_2)=0.833/20=0.041>rc_1=0.878/25=0.035$. Similarly, for NTD-CB/R, we use another index $bc_r=BW(G_r)/Cost(G_r)$, and consider a feasible topology G_i *better* than G_j if (i) $bc_i>bc_j$, or (ii) $bc_i=bc_j$ and $Rel_2(G_i)\geq Rel_2(G_j)$.

Using the rc_r index, we have the following alternative NTD-CR/B problem:

$$\text{Max } (Rel_2(\bigcup_{i=1}^{|P_G|} P_i\ X_i)/Cost(\bigcup_{i=1}^{|P_G|} P_i\ X_i)) \quad (5.5)$$

$$\text{Subject to } BW(\bigcup_{i=1}^{|P_G|} P_i\ X_i)\geq B_{min} \quad (5.6)$$

Similarly, considering bc_r ratio, we can re-state the NTD-CB/R problem as:

$$\text{Max } (BW(\bigcup_{i=1}^{|P_G|} P_i\ X_i)/Cost(\bigcup_{i=1}^{|P_G|} P_i\ Y_i)) \quad (5.7)$$

$$\text{Subject to } \text{Rel}_2\left(\bigcup_{i=1}^{|P_G|} P_i \; Y_i\right) \geq R_{min} \tag{5.8}$$

Notice that when both cost and reliability in Eq. (5.1) are minimized and maximized, respectively, Eq. (5.5) is maximized. Similarly, Eq. (5.7) is maximized when both cost and bandwidth in Eq. (5.3) are minimized and maximized, respectively.

The Multiple Objective Integer Knapsack Problem (MOIKP) approach proposed by Figueira *et al.* [42], described in Section 2.5.2, cannot be used to solve NTD-CR/B and NTD-CB/R for three main reasons. First, MOIKP aims to maximize all objective functions, while our problems maximize one objective and minimize the other, *i.e.*, Min-Max objective. Using DPA used by the authors in [42] to solve either NTD-CR/B or NTD-CB/R can produce infeasible non-dominated solutions. Second, both objectives in [42] can be solved using simple additive functions, in contrast to the significantly more complex functions used to compute the objectives in our problems, *i.e.*, $\text{Cost}(G_i)$, $\text{BW}(G_i)$, and $\text{Rel}_2(G_i)$. Finally, in MOIKP, the total cost of two items is the sum of each item's cost, while in NTD-CR/B and NTD-CB/R, similar to in NTD-R/C and NTD-C/R described in Section 3.3.3.1 and 4.2, $\text{Cost}(P_i)+\text{Cost}(P_j) \geq \text{Cost}(P_i \cup P_j)$ and $\text{Rel}_2(P_i)+\text{Rel}_2(P_p) \geq \text{Rel}_2(P_i \cup P_p)$.

5.3 Converting BO into SO Problem

The following two subsections explain two methods that use the Lagrange-relaxation and weighted sum technique to convert the bi-objective problems, NTD-CR/B and NTD-CB/R, into their corresponding O_1/C_1 problems.

5.3.1 SO Problem for NTD-CR/B

We use the Lagrange Relaxation method [51], [52] to combine the objectives Cost(G) and $Rel_2(G)$ and the constraint BW(G) for NTD-CR/B problem into a weight Ψ(G) as:

$$\Psi(G)=Cost(G)\times Rel_2(G)+\eta\times BW(G) \quad (5.9)$$

The Lagrange relaxation procedure uses the idea of relaxing the explicit constraint, *i.e.*, bandwidth, by bringing it into the objective function with an associated vector η, called Lagrange multiplier. The value of η should be set properly to minimize Cost(G) and maximize $Rel_2(G)$ in Eq. (5.1). For our problem, we set η as

$$\eta=C_{min}\times R_{max}/B_{min} \quad (5.10)$$

where C_{min} and R_{max} are the minimum cost and maximum reliability of all possible paths in the network respectively. Using Eq. (5.9) and Eq. (5.10) we convert BO optimization in Eq. (5.1) into the following SO optimization:

$$Max(\Psi(G)=Cost(\cup_{i=1}^{|P_G|} P_i X_i)\times Rel_2(\cup_{i=1}^{|P_G|} P_i X_i)+\eta\times BW(\cup_{i=1}^{|P_G|} P_i X_i)) \quad (5.11)$$

To achieve the objective in Eq. (5.11), we first combine the cost c_j, reliability r_j and bandwidth b_j of each link e_j into a weight w_j, rather than considering them separately. Here, we use the Lagrange Relaxation technique [52] 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 all possible links in the network. Note that the computation of each w_j and η_w corresponds to Eq. (5.10) and Eq. (5.11), respectively. Then, we generate all (*s*, *t*) paths of G in increasing order of their weights; the path weight is calculated as the summation of the weight of each link in the path. We refer to such order criterion as

OC7, shown in Table 2.6 in Section 2.6.2. Our simulations in Section 5.7.1 show the benefits of using OC7 in producing near optimal topologies.

5.3.2 SO Problem for NTD-CB/R

For NTD-CB/R problem, the Lagrange Relaxation method (Fisher 2004; Loh *et al*. 2009) combine objectives Cost(G) and BW(G) and constraint $Rel_2(G)$ of NTD-CR/B into a weight $\Psi(G)$ as:

$$\Psi(G)=Cost(G)\times BW(G)+\eta\times Rel_2(G) \quad (5.12)$$

The value of η should be set properly to minimize Cost(G) and maximize BW(G) in Eq. (5.3). For our problem, we set η as

$$\eta=C_{min}\times B_{max}/R_{min} \quad (5.13)$$

where C_{min} and B_{max} are the minimum cost and maximum bandwidth of all possible paths in the network respectively. Using Eq. (5.12) and Eq. (5.13) we convert BO optimization in Eq. (5.3) into the SO optimization:

$$Max(\Psi(G)=Cost(\bigcup_{i=1}^{|P_G|} P_i X_i)\times Rel_2(\bigcup_{i=1}^{|P_G|} P_i X_i)+\eta\times BW(\bigcup_{i=1}^{|P_G|} P_i X_i)) \quad (5.14)$$

Similar to NTD-CR/B, to achieve the objective in Eq. (5.14), we first combine the cost c_j, reliability r_j and bandwidth b_j of each link e_j into a weight w_j, rather than considering them separately. We use the Lagrange Relaxation technique [52] to compute $w_j=c_j\times b_j+\eta_w\times r_j$, where $\eta_w=c_{min}\times b_{max}/R_{min}$ and c_{min} (b_{max}) is the minimum cost (maximum bandwidth) of all possible links in the network. The computation of each w_j and η_w corresponds to Eq. (5.12) and Eq. (5.13), respectively. Then, we generate all (s, t) paths of G in increasing order of their weights; the path weight is calculated as the summation of the weight of each link in the path. We call such order criterion OC8,

shown in Table 2.6 in Section 2.6.2. Our simulations in Section 5.7.1 show the benefits of using OC8 in producing near optimal topologies.

5.3.3 Solving the SO Problems

For NTD-CR/B, one may generate all 2^m possible combinations of paths, and for each combination that has bandwidth at least B_{min}, use Eq. (5.11) and, thus Eq. (5.1), to select one with the best value as G_{best}; similar brute-force approach applies for NTD-CR/B. However, the brute-force solution is prohibitive for use in large networks since in general a network contains $m=2^{|E|-|V|+2}$ (s, t) simple paths [21]. Therefore, this thesis proposes heuristic techniques to solve the two difficult problems.

We observe that after conversion, the single objective version of NTD-CR/B is in essence the same as the O_1/C_1 problem in Chapter 3, *i.e.*, NTD-R/C. Note that, in the SO version, NTD-CR/B aims to maximize the weight $\Psi(G)$ in Eq. (5.11), while NTD-R/C problem described in Section 3.2 attempts to maximize reliability. However, the two problems use different constraints, *i.e.*, the former has bandwidth constraint while the latter uses cost constraint. We also observe that the SO version of NTD-CB/R problem is similar to the NTD-C/R problem described in Section 4.2, *i.e.*, both problems use the same reliability constraint. However, the former problem aims to maximize weight $\Psi(G)$ in Eq. (5.14) while the latter's objective is to minimize cost in Eq. (4.3). In Section 5.4, this thesis describes DPCR/B, the solution to NTD-CR/B, which utilizes similar concepts used in DPR/C for NTD-R/C of Chapter 3. Further, the solution to NTD-CR/B, called DPCB/R, in Section 5.5 uses similar concepts to the solution DPC/R for NTD-C/R in Chapter 4. However, for clarity, the sections describe the two solutions completely, ignoring some possible repetitive explanations.

5.3.4 Order Criteria

Similar to DPR/C and DPC/R algorithms, described in Chapter 3 and 4 respectively, the performance of DPCR/B and DPCB/R are affected by the order of their inputs, *i.e.*, sequence of (*s*, *t*) paths. One can use the six order criteria, OC1 to OC6 (described in Section 3.4.3 and Appendix B) for both DPCR/B and DPCB/R. Further, we have proposed OC7 and OC8 in Section 5.3.1 and Section 5.3.2 for DPCR/B and DPCB/R respectively; recall that for OC7, $w_j=c_j\times r_j+\eta_w\times b_j$ while for OC8, $w_j=c_j\times b_j+\eta_w\times r_j$. For DPCR/B that has bandwidth constraint, we propose another order criteria, OC9, which sets $w_j=b_j$ for each link e_j. Finally, for DPCB/R, we propose OC10 that computes $w_j=c_j/b_j$ for each link e_j. Section 5.7.1 compares the effects of using the 10 order criteria on the performance of both DPCR/B and DPCB/R.

5.4 Solution for NTD-CR/B

This section describes our DP formulation to solve the problem. Next, it presents DPCR/B algorithm followed by an illustrating example. Further, it shows the time complexity of DPCR/B.

5.4.1 DP Formulation

Let PX_i, for i=1,2, …,m, be a sequence of paths selected from m paths in $\{P_1,P_2,\ldots,P_{m-1},P_m\}$ and $G_i=(V,E_i\subseteq E)$ be its induced graph whose links comprise all links in PX_i. There are 2^m different PX_i, for $1\leq|PX_i|\leq m$, and we aim to select PX_m with a bandwidth $BW(G_m)\geq B_{min}$ and the maximum $\Psi(G_m)$; the latter goal aims to maximize the goal in Eq. (5.11) and hence minimizes $Cost(G_m)$ and maximizes $Rel_2(G_m)$ in Eq. (5.1).

Let DP[1.. m, 0.. B_{min}] be a 2-dimension array in which each DP[i, b], for i=1, 2, …, m, b=0, 1, 2, …, B_{min}, stores seven pieces of information: a cost C[i, b]≥0, a reliability

$0 \leq R[i, b] \leq 1.0$, a bandwidth $0 \leq B[i, b] \leq B_{min}$, a weight $W[i, b] \geq 0$, a set of (s, t) paths $X[i, b] \subseteq P_G$, a set of links $L[i, b] \subseteq E$, and an integer index $0 \leq J[i, b] \leq B_{min}$. In essence, each b^{th} column corresponds to a bandwidth constraint $b=0, 1, \ldots, B_{min}$. Each $DP[i, b]$ is used to store the information of each selected topology G_i that has $BW(G_i) \geq b$. Specifically, for each $BW(G_i) \geq b$, $W[i, b]=\Psi(PX_i)$, $X[i, b]=PX_i$. Since $W[m, B_{min}]$ is the weight of $G_m=(V, E_m \subseteq E)$ with $BW(G_m) \geq B_{min}$, NTD-CR/B aims to generate $DP[m, B_{min}]$ with maximum $W[m, B_{min}]$, which represents the G_{best}. Note that $W[m,B_{min}]=\Psi(G_m)$ is calculated using Eq. (5.9) and Eq. (5.10).

Each $W[i, b]$, for $b=0, 1, 2, \ldots, B_{min}$, is calculated using a DP formulation defined using the following four equations:

$$W[i, b]=\Psi(P_i), \text{ if } i=1 \text{ and } BW(P_i) \geq b \quad (5.15)$$

$$W[i, b]=0, \text{ if } i=1 \text{ and } BW(P_i)<b \quad (5.16)$$

$$W[i, b]=\text{Max}(W[i\text{-}1, b], \Psi(P_i)), \text{ if } i>1 \text{ and } BW(P_i) \geq b \quad (5.17)$$

$$W[i, b]=\text{Max}(W[i\text{-}1, b], \Psi(X[i\text{-}1, j] \cup P_i)), \text{ if } i>1 \text{ and } BW(X[i\text{-}1, j] \cup P_i) \geq b \quad (5.18)$$

We explain the DP formulation in Eq. (5.15) to Eq. (5.18) as follows; 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. (5.15) and Eq. (5.16) are used for the first path; the path is selected if it has bandwidth of at least b, giving $W[1, b]=\Psi(P_1)$. In contrast, when $BW(P_1)<b$, P_1 is not selected because it does not meet the bandwidth constraint b; thus Eq. (5.16) sets $W[1, b]=0$ to denote that no path is selected. Eq. (5.17) and Eq. (5.18) are used for each remaining P_i, for $i=2, 3, \ldots, m$. Eq. (5.17) 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.*, $W[i\text{-}1, b]$. In other words, Eq. (5.17) 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.

Eq. (5.18) 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.*, $\Psi(X[i\text{-}1, j] \cup P_i)>W[i\text{-}1, b]$, while satisfying the required bandwidth *b*, *i.e.*, $BW(X[i\text{-}1, j] \cup P_i)\geq b$. This step considers each possible column $j=J[i\text{-}1, b]=0, 1, \ldots, B_{min}$. Note that Eq. (5.17) and Eq. (5.18) 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 $C[i, b]$, $R[i, b]$, $B[i, b]$, $X[i, b]$, $L[i, b]$, 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. (5.15) to Eq. (5.18) is similar to the DP solution for 0/1 knapsack problem [43] as described in Section 2.5.2. However, unlike for 0/1 knapsack problem 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.*, from Table 2.2, P_3=(2,5,4,3,7) and P_7=(1,4,6,7) that share link 4 and 7 have $\Psi(P_3)$=19.8 and $\Psi(P_7)$=17.4 respectively, and $\Psi(P_3)+\Psi(P_7)=37.2\geq\Psi(P_3 \cup P_7)=20.34$. Therefore Eq. (5.18) considers all possible values of *j*, *i.e.*, $j=J[i, b]$, unlike its equivalent step for 0/1 knapsack problem [43].

5.4.2 DPCR/B Algorithm

One can design an algorithm, called DPCR/B, to solve NTD-CR/B by directly implementing the formulation in Eq. (5.15) to Eq. (5.18). However, the algorithm requires all m paths of the network, which is not feasible for networks with large m; we call this version DPCR/B-1. Alternatively, we use a greedy heuristic, called DPCR/B-2, that requires only a small number of $1 \le k \le m$ paths to improve DPCR/B's time complexity while producing almost the same result as compared to using all paths in the network. Since there is no effective way for setting the value of k a priori, we set k dynamically following the method described in Section 3.3.5 and Section 4.3.5. In this case, our algorithm generates the next path P_{i+1} with the largest weight, and if it obtains a feasible solution G_{i+1} with higher value as compared to G_i, *i.e.*, $\Psi(G_{i+1}) > \Psi(G_i)$, it keeps generating the subsequent paths. The algorithm stops when it generates 10 consecutive paths with none of which can further improve the weight value, *i.e.*, 0.5% improvement. The detail of DPCR/B-2 is as follows.

DPCR/B-2 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}, …, P_k, …, P_1)
3. Set W[1, b]=0, for b>BW(P_1) // Eq. (5.16)
4. Set W[1, b]=Ψ(P_1), for $0 \le b \le$ BW(P_1) // Eq. (5.15)
5. *Copy row*1 *to row*2 and *initialize* q=0, i=2
6. **While** ($i \le k$+10) **do** // Eq. (5.17) and (5.18)
7. **if** ($q \neq 10$)
8. W[2, b]←Max(W[1,b], Ψ(P_i)),$0 \le b \le$ BW(P_i))
9. **for** ($y \leftarrow 0$ **to** B_{min}**) do** // Eq. (5.18)
10. **if (**J[1, y]$\neq J$[1, y+1])
11. $j = J$[1, y], d=BW(P[1, j] $\cup$ P_i)
12. W[2, d] ← Max(W[1, d], Ψ(X[1, j] $\cup$ P_i))
13. **if** (W[2, B_{min}]<1.005×W[1, B_{min}])
14. q++

15. **else**
16. q=0
17. *Copy row*2 *to row*1, *i*++, *goto step* 6
18. **else**
19. **Return**

In Step 1, DPCR/B-2 calculates the weight w_j for each link e_j. Note that $\eta_w=c_{min}\times r_{max}/B_{min}$, and c_{min} (r_{max}) is the minimum cost (maximum reliability) of all possible links in the network. Ideally, Step 2 should generate the first k+10 largest-weight paths, starting from the largest, the second largest, *etc*., which can be iteratively used in Step 6. Unfortunately, generating path with the maximum weight is equivalent to generating the longest path [26], which has been shown NP-complete. Therefore, DPCR/B-2 sets k=10000, and uses Yen's algorithm [46] to generate a sequence of paths (P_{k+10},P_{k+9},…,P_k,P_{k-1},…,P_2,P_1), in order of increasing weights, *i.e.*, P_1 has the largest weight. Specifically, our implementation generates the first k=10000 smallest-weight paths, which is ideal for networks that have less than 10000 paths. Step 3 to 4 implement Eq. (5.15) and Eq. (5.16) using P_1. Note that DPCR/B-2 requires only two rows of DP table, called *row*1 and *row*2. The algorithm repeatedly executes Step 6 to 19 until it generates 10 consecutive paths with insignificant improvement in W[i, B_{min}], *i.e*., 0.5%. 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-2 also updates its other corresponding information, *e.g*., C[i, b]. Step 8 implements Eq. (5.17), and Step 9 to12 implement Eq. (5.18). Step 13 to 16 are used to stop the iteration, *i.e.*, when $\Psi(G_{k+10})<1.005\times\Psi(G_k)$. One can modify DPCR/B-2 into DPCR/B-1 by generating all (s, t) paths in Step 2 and deleting Step 6, 7 and 13-16.

5.4.3 Illustrating Example

Consider Fig. 5.1 with B_{min}=8 (a given constraint), c_{min}=2, r_{max}=0.9 and η_w=0.225. Our DPCR/B-1 constructs the DP table in Table 5.1 to obtain G_{best}; for convenience, the table shows m=7 rows. Each row of the table considers possible selection of each P_i, and each column represents bandwidth constraint from 0 to B_{min}. To save space the table shows only the values of X[i, b] and W[i, b]. Step 1 in DPCR/B-1 with order criterion OC7, described in Section 5.3.1, obtains w_1=6.3, w_2=2.47, w_3=2.93, w_4=4.87, w_5=6.42, w_6=3.3, w_7=2.93 and w_8=3.9. Step 2 uses Yen's algorithm [46] 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 $\Psi(P_1)$=19.8 and $BW(P_1)$=3. Step 3 initializes C[1, b]=∞, R[1, b]=0, B[1, b]=0, X[1, b]=( ), L[1, b]=( ) and W[1, b]=0, for b=4,…,8, and thus J[1, b]=B_{min}=8. Step 4-5 set C[1, b]=17, R[1, b]=0.24, B[1, b]=3, X[1, b]=(P_1), L[1, b]=(2,5,4,3,7) and W[1, b]=$\Psi(P_1)$=19.8, for b=0,…,3 with j=J[1,b]=3. Next in Step 8 and 9, DPCR/B-1 selects P_2=(1,4,6,7) with $\Psi(P_2)$=17.4 and $BW(P_2)$=4, because i=2<k=7 and q=0<10. Step 10 and 11 use Eq. (5.17) to update W[2, b]=$\Psi(P_2)$ only for b=4 because $\Psi(P_2)$>W[1, 4]=0. Step 12-16 consider j starting from b=0 in *row*1 with two values: j=J[1, b]=3 and j=J[1, b]=8. For j=3, $\Psi(X[1, j] \cup P_2)$=20.34>W[1, b]; thus Eq. (5.18) 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, X[1, b]=(P_1, P_2), L[1, b]=(1,2,3,4,5,6,7) and W[1, b]=20.344 for b=5 and 6. For j=8, Eq. (5.18) selects P_2 ($P_2 \cup$ ( )) at b=4. Repeating the steps for P_3 to P_7, DPCR/B-1 obtains best topology (Fig. 5.1) with X[7, B_{min}=8]=(P_2,P_3,P_5,P_7) with 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]=23.018.

Table 5.1: DP Table for CN with B_{min}=8 in Fig. 2.2

b / p_i	0	1	2	3	4	5	6	7	8
P_1	X[1, b]=(1) W[1, b]=19.8	(1) 19.8	(1) 19.8	(1) 19.8	( ) 0	( ) 0	( ) 0	( ) 0	( ) 0
P_2	(1) 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2) 20.3	(1,2) 20.3	( ) 0	( ) 0
P_3	(1) 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2) 20.3	(1,2) 20.3	(2,3) 23.018	(2,3) 23.018
P_4	(1) 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2,4) 20.3	(1,2,4) 20.3	(2,3) 23.018	(2,3) 23.018
P_5	(1) 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2,4) 20.3	(1,2,4) 20.3	(2,3,5) 23.018	(2,3,5) 23.018
P_6	1 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2,4) 20.3	(1,2,4) 20.3	(2,3,5) 23.018	(2,3,5) 23.018
P_7	1 19.8	(1) 19.8	(1) 19.8	(1) 19.8	(2) 17.4	(1,2,4,7) 20.3	(1,2,4,7) 20.3	(2,3,5,7) 23.018	(2,3,5,7) 23.018

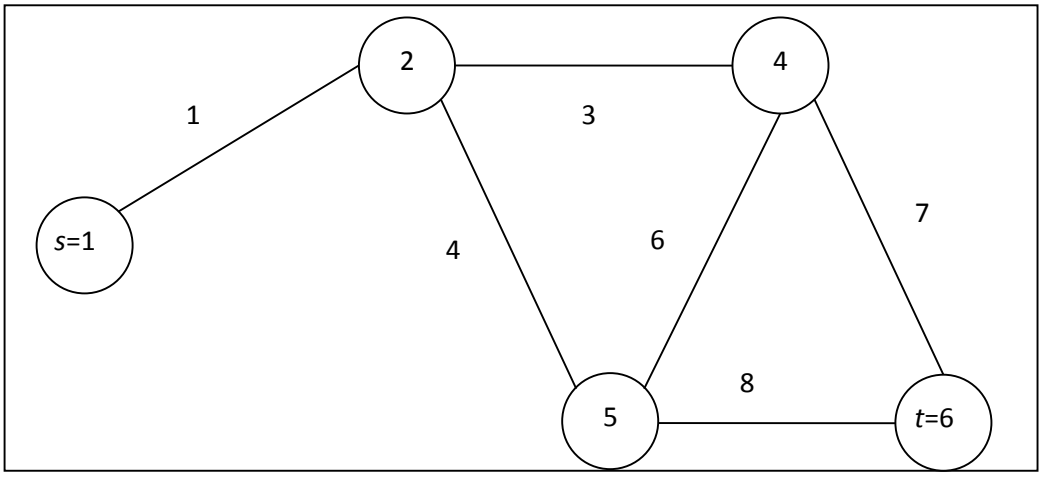


Fig. 5.1: The best solution of NTD-CR/B with B_{min}=8 for CN in Fig. 2.2

5.4.4 Time Complexity

Steps 1 requires $O(1)$, and Step 2 uses Yen's algorithm that 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 • that contains (s, t) simple paths, function BW(•) can be implemented using Max-Flow Min-Cut algorithm [53] with time $O(|V|\times|E|)$. Since BW(•) is used only for each different j in each row i, the total time of using BW(•) is $O(\Omega\times|V|\times|E|)$, where $\Omega\leq k\times B_{min}$ is the total number of different j in the table (Step 11). The Cost(•) includes

every unique link in •, 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 [22], 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(•) is computed in $O(|E|)$. DPCR/B-2 uses the function at most once for every table entry. Thus, the worst case time for using Cost(•) is $O(\Omega \times |E|)$. We use the CAREL with time complexity of $O(\Omega \times |E| \times T_{max})$ [22], see Section 3.3.1; to compute exact $Rel_2(\bullet)$ for each candidate network. In Step 12, the Dijkstra's algorithm takes $O(|V|^2 \times \log|V|)$ time to find C_{min} and R_{max}. Also, we use Cost(X), $Rel_2(X)$ and BW(X) to calculate Ψ(X) in Step 12; see Eq. (5.18). Note that Cost(X), $Rel_2(X)$ and Ψ(X) are used only when BW(X)≥*b*. Thus, DPCR/B-2 has a time complexity of $O(1+(k \times |V| \times (|E|+|V| \times \log|V|))+k \times B_{min}+|E|+|E|+(\Omega \times (|V| \times |E|+|E|+|E| \times T_{max}+1))+(|V|^2 \times \log|V|))=O(\Omega \times T_{max} \times (|V|^2))$, since *k*, B_{min} are constants and $|E| \leq |V|^2$.

5.5 Solution for NTD-CB/R

This section describes our DP formulation to solve the NTD-CB/R problem. Next, it presents DPCB/R algorithm followed by an illustrating example and time complexity of DPCB/R.

5.5.1 DP Formulation

Let DP[1.. *m*, 0.. $\check{R}_{min}$] be a 2-dimension dynamic programming table. Each element DP[*i*, *ř*], for *i*=1, 2, …, *m*, *ř*=0, 1, 2, …, $\check{R}_{min}$, stores seven pieces of information: a cost C[*i*, *ř*]≥0, a bandwidth B[*i*, *ř*]≥0, a reliability 0≤R[*i*, *ř*]≤1.0, a weight W[*i*, *ř*]≥0, a set of paths X[*i*, *ř*]⊆P_G, a set of links L[*i*, *ř*]⊆E, and an integer index 0≤*J*[*i*, *ř*]≤σ. Each DP[*i*, *ř*] is used to store seven pieces of information for each selected topology G_i that has $Rel_2(G_i) \geq r$. Specifically, for each $Rel_2(G_i) \geq r$, we set C[*i*, *ř*]=Cost(G_i), B[*i*, *ř*]=BW(G_i), R[*i*, *ř*]=$Rel_2(G_i)$, W[*i*, *ř*]=Ψ(P_i), X[*i*, *ř*]=PX_i, and L[*i*, *ř*]=E_i. For

$Rel_2(G_i)<r$, we set C[i, $\check{r}$]=0, R[i, $\check{r}$]=0, B[i, $\check{r}$]=0, W[i, $\check{r}$]=0, X[i, $\check{r}$]=(), and L[i, $\check{r}$]=(). Note that we explained $\check{r}$ and $\check{R}_{min}$ in Section 4.2. As W[m, $\check{R}_{min}$] is the weight of G_m=(V, $E_m \subseteq$E) with $Rel_2(G_m) \geq R_{min}$, NTD-CB/R aims to generate DP[m, $\check{R}_{min}$] that contains the maximum W[m, $\check{R}_{min}$], which represent the G_{best}. Notice that we used MCS [10] to estimate $Rel_2(G_m)$, discussed in Section 2.2.1. For each range of columns $\check{r}_1 \leq \check{r} \leq \check{r}_2$ in row i that contain the same reliability value, we set each j=J[i, $\check{r}$]=$\check{r}_2$. Thus, index j=J[i, $\check{r}$]=0, 1, 2, …, $\check{R}_{min}$ marks the ending column of a range of columns that have the same reliability.

Each W[i, $\check{r}$], for $\check{r}$=0, 1, 2, …, $\check{R}_{min}$, is calculated using a DP formulation defined using the following four equations:

W[i, $\check{r}$]=$\Psi(P_i)$, if i=1 and $Rel_2(P_i) \geq r$ (5.19)

W[i, $\check{r}$]=0; if i=1 and $Rel_2(P_i)<r$ (5.20)

W[i, $\check{r}$]=Max(W[i-1, $\check{r}$], $\Psi(P_i)$), if i>1 and $Rel_2(P_i) \geq r$ (5.21)

W[i, $\check{r}$]=Max(W[i-1, $\check{r}$], $\Psi(P[i\text{-}1, j] \cup P_i)$), if i>1 and $Rel_2(P[i\text{-}1, j] \cup P_i) \geq r$ (5.22)

We explain the DP formulation in Eq. (5.19)-(5.22) as follows; 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. In Eq. (5.19), when the first path has reliability of at least r, it should be selected, giving W[1, $\check{r}$]=$\Psi(P_1)$. In contrast, when $Rel_{all}(P_1)<r$, P_1 is not selected because it does not meet the reliability constraint r; thus Eq. (5.20) sets W[1, $\check{r}$]=0 to denote that no path is selected. Equations (5.21) and (5.22) are used for each remaining P_i, for i=2, 3, …, m-1, m. Eq. (5.21) considers two options, selecting or not selecting P_i when $Rel_{all}(P_i) \geq r$, and selects the option that produces the maximum weight. Specifically, when P_i is selected (not

selected), its weight is $\Psi(P_i)$ ($W[i\text{-}1, \check{r}]$), and the equation selects the maximum between the two since both options satisfy the reliability requirement r. Note that the reliability value in $R[i, \check{r}]$ would be changed to $Rel_{all}(P_i)$ if P_i is selected. Further, Eq. (5.21) considers a situation when no paths have been selected for column $\check{r}$, *i.e.*, $W[i\text{-}1, \check{r}]=0$, in which case it will select P_i. Eq. (5.22) considers the case when selecting P_i together with some previously selected paths PX_j satisfies the required reliability r, *i.e.*, $Rel_{all}(X[i\text{-}1, j]\cup P_i)\geq r$, for each possible $j=J[i\text{-}1, \check{r}]=0, 1, \ldots, \check{R}_{min}$. Like Eq. (5.21), Eq. (5.22) also considers the maximum weight between either selecting or not selecting P_i; the former obtains $\Psi(X[i\text{-}1, j]\cup P_i)$ while the latter $W[i\text{-}1, \check{r}]$. Specifically, when P_i is selected (not selected), the weight is calculated from the selected paths PX_i (PX_{i-1}). Note that the reliability value in $R[i, \check{r}]$ would be changed to $Rel_{all}(X[i\text{-}1, j]\cup P_i)$ if P_i is selected. Further, Eq. (5.22) also considers a case when no paths have been selected for column $\check{r}$, *i.e.*, $W[i\text{-}1, \check{r}]$, in which it will select P_i.

5.5.2 DPCB/R Algorithm and Time Complexity

The solution for NTD-CB/R, called DPCB/R algorithm, can directly implement the DP in Eq. (5.19) to Eq. (5.22). However, the algorithm requires all m paths of the network, which is not feasible for networks with large m; we call this version DPCR/B-1. To enhance the efficiency of DPCB/R, we use a greedy heuristic called DPCR/B-2 that requires only a small number of $1\leq k\leq m$ paths to produce almost the same result as compared to using all paths in the network. Note that DPCB/R-2 utilizes order criterion OC8, proposed in Section 5.3.2.

DPCB/R-2 Algorithm:

1. Calculate $w_j=c_j\times b_j+\eta_w\times r_j$, for each link e_j
2. Generate paths ($P_{k+10}, \ldots, P_k, \ldots, P_1$) in increasing order of weights
3. Set $W[1, \check{r}]=0$, for $r>Rel_2(P_1)$ // Eq. (5.20)

4. Set W[1, $\check{r}$]=$\Psi(P_1)$, for $0 \le r \le Rel_2(P_1)$ // Eq. (5.19)
5. *Copy row*1 *to row*2 and *initialize* q=0, i=2
6. **While** ($i \le k$+10) **do** //Eq. (5.21) and Eq. (5.22)
7. **if** ($q \neq 10$)
8. W[2, $\check{r}$]←Max(W[1, $\check{r}$], $\Psi(P_i)$), for $0 \le r \le Rel_2(P_i)$ // Eq. (5.21)
9. **for** ($y \leftarrow 0$ **to** R_{min}**) do** // Eq. (5.22)
10. **if (**$J[1,y] \neq J[1,y+1]$)
11. $j=J[1,y]$, $d=round(\sigma \times Rel_2(P[1, j] \cup P_i))$
12. W[2,d] ← Max(W[1,d], $\Psi(P[1, j] \cup P_i)$)
13. **if** (W[2, $\check{R}_{min}$]<1.005×W[1, $\check{R}_{min}$])
14. q++
15. **else**
16. q=0
17. *Copy row*2 *to row*1, i++, *goto step* 6
18. **else**
19. **Return**

The NTD-CB/R algorithm in both versions are respectively equivalent to NTD-CR/B algorithm, described in Section 5.4.2; except for (i) Step 1 uses OC8 instead of OC7, (ii) Step 3, 4 and 8 use $Rel_2(P_i)$ instead of $BW(P_i)$, and (iii) Step 9 and 13 use R_{min} instead of B_{min} constraint.

The time complexity of DPCB/R algorithm can be computed similar to that for DPCR/B. Specifically, DPCB/R-1 has time complexity of $O((\Omega \times (nr \times |V|^4 + |E| + V| \times |E| + 1)) + (|V|^2 \times \log|V| + |E| + m \times |V| \times (|E| + |V| \times \log|V|)) = O(\Omega \times m \times (|V|^4))$ nr is constant and $|E| \le |V|^2$. Further the time complexity of DPCB/R-2 is $O((\Omega \times (nr \times |V|^4 + |E| + |V| \times |E| + 1)) + (|V|^2 \times \log|V| + |E| + k \times |V| \times (|E| + |V| \times \log|V|)) = O(\Omega \times |V|^4 \times k)$**.**

5.5.3 Illustrating Example

Consider Fig. 2.2 with R_{min}=0.8 (a given constraint), c_{min}=2, c_{max}=8 and η_w=20. Our DPCB/R-1 constructs the DP table in Table 5.2 to obtain G_{best}; the table shows m=7 rows. Each row of the table considers possible selection of each P_i, and each column

represents reliability constraint from 0 to R_{min}. To save space the table shows only the values of X[i, $\check{r}$] and W[i, $\check{r}$].

Step 1 of DPCB/R-1 uses OC8 to compute all link weights; it obtains w_1=58, w_2=21, w_3=26, w_4=38, w_5=42, w_6=28, w_7=26 and w_8=33. Step 2 uses Yen's algorithm [46] to generate P_G=(P_7=(2,5,8), P_6=(1,3,7), P_5=(2,5,6,7), P_4=(1,4,8), 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 $\Psi(P_1)$=153 and $Rel_2(P_1)$=0.24. Step 3 initializes C[1, $\check{r}$]=∞, B[1, $\check{r}$]=0, R[1, $\check{r}$]=0, X[1, $\check{r}$]=( ), L[1, $\check{r}$]=( ) and W[1, $\check{r}$]=0, for $\check{r}$=25, …,80, and thus $j_=J$[1, $\check{r}$]=$\check{R}_{min}$=80. Step 4-5 sets C[1, $\check{r}$]=17, B[1, $\check{r}$]=3, R[1, $\check{r}$]=0.24, X[1, $\check{r}$]=(P_1), L[1, $\check{r}$]=(2,5,4,3,7) and W[1, $\check{r}$]=$\Psi(P_1)$=153, for $\check{r}$=0, …,24 with $j_=J$[1, $\check{r}$]=24. Next in Step 8 and 9, DPCB/R-1 selects P_2=(1,4,6,7) with $\Psi(P_2)$=150 and $Rel_2(P_2)$=0.34, because i=2<k=7 and q=0<10. Step 8 use Eq.(5.21) to update W[2, $\check{r}$]=$\Psi(P_2)$ only for $\check{r}$=25, …, 34 because $\Psi(P_2)$>W[1, $\check{r}$=25, …, 34]=0. Steps 12-16 consider j in *row*1 with two values: j=24 and j=80. For j=24, $\Psi(P[1, j] \cup P_2)$=189.3>W[1, $\check{r}$=35, …, 59]=0; thus Eq.(5.22) (Step 10 and 11) selects both P_1 and P_2 at column $\check{r}$=35, …, 59 and sets C[2, $\check{r}$]=26, B[2, $\check{r}$]=6, R[2, $\check{r}$]=0.59, X[2, $\check{r}$]=(P_1,P_2), L[2, $\check{r}$]={1,2,3,4,5,6,7}, W[1, $\check{r}$]=156.7 and $j_=J$[1, $\check{r}$]=59 for $\check{r}$=35, …, 59. For j=80, Eq. (15) selects P_2 ($P_2 \cup$ ( )) at $\check{r}$=25, …, 34. Repeating the steps for P_3 to P_7, DPCB/R-1 obtains an best topology (Fig. 5.2) with X[7, $\check{R}_{min}$=80]=(P_1,P_4,P_6,P_7) with C[7,80]=25, B[7,80]=11 and R[7,80]=0.878, L[7,80]=(1,2,3,4,5,7,8) and W[7,80]=324.3. Notice that the selected path sets, *i.e.*, (1,4), (1,4,6) and (1,4,6,7), are equivalent feasible solutions, with bandwidth of 11, cost of 25 and reliability of 0.878, because they contain the same set of links, and thus form the same G_{best}; the table shows one of the solutions, *i.e.*, X[7, $\check{R}_{min}$=80]=(P_1,P_4,P_6,P_7).

Table 5.2: DP Table for CN in Fig. 2.2 with R_{min}=0.8

p_i \ $\check{r}$	$\check{r}$=0, …, 24	$\check{r}$=25, …, 34	$\check{r}$=35, …, 59	$\check{r}$=60, …, 73	$\check{r}$=73, …, 80
P_1	X[1, $\check{r}$]=(1) W[1, $\check{r}$]=153	( ) 0	( ) 0	( ) 0	( ) 0
P_2	(1) 153	(2) 150	(1,2) 189.3	(1,2) 189.3	(1,2) 189.3
P_3	(1) 153	(2) 150	(1,2) 189.3	(2,3) 206.8	(2,3) 206.8
P_4	(1) 153	(2) 150	(1,2) 189.3	(1,4) 324.3	(1,4) 324.3
P_5	(1) 153	(2) 150	(1,2,5) 189.3	(1,4) 324.3	(1.4) 324.3
P_6	(1) 153	(2) 150	(1,2,5,6) 189.3	(1,4,6) 324.3	(1,4,6) 324.3
P_7	(1) 153	(2) 150	(1,2,5,6) 189.3	(1,4,6,7) 324.3	(1,4,6,7) 324.3

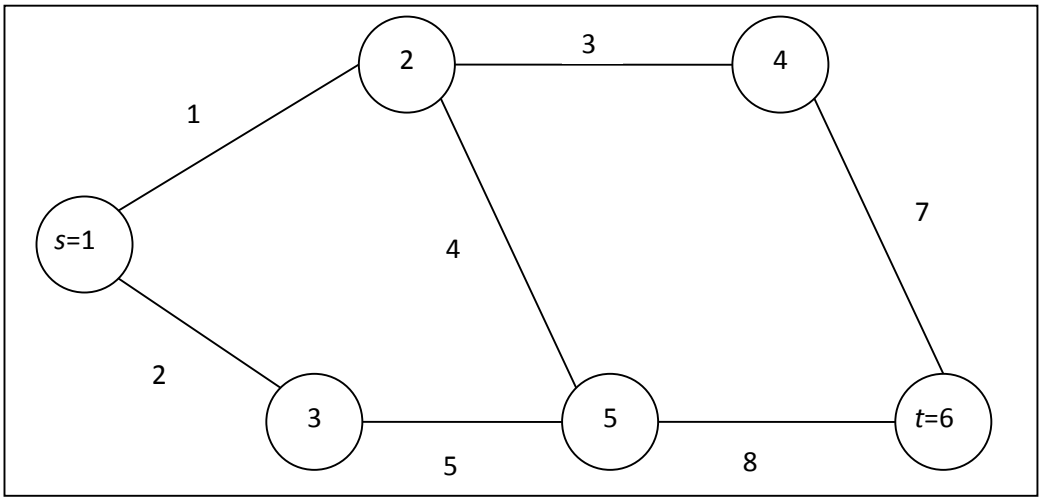


Fig. 5.2: The best solution of NTD-CB/R with R_{min}=0.8 for CN in Fig. 2.2

5.6 Pareto Optimal Set

As discussed in Section 5.2, minimizing cost and maximizing reliability or minimizing cost and maximizing bandwidth are two conflicting objectives; *e.g.*, minimizing cost by removing some links will make the network topology less reliable, while maximizing reliability by including more links will make the network more expensive. Specifically, NTD-CR/B (NTD-CB/R) is a BO problem that trade-off between cost and reliability (bandwidth) among its multiple feasible non-dominated solutions. Such solutions are called Pareto Optimal Set (POS), described in Section 2.4. In general, we

cannot get one optimal solution for such problem, but rather need to generate a POS. For this case, generating as many non-dominated solutions as possible in POS is favourable to provide users more selection options [11]; please refer to Section 2.4 for the definition of non-dominated solution.

This Section proposes DPCR/B-P and DPCB/R-P algorithms that extend DPCR/B and DPCB/R to include the POS concept. Both DPCR/B-P and DPCB/R-P algorithms require one additional set of information, called *aggregate element*, which contains non-dominated solutions. For DPCR/B-P, the element is stored in an additional array A[i, B_{min}] of its DP table. Specifically, A[i, B_{min}] is a data structure that stores cost, reliability value and path set for each non-dominated feasible solution; in the following discussion, we do not explicitly discuss the path set information of the array. Each A[i, B_{min}], for i=1, 2, …, m, is computed using the following three equations.

$$A[i, B_{min}]=(Rel_2(P_i), Cost(P_i)), \text{ if } i=1 \text{ and } BW(P_i)\geq B_{min} \quad (5.23)$$

$$A[i, B_{min}]=(\infty,\infty), \text{ if } i=1 \text{ and } BW(P_i)<B_{min} \quad (5.24)$$

$$A[i, B_{min}]=ND(A[i\text{-}1, B_{min}] \cup ((Rel_2(P_i \cup X[i\text{-}1, j]), Cost(P_i\} \cup X[i\text{-}1, j]))) \text{ if } i>1, 0\leq j\leq B_{min} \text{ and } BW(P_i \cup (X[i\text{-}1, j]))\geq B_{min} \quad (5.25)$$

Eq. (5.23) and Eq. (5.24) are used for the first path after using Eq. (5.15) and Eq. (5.16), described in Section 5.4.1. The path's reliability, $Rel_2(P_i)$ and its cost, $Cost(P_i)$, are stored in A[i, B_{min}] if the first path is a feasible solution, *i.e.*, if $BW(P_i)\geq B_{min}$, giving A[i, B_{min}]=($Rel_2(P_i)$, $Cost(P_i)$). In contrast, when $BW(P_i)<B_{min}$, P_1 is not a feasible solution and thus, is not selected in A[i, B_{min}]; this case is represented by A[i, B_{min}]=(∞,∞) in Eq. (5.24) to denote that no path is yet selected. Eq. (5.25), used for each remaining P_i, for i=2, 3, …, m, considers the case when selecting P_i together with

some previously selected paths in (X[i-1, j]) produces a feasible solution, *i.e.*, BW(P_i ∪ (X[i-1, j]))≥B_{min}. This step considers each possible column j=J[i-1, b]=0, 1, …, B_{min}. In this case, the reliability and cost values of including P_i, together with (X[i-1, j]), are computed, and each non-dominated solution is stored, *i.e.*, A[i, B_{min}]=ND((Rel_2(P_i ∪ X[i-1, j]), Cost(P_i ∪ X[i-1, j]))). Specifically, the 'ND' operator considers only the reliability and cost of all non-dominated feasible outcomes. In other words, the operator either ignores each feasible solution that is dominated by any other feasible solutions in A[i, B_{min}] or stores a new non-dominated feasible solution and deletes each existing solution in A[i, B_{min}] that it dominates. Finally, DPCR/B-P selects the best solutions from A[m, B_{min}]. In particular, the algorithm uses Eq. (5.5) to select a solution with the highest weight from the array. As later shown in Section 5.7.3, DPCR/B-P can generate a topology that on average has 2.17% larger weight as compared to DPCR/B that generates only one feasible solution in X[m, B_{min}].

For DPCB/R-P algorithm, array A[i, Ř$_{min}$] stores cost, bandwidth value and path set for each non-dominated feasible solution. Each A[i, Ř$_{min}$], for i=1, 2, …, m, is computed using the following three equations:

A[i, R_{min}]=(BW(P_i), Cost(P_i)), if i=1 and Rel_2(P_i)≥R_{min} (5.26)

A[i, R_{min}]=(∞,∞), if i=1 and Rel_2(P_i)<R_{min} (5.27)

A[i, R_{min}]=ND(A[i-1, R_{min}] ∪ (BW(P_i ∪ X[i-1, j]), Cost(P_i ∪ X[i-1, j]))) if i>1, 0≤j≤R_{min} and Rel_2((P_i ∪ (X[i-1, j]))≥R_{min} (5.28)

Equations (5.26), (5.27) and (5.28) for NTD-CB/R are respectively equivalent to Eq. (5.23), (5.25), and (5.25) for NTD-CR/B, except for (i) Rel_2(P_i) and BW(P_i) (ii) B_{min}

and R_{min}, constraint, as an example, A[i,*] in the former * is R_{min} while that for the latter * is B_{min}.

The simulation results in Section 5.7.3 show that DPCB/R-P, which selects the best solutions from each Pareto Optimal Set, can produce near optimal results with an improvement up to 3.37% as compared to DPCB/R that aims to generate only one topology for each network.

5.7 Evaluation

Section 5.7.1 describes the effects of using all 10 order criteria, described in Section 2.6.2 and 5.3.4, on the performance of DPCR/B and DPCB/R on the 20 networks, described in Section 2.6.1. To save space, for DPCR/B, the section discusses only the best three OCs, i.e., OC1, OC7 and OC9; similarly, for DPCB/R, it only reports the effects of using OC3, OC8, and OC10. Section 5.7.2 generates 100 benchmark networks with known optimal topology and uses them to gauge the effectiveness of DPCR/B and DPCB/R. Finally, Section 5.7.3 compares the performances of DPCR/B-P and DPCB/R-P against DPCR/B and DPCB/R, respectively. The section uses the 100 benchmark networks generated in Section 5.7.2 in the experiments.

5.7.1 Effects of Input Orders on DPCR/B and DPCB/R

For each of the 20 networks [22] in Table 5.3, we first randomly assign r_j, c_j and b_j for each e_j, and set B_{min} and R_{min} as described in Section 2.6.1. To evaluate the effectiveness of using order criterion OC7 with DPCR/B-1 approach, we compare its performance against three other orders, OC1, OC9, and random. We select OC1 because it shows good results for NTD-R/C and NTD-C/R; see Section 3.4.1 and Section 4.4.1, respectively. For OC9, we set w_j=b_j, while the random order uses the

same Lagrange Relaxation as OC7 to compute the weight of each link e_j, *i.e.*, $w_j=c_j\times r_j+\eta_w\times b_j$. However, in contrast to OC7 that selects the paths in decreasing order of their weights, in random order, we select the paths randomly. To implement OC1 and OC9, we replace the link weight calculation in Step 1 of DPCR/B algorithm in Section 5.2.3 with $w_j=c_j/r_j$ and $w_j=b_j$, respectively.

For DPCB/R, we use OC3 and introduce order criterion OC10 that sets $w_j=c_j/b_j$. To implement OC3 and OC10, we replace the link weight calculation in Step 1 of DPCR/B algorithm in Section 5.3 with $w_j=-(\log r_j)$ and $w_j=c_j/b_j$, respectively.

Next, we use Yen's algorithm [46] to generate each path set. As shown in first part of Table 5.3, DPCR/B-1 with each order criterion produces between 8 and 17 of the best results (in bold) as compared to only 5 using a random order. Note that we used rc_r explained in Section 5.1, to gauge the quality of the results for each networks and select the best among all results. As shown in second part of Table 5.3, DPCB/R-1 with each order criterion produces between 9 and 16 of the best results (in bold) as compared to only 7 using a random order. The table shows that OC7 is the best performer with 17 best results, followed by OC9 with 9 best results for DPCR/B-1. Further, the table also shows that OC8 is the best performer with 16 best results, followed by OC10 with 14 best results for DPCB/R-1.

However, OC1 (OC3) produces only 8 (9) best results for DPCR/B-1 and DPCB/R-1, respectively. One may run DPCR/B-1 and DPCB/R-1 with all three order criteria and select the best results among them; for the simulations, we would have obtained all 20 best results.

Table 5.3: The effects of path orders on DPCR/B-1 and DPCB/R-1 performance

		DPCR/B-1				DPCB/R-1				
		Cost(G_m), Rel$_2$(G_m)				Cost(G_m),BW(G_m)				
CN	B_{min}	**Random**	**OC1**	**OC7**	**OC9**	R_{min}	**Random**	**OC3**	**OC8**	**OC10**
$CN_4^{4,5}$	6	**(18,0.912)**	**(18,0.912)**	**(18,0.912)**	**(18,0.912)**	0.85	**(18,6)**	**(18, 6)**	**(18, 6)**	**(18, 6)**
$CN_9^{5,8}$	9	**(14,0.902)**	**(14,0.902)**	**(14,0.902)**	**(14,0.902)**	0.7	**(14, 8)**	**(14, 8)**	**(14, 8)**	**(14, 8)**
$CN_7^{6,8}$	6	(25,0.761)	**(20,0.833)**	**(20,0.833)**	**(20,0.833)**	0.8	(20, 8)	(20, 8)	**(25, 11)**	(20, 8)
$CN_{13}^{6,9}$	10	(25,0.843)	(28,0.917)	**(26,0.937)**	(27,0.923)	0.75	(27, 10)	(27, 10)	**(26, 12)**	(27, 10)
$CN_{25}^{7,12}$	12	(35,0.896)	(34,0.894)	**(36,0.920)**	(34,0.894)	0.80	**(34, 12)**	**(34, 12)**	**(34, 12)**	**(34, 12)**
$CN_{14}^{7,15}$	12	(31,0.648)	(28,0.648)	**(30,0.702)**	**(30,0.702)**	0.80	(33, 13)	(28,11)	**(30, 15)**	**(30, 15)**
$CN_{20}^{8,12}$	10	**(37,0.899)**	(38,0.886)	**(37,0.899)**	(36,0.704)	0.80	**(37, 13)**	(36, 10)	**(37, 13)**	**(37, 13)**
$CN_{24}^{8,12}$	10	(40,0.907)	**(34,0.892)**	**(34,0.892)**	**(34,0.892)**	0.60	(42, 10)	**(34, 11)**	(44, 12)	**(34, 11)**
$CN_{29}^{8,13}$	14	**(34,0.97)**	**(34,0.970)**	(36,0.95)	(36,0.972)	0.50	(34, 14)	(34, 14)	(34, 14)	**(36, 16)**
$CN_{13}^{9,12}$	12	**(27,0.837)**	(29,0.817)	(30,0.707)	(28,0.768)	0.60	**(27, 14)**	(29, 12)	(29, 12)	**(27, 14)**
$CN_{18}^{9,13}$	14	(66,0.994)	(57,0.985)	**(53,0.991)**	(55,0.987)	0.75	(63, 14)	(58, 12)	**(53, 13)**	(56, 12)
$CN_{44}^{9,14}$	10	(79,0.783)	**(77,0.799)**	**(77,0.799)**	**(77,0.799)**	0.75	(79, 12)	**(77, 13)**	**(77, 13)**	**(77, 13)**
$CN_{64}^{10,21}$	15	(46,0.913)	**(43,0.917)**	**(43,0.917)**	(45,0.910)	0.80	(46, 10)	(45, 11)	**(43, 16)**	**(43, 16)**
$CN_{18}^{11,21}$	10	(33,0.979)	(33,0.979)	**(34,0.984)**	(33,0.979)	0.82	**(33, 11)**	**(33, 11)**	**(33, 11)**	**(33, 11)**
$CN_{281}^{13,22}$	10	(78,0.980)	(75,0.965)	**(68,0.982)**	**(68,0.982)**	0.60	(68, 11)	(69, 13)	**(68, 16)**	**(68, 16)**
$CN_{36}^{16,30}$	10	(38,0.867)	(43,0.935)	**(39,0.956)**	(40,0.935)	0.70	(40, 10)	**(39, 13)**	**(39, 13)**	(37,11)
$CN_{136}^{17,25}$	10	(64,0.970)	**(58,0.985)**	(64,0.970)	(65,0.982)	0.60	(64, 10)	(58, 8)	**(58, 12)**	(66, 10)
$CN_{281}^{18,27}$	10	(95,0.963)	(97,0.975)	**(93,0.972)**	**(93,0.977)**	0.70	(92, 10)	**(93, 15)**	**(93, 15)**	**(93, 15)**
$CN_{780}^{20,30}$	15	(82,0.974)	(80,0.977)	**(75,0.981)**	(79,0.972)	0.60	(78, 16)	**(75, 18)**	(77, 15)	(77, 14)
$CN_{44}^{21,26}$	12	(93,0.998)	(93,0.998)	**(89,0.996)**	**(89,0.996)**	0.60	**(89, 15)**	(93, 12)	**(89, 15)**	**(89, 15)**
Total		5	**8**	17	**9**		7	**9**	16	**14**

5.7.2 DPCR/B and DPCB/R on Benchmark Networks

Since we could not find any benchmark network, *i.e.*, network with known optimal solution, to evaluate DPCR/B and DPCB/R in both versions, 1 and 2, we used the following steps to generate our benchmark networks. First, we consider the best solution of each network G=(V, E) with only α=1 deleted link $e_i \in E$. For each possible |E| deletion, we generate G_i=(V, E-{e_j}), for i=1,2,…,|E|, and compute its BW(G_i) for NTD-CR/B and Rel$_2$(G_i) for NTD-CB/R. Then, we select a G_i with the largest rc_i=Rel$_2$(G_i)/Cost(G_i) (bc_i=BW(G_i)/Cost(G_i)) as the G_{best}, and set B_{min}=BW(G_i) (R_{min}=Rel$_2$(G_i)), for NTD-CR/B and NTD-CB/R, respectively. Note that given G, B_{min}

and R_{min}, the best solution would generate $G_{best}=G_i$. We repeat the steps for α=2,3,4,5. Using the steps, we generated 100 networks from the 20 topologies in [22].

For DPCR/B and DPCB/R approaches in both versions, we have run thrice, once using (OC1, OC7 and OC9) and (OC3, OC8 and OC10), respectively, to generate topologies from the 100 benchmark networks, and take the best results among them. For each benchmark network G_i, we set B_{min}=BW(G_{best}) and R_{min}=Rel_2(G_{best}), and use DPCR/B and DPCB/R, respectively, to generate its topology G'$_{best}$. Notice that our method evaluates the worst case performance of DPCR/B and DPCB/R because each B_{min} and R_{min} is the tightest constraint. Nevertheless, DPCR/B, in both versions, generates 92% best results; due to limited space, Table 5.4 shows only the 8 non-optimal results.

Table 5.4: Comparison between DPCR/B and optimal results

Input			***k***	**DPCR/B-1 and DPCR/B-2**	
CN	α	B_{min}		**$Rel_2(G_k)$**	**Cost(G_k)**
$CN_{24}^{8,12}$	2	9	15(60%)	0.920(-0.7%)	36(1.05%)
$CN_{13}^{9,12}$	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%)
$CN_{281}^{13,22}$	4	13	66(23.4%)	0.982(-0.02%)	68(-0.98%)
$CN_{36}^{16,30}$	3	12	12(33.3%)	0.956(1.02%)	39(1.05%)
$CN_{136}^{17,25}$	5	7	41(30.1%)	0.974(-0.09%)	77(-0.96%)
$CN_{136}^{17,25}$	3	11	37(18%)	0.969(-0.4%)	78(0%)
$CN_{281}^{18,27}$	5	15	102(36.1%)	0.97(-0.04%)	82(-0.94%)

Interestingly, of the non-optimal results, DPCR/B-2 produces network with reliability no worse than 1.02% off from the (see % at column $Rel_2(G_{best})$ in Table 5.4) with some of which have lower cost than that of optimal by up to 1.05%. Furthermore, it uses only between 18% to 61.5% of (*s*, *t*) paths; see the % at column '*k*'. Moreover, DPCB/R generates 89% optimal results; Table 5.5 shows only the 11 non-optimal results. The non-optimal results, DPCB/R-2 produces network with bandwidth no worse than 1.47% off optimal (see % at column (BW(G_{best}) in Table 5.5) with some

of which have lower cost than that of optimal by up to 2.1%. Furthermore, it uses only between 22.4% to 64.2% of paths; see the % at column '*k*'.

Table 5.5: Comparison between DPCB/R and Optimal results

Input			*k*	DPCB/R-1and DPCB/R-2	
CN	**α**	R_{min}		**BW(G_k)**	**Cost(G_k)**
$CN_{25}^{7,12}$	2	0.88	15(60%)	15(-0.9%)	33(-0.8%)
$CN_{14}^{7,15}$	1	0.80	9(64.2%)	12(0.17%)	28(2.1%)
$CN_{20}^{8,12}$	2	0.95	11(55%)	13(1.01%)	36(1.03%)
$CN_{29}^{8,13}$	2	0.92	12(41.3%)	12(0.7%)	36(0.32%)
$CN_{44}^{9,14}$	5	0.86	21(47.7%)	14(0.38%)	56(0.69%)
$CN_{64}^{10,21}$	5	0.94	33(51.5%)	17(0.62%)	68(1.03%)
$CN_{281}^{13,22}$	4	0.97	63(22.4%)	12(-0.34%)	65(-0.84%)
$CN_{36}^{16,30}$	3	0.96	10(27.7%)	8(1.47%)	41(1.05%)
$CN_{136}^{17,25}$	3	0.93	47(34.5%)	10(-0.4%)	68(0%)
$CN_{281}^{18,27}$	5	0.94	86(30.4%)	10(0.52%)	75(0.72%)
$CN_{780}^{20,30}$	4	0.95	213(27.3%)	18(-0.18%)	87(-0.73%)

5.7.3 Performance of DPCR/B-P and DPCB/R-P

We evaluate the performances of DPCR/B-P and DPCB/R-P on the 100 benchmark networks generated in Section 5.7.2, and compare their results against those generated by DPCR/B and DPCB/R respectively.

We use OC1, OC7 and OC9 for DPCR/B-P and DPCR/B, and OC3, OC8 and OC10 for DPCB/R-P and DPCB/R. For each OC and each network, both DPCR/B-P and DPCB/R-P generate Pareto Optimal Set of topologies, and select the topology with the highest quality indices, *i.e.*, $rc_i=Rel_2(G_i)/Cost(G_i)$, and $cb_i=BW(G_i)/Cost(G_i)$ respectively (discussed in Section 5.2), as their outcomes. For example, for $CN_{13}^{6,9}$ with B_{min}=0.85, DPCR/B-P generates POS={(29, 0.895), (26, 0.901), (25, 0.863), (23, 0.854)}, and thus $rc_1=Rel_2(G_1)/Cost(G_1)=0.03086$, $rc_2=Rel_2(G_2)/Cost(G_2)=0.03465$, $rc_3=Rel_2(G_3)/Cost(G_3)=0.03452$ and $rc_4=Rel_2(G_4)/Cost(G_4)=0.03713$. Since $rc_4=0.03713$ is the largest among the feasible solutions, DPCR/B-P selects G_4 as the best solution.

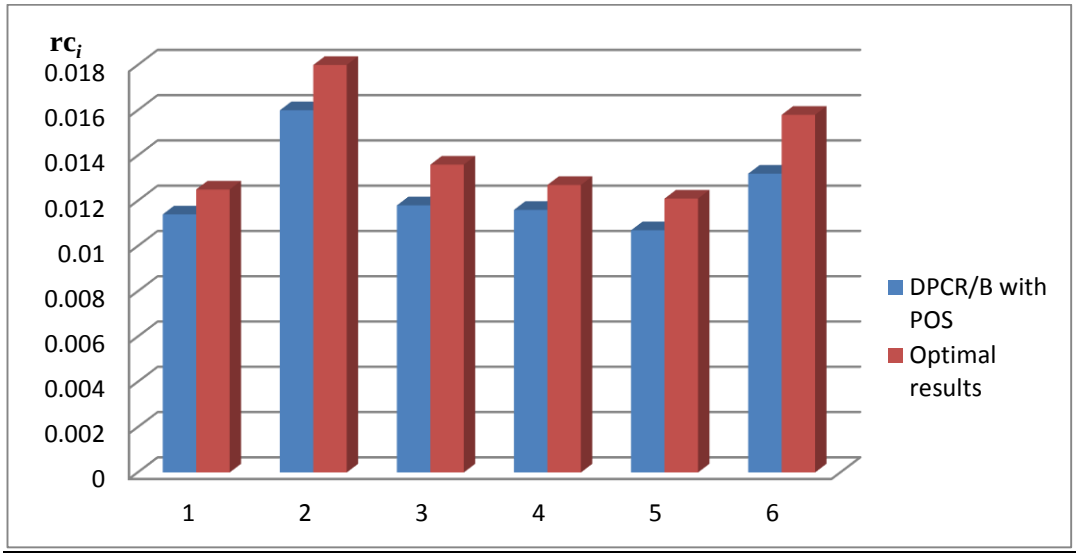


Fig. 5.2: Comparison between the results of DPCR/B-P and optimal results

DPCR/B-P and DPCB/R-P generate 94% and 92% optimal results respectively; Fig. 5.2 shows only the 6 non-optimal results of DPCR/B-P and Fig. 5.3 shows the 8 non-optimal results DPCB/R-P. As shown in Fig. 5.2, in term of index rc_i, DPCR/B-P produces results no larger than 16.12% off from optimal, *i.e.*, rc_6=0.0155 versus rc_6=0.013 at the rightmost bar of the figure. Further, Fig. 5.3 shows that, in term of cb_i, DPCB/R-P produces results no worse than 16.66% off from optimal, *i.e.*, bc_6=0.014 versus bc_6=0.0168. Recall that, as reported in Section 5.4.2, DPCR/B and DPCB/R generate 92% and 89% optimal results for the 100 benchmark networks, respectively. Thus, DPCR/B-P and DPCB/R-P improve the performance of DPCR/B or DPCB/R by 2.17% and DPCR/B 3.37% respectively.

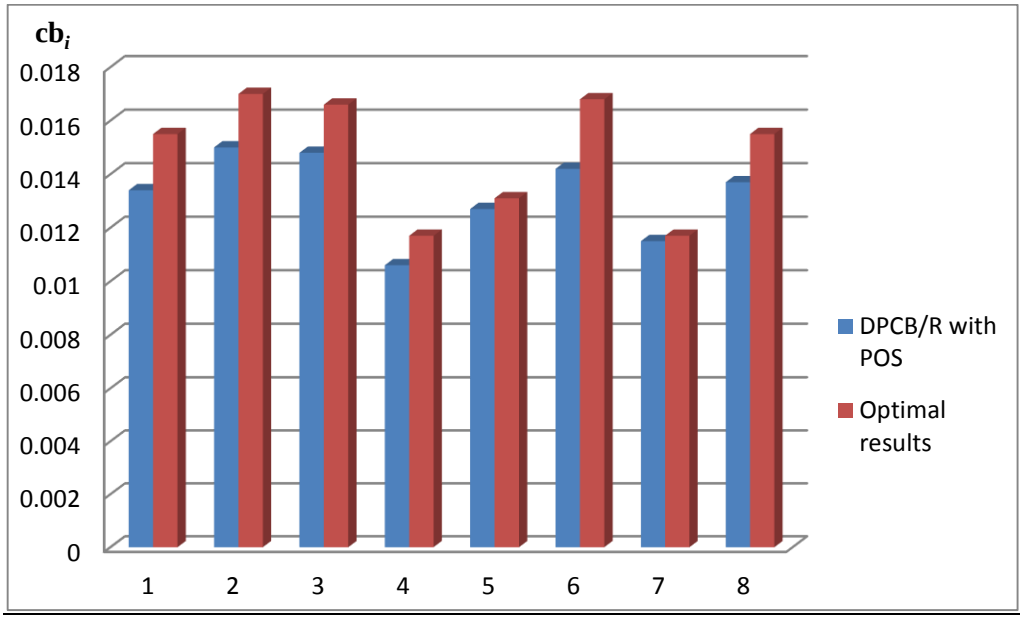


Fig. 5.3: Comparison between the results of DPCB/R-P and optimal results

5.8 Chapter Summary

We have defined two network topology design problems, namely NTD-CR/B and NTD-CB/R. NTD-CR/B aims to generate a topology that has the minimum cost and maximum reliability subject to a bandwidth constraint B_{min}, while the goal of NTD-CB/R is to minimize cost and maximize bandwidth subject to a reliability constraint R_{min}. We have proposed two heuristic dynamic programming methods, DPCR/B and DPCB/R, to solve NTD-CR/B and NTD-CB/R respectively. We have proposed two new metrics, rc_i and bc_i, to rank the goodness of feasible topologies generated for NTD-CR/B and NTD-CB/R problems respectively. Further, for both solutions, we have proposed Lagrange Relaxation method and weighted sum to convert each of the BO optimization into SO optimization. The first version of the solutions, namely DPCR/B-1 and DPCB/R-1, require all (s, t) paths of the networks to generate the results, while their second versions, respectively called DPCR/B-2 and DPCB/R-2, incrementally generate only selected k paths from the network and, thus, are scalable

on networks with large number of paths. We have proposed two new order criteria, for DPCR/B, *i.e.*, OC7 and OC9, and other two order criteria for DPCB/R, *i.e.*, OC8 and OC10, to optimize the effectiveness and efficiency of both versions of the algorithms. Finally, we have extended both versions of DPCR/B and DPCB/R to include POS concept; the extensions are called DPCR/B-P and DPCB/R-P respectively. On 100 benchmark networks, our experiments show that both versions of DPCR/B and DPCB/R generate respectively 92% and 89% optimal solutions. On the same networks, DPCR/B-P and DPCB/R-P generate 94% and 92% optimal results, and thus improve DPCR/B and DPCB/R by 2.17% and 3.37% respectively. Further, the simulations have shown the effectiveness of OC1, OC7 and OC9 for DPCR/B, and OC3, OC8 and OC10 for DPCB/R.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Many network applications require sufficient level of quality of service (QoS) to function correctly. Topology design of communication network (CN) is critical to the performance of the applications that use the network. The research presented in this thesis focuses on using dynamic programming (DP) approach to optimize one or two performance objectives while satisfying one given QoS constraint. We have formally defined four reliable network topology design problems for CN and their corresponding DP solutions. Each problem considers Rel_{all} and/or Rel_2 reliability measures, and its solution requires the network's set of spanning trees and (s, t) paths as input respectively. The thesis has shown that each solution can produce better results when the inputs, spanning trees or (s, t) paths, are optimally sorted. Because generating optimal input orders is shown NP-complete problem, the thesis has proposed 10 heuristic order criteria, *i.e.*, OC1 to OC10. The thesis has shown that a good selection of order criterion for each problem can improve the optimality of the solution as well as improving its computational time efficiency. However, due to the heuristic nature of each order criterion, a criterion that is best used in one solution may not fit for used in other solution. Table 6.1 summarizes the works covered in this thesis and its findings on the best order criterion for use in each solution.

Table 6.1: Four NTD problems with proposed solutions, functions and best order

Problem	Solution	Reliability measure	Best order criterion	Chapter
NTD-R/C	DPR/C	Rel_{all}	OC2	Chapter 3
		Rel_2	OC1	
NTD-C/R	DPC/R	Rel_{all}	OC1	Chapter 4
		Rel_2	OC6	
NTD-CR/B	DPCR/B	Rel_2	OC7	Chapter 5
	DPCR/B-P			
NTD-CB/R	DPCB/R	Rel_2	OC8	Chapter 5
	DPCB/R-P			

In Chapter 3, we have formally defined the first network topology design problem, NTD-R/C, to generate a topology that has the maximum reliability subject to a cost constraint C_{max}. We have proposed a heuristic DP approach, DPR/C, to solve NTD-R/C. The first version of the method, called DPR/C-1, requires all spanning trees as input, while its second version, called DPR/C-2, uses only *k* spanning trees to improve the time efficiency of DPR/C-1 while producing similar results. The chapter has described five order criteria, OC1 to OC5, which allow DPR/C-2 to incrementally generate only the first *k* spanning trees from the network while producing almost similar results as compared to using all spanning trees in the network; for this case, OC2 is the most suitable criterion. Appendix A has shown how to use DPR/C for Rel_2(G) measure using (*s*, *t*) simple paths as input; in term of order criterion, OC1 is the best for this case. Our simulations on various large grid networks that contain up to 200 nodes, 298 links, 2^{99} paths and 1.899^{102} spanning trees show the practicality of our techniques. These results provide clear evidence that DPR/C is effective and efficient for designing large networks. Note that DPR/C utilizes the sum of disjoint products technique [22] to compute Rel_{all} and Rel_2, and our simulations show the

benefits of using lexicographic order coupled with any one of the order criteria to speed up the computation.

In Chapter 4, we have proposed a heuristic DP algorithm, DPC/R, to solve an NP-hard problem, called NTD-C/R, to design a minimal-cost CN topology that satisfies a pre-defined reliability constraint. For Rel_{all}, DPC/R generates the topology using a selected sequence of spanning trees of the network; Appendix B shows how to use DPC/R for $Rel_2(G)$ measure using (s, t) simple paths of the network as input. The first version of the algorithm, namely DPC/R-1, requires all spanning trees of the network. Theoretical analysis shows that DPC/R always obtains a feasible solution, and produces an optimal topology given an optimal order of spanning trees. The chapter has described a formal proof to show that generating optimal order of spanning trees is NP-complete, and uses five greedy heuristics, *i.e.*, OC1 to OC5 to improve the effectiveness of DPC/R; for DPC/R with Rel2, Appendix B proposes order criterion OC6. The OCs allow the second version of the algorithm, namely DPC/R-2, to use only k spanning trees or (s, t) paths of the network while, similar to DPC/R-1, producing near optimal topologies. Simulations using fully connected networks that contain up to 2.3×10^9 spanning trees show the merits of using the ordering methods, especially OC1, and the effectiveness of DPC/R-2 vis-à-vis to four existing state-of-the-art techniques. Our DPC/R-2 is able to generate 81.5% optimal results, while using only 0.77% of the spanning trees contained in networks. Further, for a typical 2×100 grid network that contains up to 1.899^{102} spanning trees, DPC/R-2 requires only k=1214 spanning trees to generate a topology with a reliability no larger than 5.05% off from optimal. Similarly, the chapter has shown that DPC/R-2 is able to generate 91% optimal solutions on the networks using only 8.89% to 27.5% of all (s, t) paths of the networks. Further, its

non-optimal results are no more than 10.97% off from optimal. Our simulations have shown that OC6 is the best criterion for DPC/R with Rel_2 measure.

Finally, in Chapter 5, we have presented two bi-objective network topology design problems: (i) NTD-CR/B that aims to minimize network cost and maximize its Rel_2 subject to a given network bandwidth, and (ii) NTD-CB/R that aims to minimize network cost and maximize its bandwidth subject to a given Rel_2 constraint. We formulate DP schemes, called DPCR/B and DPCB/R, with Lagrange Relaxation method to solve NTD-CR/B and NTD-CB/R, respectively. The first version of each scheme, called DPCR/B-1 and DPCB/R-1 respectively, generate the topology using all (s, t) paths of the network. We have proposed three greedy heuristics, *i.e.*, OC7 to OC10, each of which allows the second version of DPCR/B and DPCB/R, *i.e.*, DPCR/B-2 DPCB/R-2 respectively, to produce results using only $k \leq m$ paths, improving the time complexity while producing near optimal topologies. We have extended both DPCR/B and DPCB/R to DPCR/B-P and DPCB/R-P, respectively, to generate the Pareto Optimal Set of non-dominated solutions. Our simulations using benchmark networks with various sizes have shown the advantages of using the OCs and effectiveness of our approaches. The results show that OC7 is the best for DPCR/B while OC8 is the most suitable for DPCB/R. Our evaluations show that DPCR/B and DPCB/R are able to generate 92% and 89% optimal results while using only 18% to 61.5% and 22.4% to 64.2% of all (s, t) paths, respectively. Finally, DPCR/B-P and DPCB/R-P produce on average 2.17% and 3.37% better topologies as compared to DPCR/B and DPCB/R respectively.

6.2 Future Work

All of our DP algorithms presented in this thesis incrementally and selectively add spanning trees or paths, and thus links, to form an optimal topology. As an alternative, we plan investigate the possibility of designing a DP approach that heuristically deletes links from the original topology to find an optimal design. Further, we plan to extend the four problems, NTD-R/C, NTD-C/R, NTD-CR/B and NTD-CB/R, for the case when nodes can fail and incur a set up cost. We also plan to investigate the possibility of extending our approaches for network topology design that includes other network performance constraints, *e.g*., delay and throughput.

Appendix A

DPR/C for NTD-R/C with Rel_2

In this appendix, we show how to use DPR/C algorithm (explained in Section 3.3) to solve NTD-R/C with Rel_2 measure. For brevity, in this appendix, we call NTD-R/C with Rel_{all} and Rel_2 measures R/C-A and R/C-B. For R/C-B, both versions of DPR/C, *i.e.*, DPR/C-1 and DPR/C-2, use a sequence of (s, t) simple paths of the network as input. DPR/C-1 requires all of m (s, t) simple paths of the network as its input, while DPR/C-2 generates and uses only $k \leq m$ paths.

A.1 Dynamic Programming Formulation

The following dynamic programming (DP) formulation defines R/C-B:

$R[i, c]=0$, for $i=1$ with $Cost(P_1)>c$ (A.1)

$R[i, c]=Rel_2(P_1)$, for $i=1$ with $Cost(P_1)\leq c$ (A.2)

$R[i, c]=R[i\text{-}1, c]$, for each $i>1$ and $j=0, \ldots, c$ with $Cost(X[i\text{-}1, j] \cup P_i)>c$ (A.3)

$R[i,c]=Max(R[i\text{-}1,c], Rel_2(X[i\text{-}1, j] \cup P_i))$, for each $i>1$ and $j=0, \ldots, c$ with $c \leq Cost(X[i\text{-}1,j] \cup P_i) \leq C_{max}$ (A.4)

Equations (A.1), (A.2), (A.3) and (A.4) are respectively equivalent to Eq. (3.5), (3.6), (3.7) and (3.8) for R/C-A, described in Section 3.2, except for (i) $Cost(P_i)$ and $Cost(ST_i)$, (ii) $Rel_2(P_i)$ and $Rel_{all}(ST_i)$, and (iii) the contents of their DP tables. Specifically, the DP table used for Eq. (A.1) to (A.4) store information related to (s, t) paths and Rel_2, in contrast to that for Eq. (3.5) to (3.8) that store information related to

spanning trees and Rel_{all}; as an example, R[i, c] in the former stores Rel_2 while that for the latter stores Rel_{all} values.

A.2 Illustrating Example

To illustrate DPR/C-1 for R/C-B, consider the CN in Fig. 2.2 with its link weight and path set in Table 2.2, and C_{max}=20. DPR/C-1 constructs the DP table in Table A.1, to obtain the optimal network in Fig. 2; for convenience, we show all m=7 rows although our implementation creates only two rows. Each row of the table considers each P_i for possible selection and each column shows the budget cost $c \leq C_{max}$. Since the minimum cost of any path is 9 and C_{max}=20, DPR/C-1 requires 20-9+1=12 columns. Each element in the table shows the set of paths selected from Table 2.2 and its reliability value. Consider that DPR/C-1 algorithm with DP formulation in Eq. (A.1) - Eq. (A.4). Since Cost(P_1)=12, DPR/C-1 algorithm initialize the first row with X[1, c]=( ), R[1, c]=0 and J[1, c]=0 for c=9, …, 11, and X[1, c]=(P_1), R[1, c]=0.486 and J[1, c]=12 for c=12, 13, …, 20.

Next, for P_2, we copy *row* 1 to *row* 2 using Eq. (A.3), then use Eq. (A.4) to update *row* 2 to improve reliability at some columns when possible. For j=J[6, 12]=12 and Cost(X[1, 12] $\cup$ P_2)=18<C_{max}, selecting P_2 would generate Rel_2(X[1, 12] $\cup$ P_2)=0.509. On the other hand, if P_2 is not selected, the maximum achievable reliability is R[1,18]=0.486; thus P_2 is selected with P_1 for c=18; similarly for c=19, 20. Thus, for this case, DPR/C-1 set X[2, c]=(P_1, P_2), R[2, c]=0.509 and J[2, c]=18 at each column c=18, 19, 20.

When j=J[1, 0]=0, Cost(X[i+1, 0]=( ) $\cup$ P_2)=15<C_{max}. Since Rel_2(X[1, 0]={} $\cup$ P_2)<R[1, c]=Rel_2(P_1), keeping P_1 at each column c=15, 16 , …, 17 is better; thus DPR/C-1 keeps columns c=0, 1, …,15 unchanged.

As another example, take Cost(P_3)=17 and lines 7 to 11 set X[3, *c*]=X[2, *c*] and R[3, *c*]=R[2, *c*] for each column *c*. When *j*=18, Cost(X[*i*-1, 18]=(P_1, P_2) ∪ P_3=24>C_{max} and thus, following Eq. (A.3), selecting P_3 is not feasible. However, for *j*=12, Cost(X[*i*-1, 12]=P_1 ∪ P_3)=20≤C_{max}, and following Eq. (A.4); P_3 is selected with P_1 and update column *c*=20 with X[3, *c*]=(P_1, P_3), R[3, *c*]=0.51 and *J*[3, *c*]=20.

Table A.1: DP table for CN with C_{max}=20 in Fig. 2.2

c / P_i	9	10	11	12	13	14	15	16	17	18	19	20
1	X[1,9]=( ) R[1, 9]=0 *J*[1,9]=0	() 0 0	() 0 0	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12
2	() 0 0	() 0 0	() 0 0	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1,2) 0.50 18	(1,2) 0.50 18	(1,2) 0.50 18
3	() 0 0	() 0 0	() 0 0	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1,2) 0.50 18	(1,2) 0.50 18	(1,3) 0.51 20
4	() 0 0	() 0 0	() 0 0	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1,2) 0.50 18	(1,2) 0.50 18	(1,3) 0.51 20
5	(5) 0.44 9	(5) 0.44 9	(5) 0.44 9	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1) 0.49 12	(1,2) 0.50 18	(1,2) 0.50 18	(1,3) 0.51 20
6	(5) 0.44 9	(5) 0.44 9	(5) 0.44 9	(6) 0.73 12	(6) 0.73 12	(6) 0.73 12	(6) 0.73 12	(6,5) 0.81 16	(6,5) 0.81 16	(6,5) 0.81 16	(6,5) 0.81 16	(6,5) 0.81 16
7	(5) 0.44 9	(5) 0.44 9	(5) 0.44 9	(6) 0.73 12	(6) 0.73 12	(6) 0.73 12	(6) 0.73 12	(6,5) 0.81 16	(6,5) 0.81 16	(6,5) 0.81 16	(6,5) 0.81 16	(7,6,5) 0.83 20

For the budget, Cost(X[6, 20] ∪ P_7)=Cost((P_5, P_6) ∪ P_7)=Cost({1, 3, 4, 6, 7, 8})=20 and Rel_2(P_5, P_6, P_7)=0.833>R[6, 20]; thus P_7 is selected, and DPR/C-1 sets X[7, 20]=(P_5, P_6, P_7) as its output with reliability R[7, 20]=0.833. The best topology G_{opt}, was shown in Fig. A.1, is obtained by selecting all links in the paths {P_5, P_6, P_7}. Notice that paths (P_5, P_6, P_7) and (P_7, P_6, P_5, P_4) contain the same set of links, and thus they form the same best topology G_{opt} with Rel_2(G_{opt})=0.833 calculated from (P_7, P_6, P_5, P_4).

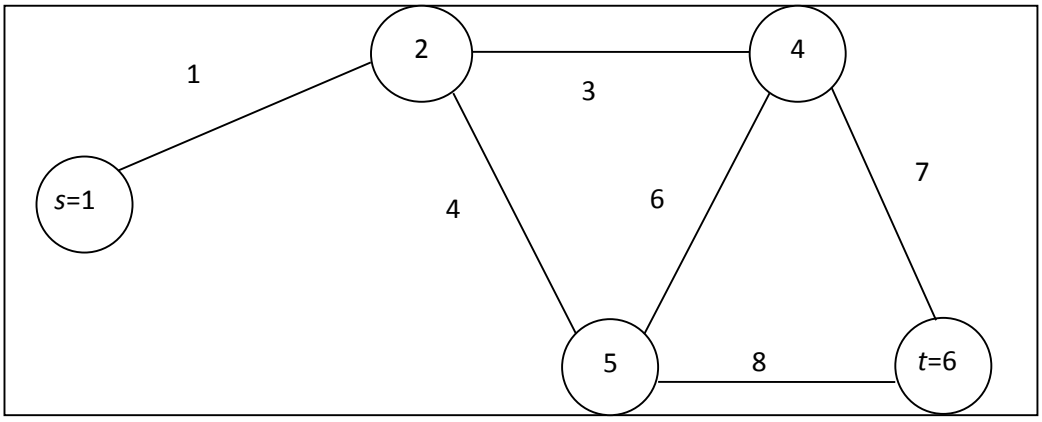


Fig. A.1: The best solution of NTD-R/C with C_{max}=20 for CN in Fig. 2.2

A.3 Order Criteria

The five order criteria, described in Section 3.3.4, can be used for DPR/C with Rel_2. However, because DPR/C with Rel_2 requires (*s*, *t*) paths as input, after computing link weight for each order criterion, the algorithm uses Yen's algorithm [46] to generate the paths in increasing weights. For example, we obtain the following orders for the paths in Table 2.2; OC1:(P_6,P_5,P_1,P_4,P_7,P_2,P_3), OC2:(P_5,P_1,P_6,P_4,P_2,P_7,P_3), OC3:(P_2,P_3,P_4,P_7,P_5,P_1,P_6), OC4:(P_5,P_1,P_6,P_4,P_2,P_7,P_3), and OC5:(P_6,P_5,P_1,P_4, P_7,P_2,P_3).

A.4 Algorithm Analysis

The algorithm analysis in Section 3.3, except for its time complexity, applies to DPR/C with Rel_2 measure. The time complexity of DPR/C for R/C-B is calculated as follows. Recall that, without using any order criterion, DPR/C-1 requires $O(\Omega\times|E|\times T_{max}+m\times|E|\times C_{max})$; see Section 3.3.1. Yen's algorithm requires $O(m\times|V|\times(|E|+|V|\times\log|V|))$ to generate *m* paths of the network and therefore DPR/C-1 requires in total $O(\Omega\times|E|\times T_{max}+m\times|E|\times C_{max}+m\times|V|\times(|E|+|V|\times\log|V|))$ time. Notice that DPR/C-2 needs only $k\leq m$ pahs, and thus its time complexity becomes $O(\Omega\times|E|\times T_{max}+k\times|E|\times C_{max}+k\times|V|\times(|E|+|V|\times\log|V|))$.

Appendix B

DPC/R for NTD-C/R with Rel_2

This appendix explains how to use DPC/R algorithm (explained in Section 4.3) to solve NTD-R/C with Rel_2 measure using (s, t) simple path. For brevity, in this appendix, we call NTD-C/R with Rel_{all} and Rel_2 measures C/R-A and C/R-B, respectively. For C/R-B, the first version of DPC/R, called DPC/R-1, requires all (s, t) simple paths as its input, while its second version, called DPC/R-2, uses only k (s, t) simple paths to improve the time efficiency of DPC/R-1 while producing similar results. Note that our work in Appendix B has been published in [4].

B.1 Dynamic Programming Formulation

The following dynamic programming (DP) formulation defines C/R-B:

$$C[i, \check{r}]=Cost(P_i) \text{ for } i=1 \text{ with } Rel_2(P_i)\geq r \quad (B.1)$$

$$C[i, \check{r}]=\infty \text{ for } i=1 \text{ with } Rel_2(P_i)<r \quad (B.2)$$

$$C[i, \check{r}]=Min(C[i\text{-}1, \check{r}], Cost(P_i)) \text{ for } i>1, \text{ and } Rel_2(P_i)\geq r \quad (B.3)$$

$$C[i, \check{r}]=Min(C[i\text{-}1, \check{r}], Cost(L[i\text{-}1, j] \cup L_i)) \text{ for } i>1,\ j\leq\check{r},$$
$$\text{and } Rel_2(L[i\text{-}1, j] \cup L_i)\geq r \quad (B.4)$$

Equations (B.1), (B.2), (B.3) and (B.4) are respectively equivalent to Eq. (4.5), (4.6), (4.7) and (4.8) for C/R-A, described in Section 4.2, except for (i) $Cost(P_i)$ and $Cost(ST_i)$, (ii) $Rel_2(P_i)$ and $Rel_{all}(ST_i)$, and (iii) the contents of their DP tables. Specifically, the DP table used for Eq. (B.1) to (B.4) store information related to (s, t) paths and Rel_2, in contrast to that for Eq. (4.5) to (4.8) that store information related to

spanning trees and Rel_{all}; as an example, R[i, c] in the former stores Rel_2 while that for the latter stores Rel_{all} values. Notice that the algorithm and analysis in Section 4.3 apply to DPC/R-1 with Rel_2 measure.

B.2 Illustrating Example

To illustrate DPC/R-1 to solve C/R-B, consider the CN in Fig. 2.2 with the following sequence of P_G=(P_1=(2,5,8), P_2=(2,5,6,7), P_3=(2,5,4,3,7), P_4=(1,3,6,8), P_5=(1,3,7), P_6=(1,4,6,7), P_7=(1,4,8)) and R_{min}=0.76. Table 2.2 shows the network's link reliability and cost, and path reliability and cost. DPC/R-1 constructs the DP table shown in Table B.1, and obtains the best topology in Fig. B.1 using Eq. (B.1) - Eq. (B.4).. Each row of the table considers a P_i for possible selection and its columns are labelled by reliability values from 0 to $\check{R}_{min}$.

Since $Rel_2(P_1)$=0.486, DPC/R-1 sets C[1, $\check{r}$]=12, R[1, $\check{r}$]=0.49, X[1, $\check{r}$]=(P_1) and L[1, $\check{r}$]=(2, 5, 8) for $\check{r}$=0, …, 49 with j=J[1, $\check{r}$]=49. Further, Eq. (B.1) initializes the first row with C[1, $\check{r}$]=∞, R[1, $\check{r}$]=0, X[1, $\check{r}$]=( ) and L[1, $\check{r}$]=( ) for $\check{r}$=50, …, $\check{R}_{min}$ and thus J[1, $\check{r}$]=$\check{R}_{min}$=83.

Next, for P_2 with $Rel_2(P_2)$=0.226, Eq. (B.3), for $\check{r}$=0, …, 49, produces C[2, $\check{r}$]=C[1, $\check{r}$], R[2, $\check{r}$]=R[1, $\check{r}$], X[2, $\check{r}$]=X[1, $\check{r}$] and L[2, $\check{r}$]=L[1, $\check{r}$], because C[1, $\check{r}$]=12<Cost(P_2)=15. For $\check{r}$=50, …, $\check{R}_{min}$, we consider j's starting from $\check{r}$=0 in *row* 1 with two values: j=J[1, $\check{r}$]=49 and j=J[1, $\check{r}$]=83. For j=49, $Rel_2((L[i\text{-}1,\ j=49]=L_1) \cup L_2)$=0.51; thus Eq. (B.4) selects both P_1 and P_2 at column $\check{r}$=50, 51 and sets C[2, $\check{r}$]=18, R[2, $\check{r}$]=0.51, X[2, $\check{r}$]=(P_1, P_2), L[2, $\check{r}$]=(2, 5, 6, 7, 8) and j=51. Note that it is not possible to select both P_1 and P_2 at column $\check{r}$=0, ..., 49, because Cost($L_1 \cup L_2$)>C[1, $\check{r}$]. For j=83, Eq. (B.4) produces exactly the same results at columns $\check{r}$=52, ..., 83 in *row* 1.

As another example, for P_3 with $Rel_2(P_3)=0.238$ and j=49, $Rel_2((L[i\text{-}1,\ j=49]=L_1) \cup L_3)=0.50$. However, because $Cost(L_3 \cup L_1)=20>C[2, \check{r}]$, Eq. (B.4) does not select both P_1 and P_3 at column $\check{r}$=0, …, 50 and does not update DP table for $\check{r}$=0, …, 51. On the other hand, for j=51, $Rel_2((L[i\text{-}1,\ j=51]=L_1 \cup L_2) \cup L_3)=0.52$, and thus Eq. (B.4) selects both P_1, P_2 and P_3 at column $\check{r}$=52 and sets $C[3, \check{r}]=24$, $R[3, \check{r}]=0.52$, $X[3, \check{r}]=(P_1, P_2, P_3)$, $L[3, \check{r}]=(2, 3, 4, 5, 6, 7, 8)$ and j=52.

Table B.1: DP table for CN with R_{min}=0.76 in Fig. 2.2

$\check{r}$ / P_i	0 to 44	45 to 49	50 to 51	52	53 to 54	55 to 64	65 to 71	72 to 73	74 to 76
P_1	R[1, $\check{r}$]=0.486 C[1, $\check{r}$]=12 X[1, $\check{r}$]=(1) J[1, $\check{r}$]=49	0.486 12 (1) 49	( )	( )	( )	( )	( )	( )	( )
P_2	0.486 12 (1) 49	0.486 12 (1) 49	0.510 18 (1,2) 51	( )	( )	( )	( )	( )	( )
P_3	0.486 12 (1) 49	0.486 12 (1) 49	0.510 18 (1,2) 51	0.52 24 (1,2,3) 52	( )	( )	( )	( )	( )
P_4	0.486 12 (1) 49	0.486 12 (1) 49	0.510 18 (1,2) 51	0.64 23 (1,4) 64	0.64 23 (1,4) 64	0.64 23 (1,4) 64	0.77 25 (1,2,4) 77	0.77 25 (1,2,4) 77	0.77 25 (1,2,4) 77
P_5	0.441 9 (5) 44	0.486 12 (1) 49	0.543 16 (4,5) 54	0.543 16 (4,5) 54	0.543 16 (4,5) 54	0.71 21 (1,5) 71	0.71 21 (1,5) 71	0.77 25 (1,2,4) 77	0.77 25 (1,2,4) 77
P_6	0.441 9 (5) 44	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.763 16 (5,6) 76
P_7	0.441 9 (5) 44	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.729 12 (6) 73	0.763 16 (5,6) 76

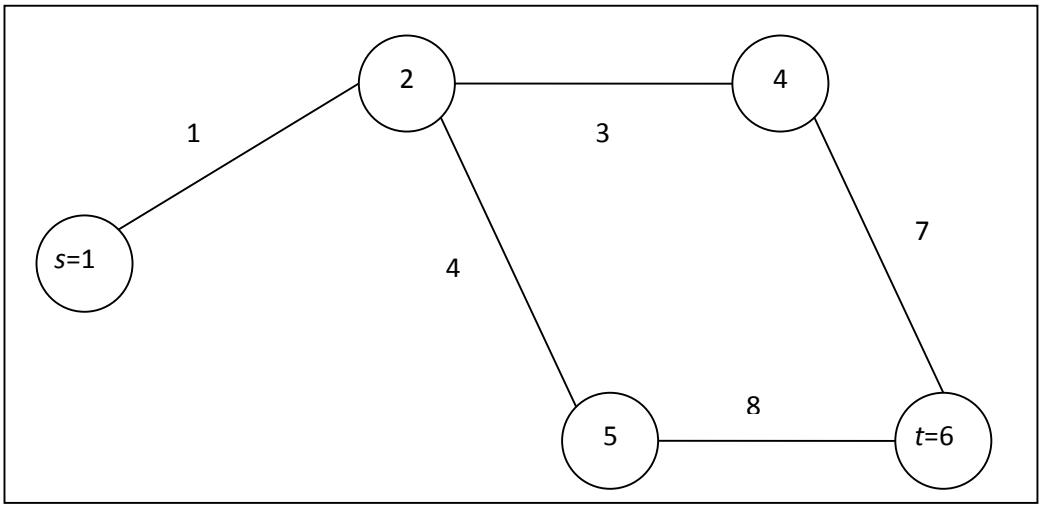


Fig. B.1: The best solution of NTD-C/R with R_{min}=0.76 for CN in Fig. 2.2

Repeating the steps for P_4 to P_7, DPC/R-1 obtains X[7, $\check{R}_{min}$]=(P_5, P_6) with reliability R[7, $\check{R}_{min}$]=0.763, cost C[7, $\check{R}_{min}$]=16 and L[7, $\check{R}_{min}$]=(1, 3, 4, 7, 8). The best topology G_{min}, shown in Fig. B.1, is obtained by selecting all links in the paths (P_5, P_6).

B.3 Order Criteria

The five order criteria, described in Section 3.3.4, can be used for DPC/R with Rel_2 measure. In this appendix, we propose another order criterion, called OC6, that uses the Lagrange Relaxation technique proposed in [51] to compute w_j as:

$$w_j=c_j+\eta_w\times(-\log r_j) \quad \text{(B.5)}$$

The value of η should be set properly to minimize Cost(G) and to maximize Rel_2(G). For λDP/RD problem that generates δ-edge disjoint paths with minimum cost, the authors [51] set η_w as:

$$\eta_w=\delta\times D_{min}/\log(R) \quad \text{(B.6)}$$

Where D_{min} is the minimum delay of all possible disjoint paths in the network. For our problem, we replace D_{min} in Eq. (B.6) with C_{min}, *i.e.*, the minimum cost of all possible paths in the network to obtain:

$$\eta_w=\delta\times C_{min}/\log(R_{min}) \quad \text{(B.7)}$$

One can obtain C_{min} using Dijkstra's algorithm [26]. Since our problem considers only one simple path, we set $\delta=1$ in Eq. (B.7); our experiments in Section 4.4.1 verifies the optimality of the chosen value for δ as compared to other values.

For each order criterion, we use Yen's algorithm [46] to generate all (*s*, *t*) paths in increasing weight order. The path weight is calculated as the summation of the weight of each link in the path. For example, using OC1 and OC6, we obtain order (P_6,P_5,P_1,P_4,P_7,P_2,P_3), and (P_5,P_2,P_6,P_3,P_1,P_7,P_4), respectively, for the paths in Table 2.2.

B.4 Algorithm Analysis

The algorithm analysis in Section 4.3.3, except for its time complexity, applies to DPC/R with Rel_2 measure. The time complexity of DPC/R for this case is calculated as follows. Recall that, without using any order criterion, DPC/R-1 requires $O(\Omega\times nr\times|V|^4+m\times|E|\times\check{R}_{min})$; see Section 4.3.3.3. Since Dijkstra's algorithm requires a time complexity of $O(|V|^2\times\log|V|)$, calculating all w_j's takes $O(|E|)$, and Yen's algorithm requires $O(m\times|V|\times(|E|+|V|\times\log|V|))$ to generate *m* paths of the network and therefore DPC/R-1 requires in total $O((\Omega\times nr\times|V|^4+m\times|E|\times\check{R}_{min})+(|V|^2\times\log|V|+|E|+m\times|V|\times(|E|+|V|\times\log|V|))=O(\Omega\times m\times|V|^4)$ time. For the second improvement, we propose using Yen's algorithm [46] to generate only the first $k\geq1$ least weighted paths in order. Thus, this improvement does not

require all (*s*, *t*) paths a priori, which improves DPCR-1's time complexity while producing almost the same result as compared to using all paths in the network, this version of DPC/R-1, called DPC/R-2. Since there is no effective way for setting the value of *k* a priori, we set *k* dynamically following the method in Section 4.3.5. Since Yen's algorithm requires a time complexity of $O(|V|^2\times\log|V|+|E|+k\times|V|\times(|E|+|V|\times\log|V|))$ to generate the first *k* least weighted paths, DPC/R-2 has a time complexity of $O((\Omega\times nr\times|V|^4+k\times|E|\times\check{R}_{min})+((|V|^2\times\log|V|+|E|+k\times|V|\times(|E|+|V|\times\log|V|)))|))=O(\Omega\times(|V|^4))$, since *k* and *nr* are constants, $|E|\leq|V|^2$ and $\Omega\leq k\times\check{R}_{min}$**.**

Appendix C

Simulation data

C.1 The Topology of Networks in Table 2.3 and 2.4

Fig C.1 shows the configuration of each of the 20 CNs described in Tables 2.3 and 2.4 of Section 2.6.1. For each $CN_n^{|V|,|E|}$ and $CN_m^{|V|,|E|}$, a tuple ((*x*, *y*), c_j, r_j, b_j) in Fig. C.1 denotes a link between nodes *x* and y ((*x*, *y*)), and its cost (c_j), reliability (r_j), and bandwidth (b_j); notice that *n* and *m* are the number of spanning trees and (*s*, *t*) paths respectively.

Topology: $CN_8^{4,5}$ (Table 2.3) and $CN_4^{4,5}$(Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((2,3),2,0.7,6), ((2,4),4,0.9,5), ((3,4),6,0.9,4)
Topology: $CN_{21}^{5,8}$ (Table 2.3) and $CN_9^{5,8}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),2,0.7,6), ((1,4),3,0.6,3), ((2,4),4,0.9,5), ((2,5),4,0.6,4), ((3,4),6,0.9,4), ((3,5),2,0.7,6), ((4,5),3,0.9,8)
Topology:$CN_{21}^{6,8}$ (Table 2.3) and $CN_7^{6,8}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((2,4),2,0.7,6), ((2,5),4,0.9,5), ((3,5),6,0.9,4), ((4,5),4,0.6,4), ((4,6),2,0.7,6), ((5,6),3,0.9,8)
Topology:$CN_{55}^{6,9}$ (Table 2.3) and $CN_{13}^{6,9}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((2,3),2,0.7,6), ((2,4),4,0.9,5), ((3,4),6,0.9,4), ((3,5),4,0.6,4), ((5,4),2,0.7,6), ((4,6),3,0.9,8), ((5,6),2,0.9,6)
Topology:$CN_{368}^{7,12}$ (Table 2.3) and $CN_{25}^{7,12}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((2,5),3,0.6,3), ((2,7),2,0.7,6), ((1,3),4,0.9,5), ((3,5),6,0.9,4), ((5,7),4,0.6,4), ((1,4),2,0.7,6), ((4,3),3,0.9,8), ((3,6),2,0.9,6), ((5,6),4,0.9,3), ((4,6),3,0.9,5), ((6,7),2,0.9,7)
Topology:$CN_{1033}^{7,15}$ (Table 2.3) and $CN_{14}^{7,15}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((1,4),2,0.7,6), ((3,2),4,0.9,5), ((3,5),6,0.9,4), ((3,6),4,0.6,4), ((3,4),2,0.7,6), ((2,5),3,0.9,8), ((5,6),2,0.9,6), ((5,1),4,0.9,3), ((4,6),3,0.9,5), ((2,7),2,0.9,7), ((5,7),3,0.9,13), ((6,7),4,0.9,9), ((4,7),3,0.9,7)
Topology:$CN_{247}^{8,12}$ (Table 2.3) and $CN_{20}^{8,12}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((3,2),2,0.7,6), ((4,2),4,0.9,5), ((3,4),6,0.9,4), ((3,6),4,0.6,4), ((4,5),2,0.7,6), ((7,5),3,0.9,8), ((5,6),2,0.9,6), ((7,8),4,0.9,3), ((8,6),3,0.9,5), ((4,7),2,0.9,7)

Topology:$CN_{256}^{8,12}$ (Table 2.3) and $CN_{24}^{8,12}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((2,3),3,0.6,3), ((1,3),2,0.7,6), ((1,4),4,0.9,5), ((3,4),6,0.9,4), ((3,7),4,0.6,4), ((4,6),2,0.7,6), ((4,5),3,0.9,8), ((5,6),2,0.9,6), ((5,8),4,0.9,3), ((7,8),3,0.9,5), ((8,7),2,0.9,7)
Topology:$CN_{576}^{8,13}$ (Table 2.3) and $CN_{29}^{8,13}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((1,4),2,0.7,6), ((2,3),4,0.9,5), ((3,4),6,0.9,4), ((2,5),4,0.6,4), ((3,6),2,0.7,6), ((4,7),3,0.9,8), ((5,6),2,0.9,6), ((6,7),4,0.9,3), ((5,8),3,0.9,5), ((8,7),2,0.9,7), ((6,8),3,0.9,13)
Topology:$CN_{171}^{9,12}$(Table 2.3) and $CN_{13}^{9,12}$ (Table 2.4) Links: ((1,5),5,0.9,8), ((1,2),3,0.6,3), ((2,3),2,0.7,6), ((3,4),4,0.9,5), ((4,5),6,0.9,4), ((7,5),4,0.6,4), ((3,6),2,0.7,6), ((7,6),3,0.9,8), ((3,8),2,0.9,6), ((7,8),4,0.9,3), ((7,9),3,0.9,5), ((8,9),2,0.9,7)
Topology: $CN_{327}^{9,13}$ (Table 2.3) and $CN_{18}^{9,13}$ (Table 2.4) Links: ((3,4),5,0.9,8), ((2,3),3,0.6,3), ((1,2),2,0.7,6), ((1,6),4,0.9,5), ((6,5),6,0.9,4), ((4,5),4,0.6,4), ((3,7),2,0.7,6), ((2,9),3,0.9,8), ((9,8),2,0.9,6), ((7,8),4,0.9,3), ((8,2),3,0.9,5), ((5,7),2,0.9,7), ((3,8),3,0.9,13)
Topology:$CN_{647}^{9,14}$ (Table 2.3) and $CN_{44}^{9,14}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((2,3),2,0.7,6), ((2,4),4,0.9,5), ((2,5),6,0.9,4), ((3,5),4,0.6,4), ((5,4),2,0.7,6), ((4,8),3,0.9,8), ((4,6),2,0.9,6), ((5,7),4,0.9,3), ((6,8),3,0.9,5), ((6,7),2,0.9,7), ((8,7),3,0.9,13), ((8,9),4,0.9,9)
Topology: $CN_{2112}^{10,21}$ (Table 2.3) and $CN_{64}^{10,21}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((1,4),2,0.7,6), ((5,10),4,0.9,5), ((8,10),6,0.9,4), ((9,10),4,0.6,4), ((2,5),2,0.7,6), ((2,6),3,0.9,8),((2,3),2,0.9,6), ((7,3),4,0.9,3), ((4,9),3,0.9,5), ((4,3),2,0.9,7), ((5,6),3,0.9,13), ((8,9),4,0.9,9), ((9,7),3,0.9,7), ((6,3),4,0.9,8), ((5,8),3,0.9,10), ((4,7),3,0.9,7), ((3,8),2,0.9,11), ((7,8),4,0.9,8),((7,8),2,0.8,13)
Topology: $CN_{1598}^{11,21}$ (Table 2.3) and $CN_{18}^{11,21}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,4),3,0.6,3), ((1,6),2,0.7,6), ((1,8),4,0.9,5), ((2,4),6,0.9,4), ((2,3),4,0.6,4), ((3,4),2,0.7,6), ((10,3),4,0.9,3), ((4,10),3,0.9,5), ((4,5),2,0.9,7), ((5,6),3,0.9,13), ((6,7),4,0.9,9), ((8,7),3,0.9,7), ((7,9),4,0.9,8), ((9,11)2,0.8,13), ((5,7),3,0.9,10), ((5,11),3,0.9,7), ((4,6),3,0.9,8),((6,8),2,0.9,6), ((9,5),2,0.9,11), ((10,11),4,0.9,8)
Topology: $CN_{3666}^{13,22}$ (Table 2.3) and $CN_{281}^{13,22}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((2,3),3,0.6,3), ((2,5),2,0.7,6), ((1,3),4,0.9,5), ((3,6),6,0.9,4), ((6,9),4,0.6,4), ((9,11),2,0.7,6), ((9,10),3,0.9,8),((6,10),2,0.9,6), ((5,6),4,0.9,3), ((4,5),3,0.9,5), ((4,3),2,0.9,7), ((8,4),3,0.9,13), ((7,3),4,0.9,9), ((8,7),3,0.9,7), ((8,12),4,0.9,8), ((11,12),3,0.9,10), ((12,7),3,0.9,7), ((13,11),2,0.9,11), ((7,13),4,0.9,8), ((10,12),2,0.8,13), ((1,4),3,0.7,9)

Topology: $CN_{7683}^{16,30}$ (Table 2.3) and $CN_{36}^{16,30}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,4),3,0.6,3), ((1,7),2,0.7,6), ((1,9),4,0.9,5), ((1,13),6,0.9,4), ((3,2),4,0.6,4), ((2,4),2,0.7,6), ((4,7),3,0.9,8),(9,7),2,0.9,6), ((9,13),4,0.9,3), ((15,3),3,0.9,5), ((3,4),2,0.9,7), ((4,15),3,0.9,13), ((5,4),4,0.9,9), ((6,5),3,0.9,7), ((7,6),4,0.9,8), ((7,8),3,0.9,10), ((8,9),3,0.9,7), ((8,6),2,0.9,11), ((10,9),4,0.9,8),((10,8),2,0.8,13), ((9,12),3,0.7,9), ((9,14),2,0.9,11), ((12,14),4,0.9,8), ((6,16),3,0.6,7),((12,6),4,0.9,12), ((6,11),3,0.7,6), ((12,11),4,0.8,9), ((15,16),2,0.7,7), ((12,16),3,0.9,10)
Topology: $CN_{9471}^{17,25}$ (Table 2.3) and $CN_{136}^{17,25}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((1,4),2,0.7,6), ((2,5),4,0.9,5), ((2,6),6,0.9,4), ((3,7),4,0.6,4), ((3,8),2,0.7,6), ((4,9),3,0.9,8),((4,10),2,0.9,6), ((6,7),4,0.9,3), ((8,9),3,0.9,5), ((5,10),2,0.9,7), ((5,11),3,0.9,13), ((12,16),4,0.9,9), ((7,13),3,0.9,7), ((14,8),5,0.9,8), ((9,15),3,0.9,10), ((10,16),3,0.9,7), ((11,12),4,0.9,11), ((13,14),4,0.9,7),((15,16),2,0.8,13), ((11,17),3,0.7,9), ((12,17),2,0.9,11), ((14,17),4,0.9,8), ((17,16),3,0.6,7)
Topology: $CN_{21456}^{18,27}$ (Table 2.3) and $CN_{282}^{18,27}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,7),3,0.6,3), ((1,3),2,0.7,6), ((2,3),4,0.9,5), ((2,6),6,0.9,4), ((3,4),4,0.6,4), ((4,7),2,0.7,6), ((3,5),3,0.9,8),((3,17),2,0.9,6), ((5,6),4,0.9,3), ((6,7),3,0.9,5), ((8,5),2,0.9,7), ((8,11),3,0.9,13), ((11,15),4,0.9,9), ((5,9),3,0.9,7), ((9,11),5,0.9,8), ((9,12),3,0.9,10), ((12,16),3,0.9,7), ((10,7),2,0.9,11), ((10,12),4,0.9,7), ((10,13),2,0.8,13), ((13,14),3,0.7,9), ((12,17),4,0.9,11), ((14,17),4,0.9,8), ((5,16),3,0.6,7), ((15,18),4,0.9,12), ((17,18),3,0.7,6)
Topology: $CN_{24173}^{20,30}$ (Table 2.3) and $CN_{780}^{20,30}$(Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((1,4),2,0.7,6), ((2,5),4,0.9,5), ((2,6),6,0.9,4), ((3,7),4,0.6,4), ((3,8),2,0.7,6), ((4,9),3,0.9,8), ((4,10),2,0.9,6), ((5,10),4,0.9,3), ((5,11),3,0.9,5), ((6,12),2,0.9,7), ((6,7),3,0.9,13), ((13,7),4,0.9,9), ((8,14),3,0.9,7), ((9,8),4,0.9,8), ((9,15),3,0.9,10), ((10,16),3,0.9,7), ((11,17),2,0.9,11), ((11,12),5,0.9,8),((12,18),2,0.8,13), ((13,18),3,0.7,9), ((13,14),4,0.9,11), ((19,14),4,0.9,7), ((15,19),3,0.6,7), ((15,16),4,0.9,12), ((16,17),3,0.7,6), ((17,20),4,0.8,9), ((18,20),2,0.7,7), ((19,20),3,0.9,10)
Topology: $CN_{18257}^{21,26}$ (Table 2.3) and $CN_{44}^{21,26}$ (Table 2.4) Links: ((1,2),5,0.9,8), ((1,3),3,0.6,3), ((2,3),2,0.7,6), ((2,5),4,0.9,5), ((2,4),6,0.9,4), ((3,7),4,0.6,4), ((5,6),2,0.7,6), ((8,5),3,0.9,8),((9,5),2,0.9,6), ((7,11),4,0.9,3), ((10,12),3,0.9,5), ((10,13),2,0.9,7), ((12,16),3,0.9,13), ((17,21),4,0.9,9), ((4,7),3,0.9,7), ((7,6),5,0.9,8), ((10,9),3,0.9,10), ((12,11),3,0.9,7), ((14,8),4,0.9,11), ((14,17),4,0.9,7),((15,13),2,0.8,13), ((16,18),3,0.7,9), ((18,19),2,0.9,11), ((19,20),4,0.9,8), ((17,15),3,0.6,7),((21,20),4,0.9,12)

Fig. C.1: Topology configuration for the 20 CN's in Table 2.3 and 2.4

C.2 Cost matrices for the 76 networks from [12]

Let $K_{|V|}$ denote a fully connected network with |V| nodes. Note that $K_{|V|}$ has |V|(|V|-1)/2 links. The following shows the cost matrices of K_6, K_7, K_8, K_9, K_{10}, K_{11} that are obtained from [12]. Section 2.6.1 describes how to use the 26 matrices to generate the 76 networks in [12].

Cost Matrices for K_6:

Matrix 1

	1	2	3	4	5	6
1	-	3	23	31	62	38
2		-	12	5	80	46
3			-	39	85	34
4				-	95	72
5					-	84
6						-

Matrix 2

	1	2	3	4	5	6
1	-	22	32	56	79	79
2		-	81	80	100	50
3			-	75	47	33
4				-	42	71
5					-	4
6						-

Matrix 3

	1	2	3	4	5	6
1	-	96	96	21	27	48
2		-	90	47	37	41
3			-	72	30	24
4				-	79	68
5					-	59
6						-

Matrix 4

	1	2	3	4	5	6
1	-	27	24	48	19	74
2		-	48	76	84	95
3			-	21	61	31
4				-	80	5
5					-	80
6						-

Matrix 5

	1	2	3	4	5	6
1	-	52	88	68	96	45
2		-	21	8	94	5
3			-	55	79	50
4				-	12	25
5					-	5
6						-

Cost Matrices for K_7

Matrix 1

1	2	3	4	5	6	7

Matrix 2

1	2	3	4	5	6	7

1	-	65	53	35	61	34	21
2		-	58	62	44	73	95
3			-	30	17	25	30
4				-	4	10	3
5					-	45	22
6						-	16
7							-

1	-	32	87	40	70	25	16
2		-	99	32	95	79	65
3			-	27	77	22	60
4				-	49	56	2
5					-	40	13
6						-	42
7							-

Matrix 3

	1	2	3	4	5	6	7
1	-	16	52	83	79	39	30
2		-	86	64	25	71	46
3			-	24	34	48	8
4				-	78	75	40
5					-	52	67
6						-	84
7							-

Matrix 4

	1	2	3	4	5	6	7
1	-	28	7	14	37	49	96
2		-	7	94	24	11	51
3			-	2	95	24	23
4				-	35	54	25
5					-	36	88
6						-	16
7							-

Matrix 5

	1	2	3	4	5	6	7
1	-	52	54	11	2	80	85
2		-	7	72	22	29	98
3			-	20	92	36	39
4				-	94	56	69
5					-	15	66
6						-	8
7							-

Cost Matrices for K_8

Matrix 1

	1	2	3	4	5	6	7	8
1	-	59	19	98	77	35	40	93
2		-	68	39	16	48	12	81
3			-	17	41	24	89	41
4				-	60	23	72	45
5					-	23	51	84
6						-	54	1

Matrix 2

	1	2	3	4	5	6	7	8
1	-	35	47	82	8	38	29	7
2		-	70	25	45	87	67	62
3			-	47	9	47	34	14
4				-	64	58	5	82
5					-	81	75	85
6						-	61	33

7		- 33
8		-

7		- 54
8		-

Matrix 3

	1	2	3	4	5	6	7	8
1	-	19	91	21	21	34	81	76
2		-	53	14	86	2	87	65
3			-	7	67	24	25	41
4				-	50	32	25	5
5					-	61	99	69
6						-	70	81
7							-	46
8								-

Matrix 4

	1	2	3	4	5	6	7	8
1	-	29	50	45	96	93	85	28
2		-	69	73	93	71	88	52
3			-	50	18	31	41	29
4				-	21	40	46	20
5					-	65	79	20
6						-	69	9
7							-	42
8								-

Matrix 5

	1	2	3	4	5	6	7	8
1	-	49	48	33	38	100	21	4
2		-	48	68	66	34	5	35
3			-	60	68	85	11	76
4				-	59	31	4	5
5					-	91	9	92
6						-	91	23
7							-	9
8								-

Cost Matrices for K_9

Matrix 1

	1	2	3	4	5	6	7	8	9
1	-	37	77	61	97	58	41	63	3
2		-	40	30	4	53	61	37	63
3			-	56	63	71	13	90	34
4				-	33	70	39	7	35

Matrix 2

	1	2	3	4	5	6	7	8	9
1	-	36	68	98	5	1	98	85	85
2		-	13	87	79	52	62	91	72
3			-	73	6	82	48	89	42
4				-	20	38	49	17	81

5			-	89	55	97	65
6				-	23	57	88
7					-	2	70
8						-	77
9							-

5			-	7	65	58	18
6				-	21	20	62
7					-	69	92
8						-	8
9							-

Matrix 3

	1	2	3	4	5	6	7	8	9
1	-	51	44	76	96	61	2	96	42
2		-	93	91	98	47	39	72	74
3			-	82	52	14	8	88	33
4				-	78	31	11	28	42
5					-	8	67	66	55
6						-	5	36	40
7							-	57	33
8								-	12
9									-

Matrix 4

	1	2	3	4	5	6	7	8	9
1	-	45	58	64	26	78	33	72	14
2		-	21	70	11	3	86	53	97
3			-	89	56	76	39	39	54
4				-	27	94	22	52	7
5					-	51	81	5	90
6						-	3	20	29
7							-	43	83
8								-	58
9									-

Matrix 5

	1	2	3	4	5	6	7	8	9
1	-	99	25	52	67	12	66	10	40
2		-	49	29	99	25	48	51	58
3			-	27	23	52	5	87	47
4				-	6	14	48	66	80
5					-	13	69	72	73
6						-	41	43	68
7							-	10	25
8								-	79
9									-

Cost Matrices for K_{10}

Matrix 1

	1	2	3	4	5	6	7	8	9	10
1	-	8	95	51	5	10	8	88	100	86
2		-	97	37	5	9	57	15	29	20

Matrix 2

	1	2	3	4	5	6	7	8	9	10
1	-	24	26	69	25	48	3	82	45	98
2		-	12	75	22	33	82	54	4	82

3		-	66	23	21	25	51	16	28
4			-	2	39	4	18	72	82
5				-	7	86	72	61	31
6					-	53	81	12	39
7						-	11	7	28
8							-	10	87
9								-	51
10									-

3		-	30	8	75	38	21	79	23
4			-	67	18	64	50	78	12
5				-	72	92	94	21	96
6					-	5	81	18	84
7						-	19	37	34
8							-	7	98
9								-	50
10									-

Matrix 3

	1	2	3	4	5	6	7	8	9	10
1	-	25	68	72	76	91	62	64	91	21
2		-	58	95	61	17	38	22	82	62
3			-	68	83	89	98	61	71	21
4				-	43	34	33	28	53	26
5					-	26	96	3	3	94
6						-	33	9	90	37
7							-	17	17	69
8								-	72	64
9									-	66
10										-

Matrix 4

	1	2	3	4	5	6	7	8	9	10
1	-	77	35	17	78	93	22	28	41	10
2		-	4	36	43	18	97	84	20	89
3			-	8	97	43	53	85	59	86
4				-	12	84	99	42	14	80
5					-	67	32	18	79	91
6						-	98	28	42	60
7							-	61	80	87
8								-	78	73
9									-	98
10										-

Matrix 5

	1	2	3	4	5	6	7	8	9	10
1	-	37	77	43	60	80	22	66	10	52
2		-	8	19	60	27	58	74	4	56
3			-	84	92	67	51	43	82	83
4				-	95	94	10	97	18	21
5					-	53	90	60	34	73
6						-	59	7	100	87
7							-	80	73	47
8								-	35	62
9									-	67
10										-

Cost Matrices for K_{11}

Matrix 1

	1	2	3	4	5	6	7	8	9	10	11
1	-	9	37	33	7	14	42	47	25	73	86
2		-	98	79	99	46	46	16	98	47	31

3		-	81	49	28	50	22	26	11	74
4			-	47	17	92	98	66	65	37
5				-	42	14	55	22	65	93
6					-	23	62	57	30	57
7						-	30	3	94	25
8							-	36	53	49
9								-	21	39
10									-	43
11										-

Bibliography

[1] K. Bhupesh, S. Makarand and B. Misra, "Optimal Reliability Design of a System," Handbook *of Performability Engineering, K.B. Misra (Editor), Springer Verlag, London, 2008.*

[2] B. Dengue, F. Altiparmak and E. Smith, "A Genetic Algorithm Approach to Optimal Topological Design of All Terminal Networks," *Proceedings of the Artificial Neural Networks in Engineering Conference ANNIE`95, vol. 5*, pp. 405-410, 1995.

[3] L. Gen, "A Self-controlled Genetic Algorithm for Reliable Communication Network Design," *Evolutionary Computation IEEE Congress*, pp. 640-647, 2006.

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

[5] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *W. H. Freeman and Company*, San Francesco, 1979.

[6] D. Deeter, and E. Smith, "Economic Design of Reliable Networks," *IIE Transactions, vol*. 30, pp. 1161-1174, 1998.

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

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

[9] J. Won, and F. Karray, "Cumulative Update of All-Terminal Reliability for Faster Feasibility Decision," *IEEE Trans. Reliability, vol.* 59, *no.* 3, pp. 551-562, 2010.

[10] S. Yeh, J.S. Lin, and W.C. Yeh, "New Monte Carlo Method for Estimating Network Reliability," *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, pp. 723-726, 1994.

[11] E.K. Burke, "Multi-Objective Optimization,"*Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer Science+Business Media*, 2014.

[12] F. Altiparmak, B. Dengiz, and O. Belgin, "Design of Reliable Communication Networks: A Hybrid Ant Colony Optimization Approach for the Design of Reliable Networks," *IIE Transactions, vol*. 42, pp. 273-287, 2010.

[13] B. Dengiz, F. Altiparmak, and A.E. Smith, "Efficient Optimization of All Terminal Reliable Networks Using an Evolutionary Approach," *IEEE Trans. Reliability, vol.* 46, *no*. 1, pp. 18-26, 1997.

[14] B. Dengiz, F. Altiparmak, and A.E. Smith, "Local Search Genetic Algorithm for Optimal Design of Reliable Networks," *IEEE Trans. Evolutionary Computation, vol*. 1, *no.* 3, pp. 179-188, 1997.

[15] G. Hardy, C. Lucet, and N. Limnios, "A BDD-Based Heuristic Algorithm for Design of Reliable Networks with Minimal Cost," *International Conference on Mobile Ad Hoc and Sensor Networks*, pp. 13-15, 2006.
[16] K. Aggarwal, Y. Chopra, and J. Bajwa, "Topological Layout of Links for Optimizing the S-T Reliability in a Computer Communication System," *Microelectronics Reliability, vol*. 22, *no*. 3, pp. 341-345, 1982.
[17] K. Aggarwal, Y. Chopra, and J. S. Bajwa, "Topological Layout of Links for Optimizing the Overall Reliability in a Computer Communication System," *Microelectronics Reliability, vol.*22, pp.347-351, 1982.
[18] F. Shao, X. Shen, and P. Ho, "Reliability Optimization of Distributed Access Networks with Constrained Total Cost," *IEEE Trans. Reliability, vol*. 54, *no*. 3, pp. 421-430, 2005.
[19] A. Zakir, and M. Abd-El-Barr, "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*, pp. 123-128, 2002.
[20] M. Desjarlais, and R. Molina, "Counting Spanning Trees in Grid Graphs," *Congressus Numerantium, vol*. 145, pp. 177-185, 2000.
[21] K.Aggarwal, Y. Chopra, and J.Bajwa, "Modification of Cutsets for Reliability Evaluation of Communication Systems," *Microelectronics Reliability, vol.* 22, *no*. 3, pp. 337-343, 1982.
[22] S. Rai, and S. Soh, "CAREL: Computer Aided Reliability Evaluator for Distributed Computer Networks," *IEEE Trans. Parallel and Distributed Systems, vol*. 2, pp. 199-213, 1991.
[23] K. Aggarwal, J. Gupta, and K. Misra, "A Simple Method for Reliability Evaluation of a Communication System," *IEEE Trans. Comm., vol*. 23, *no*. 5, pp. 563-566, 1975.
[24] S. Soh, S. Rai, and R. Brooks, "Performability Issues in Wireless Communication Network," Handbook *of Performability Engineering, Indian Institute of Technology, K.B. Misra (Editor), Springer Verlag, London, 2008.*
[25] Y. Niu, and F. Shao, "A Practical Bounding Algorithm for Computing Two-Terminal Reliability Based on Decomposition Technique," *Computers and Mathematics with Applications, vol.* 61, *no*. 8, pp. 2241-2246, 2011.
[26] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithm*, Cambridge, Massachusetts: The MIT Press, 2009.
[27] Alice Yalaoui, Eric Châtelet and Chengbin Chu, "A New Dynamic Programming Method for Reliability & Redundancy Allocation in a Parallel-Series System," *IEEE Transactions on Reliability, vol.*54, pp. 254-261, 2005.
[28] R. Jan, "Design of Reliable Networks," *Computers and Operations Research, vol*. 20, *no*. 1, pp. 25-34, 1993.
[29] T. Koide, S. Shinmori, and H. Ishii, "Topological Optimization with a Network Reliability Constrain," *Discrete Applied Mathematics, vol*. 115, pp. 135-149, 2001.
[30] H. M. F. Abo ElFotoh, and L. S. Al-Sumait, "A Neural Approach to Topological Optimization of Communication Networks, with Reliability Constraints," *IEEE Trans. Reliability, vol*. 50, pp. 397-408, 2001.

[31] F. Altiparmak, and B. Dengiz, "Cross Entropy Approach to Design of Reliable Networks," *European Journal of Operation Research*, *vol*. 199, pp. 542-552, 2009.
[32] B. Dengiz, C. Alabas, and O. Dengiz, "A Tabu Search Algorithm for Neural Networks Training," *Journal of Operation Research*, *vol.* 60, *no.* 2, pp. 282-291, 2007.
[33] A. Kumar, R.M. Pathak, and Y.P. Gupta, "A Genetic Algorithm for Distributed System Topology Design," *Computers and Industrial Engineering*, *vol.* 28, pp. 659-670, 1995.
[34] J. Marquez, and C. Rocco, "All-Terminal Network Reliability Optimization via Probabilistic Solution Discovery," *Reliability Engineering and System Safety*, *vol*. 93, pp. 1689-1697, 2008.
[35] 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.
[36] K. Watcharasitthiwat, S. Pothiya, and P. Wardkein, "Multiple Tabu Search Algorithm for Solving the Topology Network Design," *Local Search Techniques: Focus on Tabu Search*, I-Tech, pp. 259-264, 2008.
[37] M. Atiqullah, and S. Rao, "Reliability Optimization of Communication Networks Using Simulated Annealing," *Microelectronics and Reliability*, *vol*. 33, *no*. 9, pp. 1303-1319, 1993.
[38] S.A. Khan and A. P. Engelbrecht. A Fuzzy Particle Swarm Optimization Algorithm for Computer Communication Network Topology design. *Appl Intell*. *vol*. 36, *no.* 1, pp. 161-177, 2012.
[39] S. Duarte and B. Bar´an, "Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms," *XXVII Conferencia Latinoamericana de Informatica CLEI'2001,* 2001.
[40] R. Kumar, P. P. Parida, and M. Gupta, "Topological Design of Communication Networks using Multiobjective Genetic Optimization," *Proc. CEC-02*, pp. 425-430, 2002.
[41] N. Banerjee and R. Kumar, "Multiobjective Network Design for Realistic Traffic Models," *Proc. GECCO*, pp. 1904-1911, 2007.
[42] J. Figueira and A.Rong, "Dynamic Programming Algorithms for the Bi-Objective Integer knapsack Problem," *European Journal of Operational Research*, *vol*. 236, *no*. 1, pp. 85-99, 2014.
[43] Martello, D. Pisinger, and P. Toth, "Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem," *Management Science*, *vol*. 45, pp. 414-424, 1999.
[44] B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu, "A Dynamic Programming Algorithm for Reliable Network Design," *IEEE Trans. Reliability*, *vol*. 63, *no.* 2, pp. 443-454, 2014.
[45] B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu, "A Practical Algorithm for Reliable Communication Network Design," *International Journal of Performability Engineering*, *vol*. 9, *no*. 4, pp. 397-408, 2013.
[46] J. Y. Yen, "Finding the k Shortest Loopless Paths in a Network," *Management Science*, *vol*. 17, *no*. 11, 712-716, 1971.

[47] B. Elshqeirat, S. Soh, S. Rai, M. Lazarescu, "Dynamic Programming for Minimal Cost Topology with Reliability Constraint," *Proceedings of the 5th International Conference on Communication Software and Networks*, 2013.
[48] B. Elshqeirat, S. Soh, S. Rai, M. Lazarescu, "Topology Design with Minimal Cost Subject to Network Reliability Constraint," *IEEE Trans. Reliability*. In press.
[49] S. Kuo, S. Lu, and F. Yeh, "Determining Terminal Pair Reliability Based on Edge Expansion Diagrams using OBDD," *IEEE Trans. Reliability, vol*. 48, *no*. 3, pp. 234-246, 1999.
[50] G. Hardy, C. Lucet, and N. Limnios, "K-Terminal Network Reliability Measures with Binary Decision Diagrams," *IEEE Trans on Reliability, vol*. 56, *no*. 3, pp. 506-515, 2007.
[51] R. Loh, S. Soh and M. Lazarescu, "Edge Disjoint Paths with Minimum Delay Subject to Reliability Constraint," *Proc. IEEE APCC*, China, 2009.
[52] M. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science, vol*. 5, *no.* 12, pp. 1861-1871, 2004.
[53] J. B. Orlin, "Max Flows in O(nm) Time, or Better," *ACM Symposium on Theory of Computing. ACM*, 2013.

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE.Please read the form carefully and keep a copy for your files.

TITLE OF PAPER/ARTICLE/REPORT, INCLUDING ALL CONTENT IN ANY FORM, FORMAT, OR MEDIA (hereinafter, "The Work"):**Dynamic Programming for Minimal Cost Topology with Two Terminal Reliability Constraint**

COMPLETE LIST OF AUTHORS:**Basima Elshqeirat, Suresh Rai, Mihai M Lazarescu and Sieteng Soh**

IEEE PUBLICATION TITLE (Journal, Magazine, Conference, Book):**19th Asia-Pacific Conference on Communications (APCC 2013)**

COPYRIGHT TRANSFER

1. The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers,Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the above Work,including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

CONSENT AND RELEASE

2. ln the event the undersigned makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the undersigned, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his,&er presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction ofthe Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast ofthe Presentation during or after the conference.

3. In connection with the permission granted in Section 2, the undersigned hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

4. The undersigned hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the undersigned has obtained any necessary permissions. Where necessary, the undersigned has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE.

[ ] Please check this box if you do not wish to have video/audio recordings made of your conference presentation.

See below for Retained Rights/Terms and Conditions, and Author Responsibilities.

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE

PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEEs publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/pub_tools_policies.html. Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

General

1. Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.

2. Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.

3. In the case of a Work performed under a U.S. Government contract or grant, the IEEE recognizes that the U.S. Government has royalty-free permission to reproduce all or portions of the Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract/grant so requires.

4. Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.

5. Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

Author Online Use

6. Personal Servers. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore.Authors shall not post the final, published versions of their papers.

7. Classroom or Internal Training Use. An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the authors personal web site or the servers of the authors institution or company in connection with the authors teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.

8. Electronic Preprints. Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employers site, or to another server that invites constructive comment from colleagues.Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

INFORMATION FOR AUTHORS

IEEE Copyright Ownership

It is the formal policy of the IEEE to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein, in order to protect the interests of the IEEE, its authors and their employers, and, at the same time, to facilitate the appropriate re-use of this material by others.The IEEE distributes its technical publications throughout the world and does so by various means such as hard copy, microfiche, microfilm, and electronic media.It also abstracts and may translate its publications, and articles contained therein, for inclusion in various compendiums, collective works, databases and similar publications.

Author/Employer Rights

If you are employed and prepared the Work on a subject within the scope of your employment, the copyright in the Work belongs to your employer as a work-for-hire. In that case, the IEEE assumes that when you sign this Form, you are authorized to do so by your employer and that your employer has consented to the transfer of copyright, to the representation and warranty of publication rights, and to all other terms and conditions of this Form. If such authorization and consent has not been given to you, an authorized representative of your employer should sign this Form as the Author.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to identify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing grant of rights shall become null and void and all materials embodying the Work submitted to the IEEE will be destroyed.
4. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.

Basima — **03-07-2013**

Author/Authorized Agent For Joint Authors — **Date(dd-mm-yy)**

THIS FORM MUST ACCOMPANY THE SUBMISSION OF THE AUTHOR'S MANUSCRIPT.
Questions about the submission of the form or manuscript must be sent to the publication's editor.Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966 (telephone)

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

TITLE OF PAPER/ARTICLE/REPORT, INCLUDING ALL CONTENT IN ANY FORM, FORMAT, OR MEDIA (hereinafter, "the Work"): TR2013-141

"Topology Design with Minimal Cost Subject to Network Reliability Constraints"

COMPLETE LIST OF AUTHORS:

Basima Elshqeirat
Sieteng Soh
Suresh Rai
Mihai Lazarescu

IEEE PUBLICATION TITLE (Journal, Magazine, Conference, Book): IEEE Transactions on Reliability

COPYRIGHT TRANSFER

1. The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the above Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

CONSENT AND RELEASE

2. In the event the undersigned makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the undersigned, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.

3. In connection with the permission granted in Section 2, the undersigned hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

4. The undersigned hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the undersigned has obtained any necessary permissions. Where necessary, the undersigned has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE.

☐ Please check this box if you do not wish to have video/audio recordings made of your conference presentation.

See reverse side for Retained Rights/Terms and Conditions, and Author Responsibilities.

GENERAL TERMS

- The undersigned represents that he/she has the power and authority to make and execute this assignment.
- The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
- In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall become null and void and all materials embodying the Work submitted to the IEEE will be destroyed.
- For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.

(1)_____ May, 6th, 2013
Author/Authorized Agent for Joint Authors **Date**

U.S. GOVERNMENT EMPLOYEE CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are U.S. government employees and prepared the Work on a subject within the scope of their official duties. As such, the Work is not subject to U.S. copyright protection.

(2)_____ _____
Authorized Signature **Date**

(Authors who are U.S. government employees should also sign signature line (1) above to enable the IEEE to claim and protect its copyright in international jurisdictions.)

CROWN COPYRIGHT CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties. As such, the Work is subject to Crown Copyright and is not assigned to the IEEE as set forth in the first sentence of the Copyright Transfer Section above. The undersigned acknowledges, however, that the IEEE has the right to publish, distribute and reprint the Work in all forms and media.

(3)_____ _____
Authorized Signature **Date**

(Authors who are British or British Commonwealth Government employees should also sign line (1) above to indicate their acceptance of all terms

RAMS Consultants, India

Website: www.ramsconsultants.org

International Journal of Performability Engineering
(http://www.ijpe-online.com)

TRANSFER OF COPYRIGHT AGREEMENT

The transfer of copyright from author to publisher must be clearly stated in writing to enable the publisher to assure maximum dissemination of the author's work. Therefore, the following agreement, executed and signed by the author, is required with each manuscript submission. (If the article is a "work made for hire" it must be signed by the employer.)

The article entitled A PRACTICAL ALGORITHM FOR RELIABLE NETWORK TOPOLOGY DESIGN

Manuscript # : IJPE 443: 12
is herewith submitted for publication in the **International Journal of Performability Engineering**.

This paper has not been published before, and it is not under consideration for publication in any other journals. It contains no matter that is scandalous, obscene, libelous, or otherwise contrary to law. When the article is accepted for publication, I, as the author (U.S. Government employees: see bottom of page), hereby agree to transfer to **RAMS Consultants, India,** all rights, including those pertaining to electronic forms and transmissions, under existing copyright laws, except for the following, which the author(s) specifically retain(s):

1. The right to make further copies of all or part of the published article for my use in classroom teaching or for any other;
2. The right to reuse all or part of this material in a compilation of my own works or in a textbook of which I am the author;
3. The right to make copies of the published work for internal distribution within the institution that employs me.

I agree that copies made under these circumstances will continue to carry the copyright notice that appeared in the original published work. I agree to inform my co-authors, if any, of the above terms. I certify that I have obtained written permission for the use of text, tables, and/or illustrations from any copyrighted source(s), and I agree to supply such written permission(s) to **RAMS Consultants, India** upon request.

30/8/2012	
(1) Signature and date	(2) Signature and date
SIETENG SOH, AUTHOR	
(1) Name and title	(2) Name and title
(1) Institution or company (if appropriate)	(2) Institution or company (if appropriate)

Government Copyright
I certify that the above article has been written in the course of the author's employment by the United States Government, so that it is not subject to U.S. copyright laws, or that it has been written in the course of the author's employment by the United Kingdom Government (Crown Copyright).

Signature Date Title

Note to U.S. Government Employees:

- If the above article was not prepared as part of the employee's duties, it is not a U.S. Government work.
- If the above article was prepared jointly, and any co-author is not a U.S. Government employee, it is not a U.S. Government work.

IMPORTANT: Please dispatch this form by **AIRMAIL** after completing it and signing on behalf of all coauthors to: Prof. Krishna B.Misra, Editor-in-Chief, IJPE, 71, Krishna Kutir, Vrindaban Vihar, Mandir Marg, Ajmer Road, Jaipur -302019, Rajasthan, INDIA

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

TITLE OF PAPER/ARTICLE/REPORT, INCLUDING ALL CONTENT IN ANY FORM, FORMAT, OR MEDIA (hereinafter, "the Work"): TR2012-185

"A dynamic programming algorithm for reliable network design"

COMPLETE LIST OF AUTHORS:

Basima Elshqeirat
Sieteng Soh
Suresh Rai
Mihai Lazarescu

IEEE PUBLICATION TITLE (Journal, Magazine, Conference, Book): IEEE Transactions on Reliability

COPYRIGHT TRANSFER

1. The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the above Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

CONSENT AND RELEASE

2 In the event the undersigned makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the undersigned, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.

3. In connection with the permission granted in Section 2, the undersigned hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

4. The undersigned hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the undersigned has obtained any necessary permissions. Where necessary, the undersigned has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE.

☐ Please check this box if you do not wish to have video/audio recordings made of your conference presentation.

See reverse side for Retained Rights/Terms and Conditions, and Author Responsibilities.

GENERAL TERMS

- The undersigned represents that he/she has the power and authority to make and execute this assignment.
- The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
- In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall become null and void and all materials embodying the Work submitted to the IEEE will be destroyed.
- For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.

(1) _____ 27/8/2012
Author/Authorized Agent for Joint Authors **Date**

U.S. GOVERNMENT EMPLOYEE CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are U.S. government employees and prepared the Work on a subject within the scope of their official duties. As such, the Work is not subject to U.S. copyright protection.

(2) _____ _____
Authorized Signature **Date**

(Authors who are U.S. government employees should also sign signature line (1) above to enable the IEEE to claim and protect its copyright in international jurisdictions.)

CROWN COPYRIGHT CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties. As such, the Work is subject to Crown Copyright and is not assigned to the IEEE as set forth in the first sentence of the Copyright Transfer Section above. The undersigned acknowledges, however, that the IEEE has the right to publish, distribute and reprint the Work in all forms and media.

(3) _____ _____
Authorized Signature **Date**

(Authors who are British or British Commonwealth Government employees should also sign line (1) above to indicate their acceptance of all terms

IEEE COPYRIGHT FORM *(continued)*

RETAINED RIGHTS/TERMS AND CONDITIONS

General

1. Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work
2. Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
3. In the case of a Work performed under a U.S. Government contract or grant, the IEEE recognizes that the U.S. Government has royalty-free permission to reproduce all or portions of the Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract/grant so requires
4. Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use. The IEEE Intellectual Property Rights office must handle all such third-party requests
5. Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

Author Online Use

6. **Personal Servers.** Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
7. **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses
8. **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore

INFORMATION FOR AUTHORS

Author Responsibilities

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/pub_tools_policies.html. Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

Author/Employer Rights

If you are employed and prepared the Work on a subject within the scope of your employment, the copyright in the Work belongs to your employer as a work-for-hire. In that case, the IEEE assumes that when you sign this Form, you are authorized to do so by your employer and that your employer has consented to the transfer of copyright, to the representation and warranty of publication rights, and to all other terms and conditions of this Form. If such authorization and consent has not been given to you, an authorized representative of your employer should sign this Form as the Author.

IEEE Copyright Ownership

It is the formal policy of the IEEE to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein, in order to protect the interests of the IEEE, its authors and their employers, and, at the same time, to facilitate the appropriate re-use of this material by others. The IEEE distributes its technical publications throughout the world and does so by various means such as hard copy, microfiche, microfilm, and electronic media. It also abstracts and may translate its publications, and articles contained therein, for inclusion in various compendiums, collective works, databases and similar publications.