School of Electrical Engineering, Computing and Mathematical Sciences

Modelling and Reliability Evaluation of Time Varying Communication Networks

Gaurav Khanna

0000-0002-6846-3437

This thesis is presented for the Degree of Doctor of Philosophy of Curtin University

October 2020

APPROVAL OF THE VIVA-VOCE BOARD

01/03/2021

Certified that the thesis entitled MODELLING AND RELIABILITY EVALUATION OF TIME VARYING COMMUNICATION NETWORKS submitted by GAURAV KHANNA to the Indian Institute of Technology, Kharagpur, and Curtin University, Australia, under IIT Kharagpur-Curtin University Dual Doctoral Degree Programme, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

A/Prof. N. K. Goyal	**Prof. J. Maiti**	**Dr. M. Sarma**
(Member of the DSC)	(Member of the DSC)	(Member of the DSC)
Prof. S. K. Chaturvedi	**Dr. S. Soh**	**A/Prof. M. Lazarescu**
(Supervisor, IIT Kharagpur)	(Co-Supervisor, Curtin University)	(Co-Supervisor, Curtin University)
Prof. S. K. Chaturvedi	**A/Prof. W. Liu**	**Prof. A. K. Tripathi**
(Chairman and HOS)	(Chairperson, Curtin University)	(External Examiner, IIT (BHU) Varanasi)

INDIAN
INSTITUTE OF
TECHNOLOGY
KHARAGPUR-721302

भारतीय
प्रौद्योगिकी
संस्थान
खड़गपुर

SUBIR CHOWDHURY SCHOOL OF QUALITY AND RELIABILITY

CERTIFICATE

This is to certify that the thesis entitled **Modelling and Reliability Evaluation of Time Varying Communication Networks** submitted by **Mr. Gaurav Khanna** to Indian Institute of Technology, Kharagpur and Curtin University, Australia, is a record of bona fide research work under our supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of both the Institutes.

Dr. Sanjay K. Chaturvedi

Professor
Subir Chowdhury School of Quality and Reliability,
Indian Institute of Technology Kharagpur, Kharagpur – 721 302, India

Dr. Sieteng Soh

Senior Lecturer
School of Electrical Engineering, Computing and Mathematical Sciences,
Curtin University, Perth – 6102, Australia

Dr. Mihai Lazarescu

Associate Professor
School of Electrical Engineering, Computing and Mathematical Sciences,
Curtin University, Perth – 6102, Australia

Date: 01/03/2021

Phone: +91/(03222) - 282289 Tele Fax: +91/(03222) - 282290

DECLARATION

I certify that

a. The work contained in this thesis is original and has been done by me under the guidance of my supervisors.
b. The work has not been submitted to any other Institute for any degree or diploma.
c. I have followed the guidelines provided by the Institute in preparing the thesis.
d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
e. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

(**Gaurav Khanna**)

This thesis is fondly dedicated to my parents and beloved sister.

Abstract

Time Varying Communication Networks (TVCNs) like Mobile Ad Hoc Networks (MANETs), Delay Tolerant Networks (DTNs) and Opportunistic Mobile Networks (OMNs) have become an integral part of current communication systems. However, modelling of their parameters like mobility and topology is still a challenging task due to *temporal-topological* variation, which brings new challenges, generally, not observed in static networks.

The degree to which a TVCN is able to provide the required services needs to be quantitatively assessed by defining measures like network reliability, routing requests ratio and route lifetime. Among them, assessment of reliability metrics is very critical as communication links in TVCNs are intermittent and can fail due to channel disruption, transponder failures and energy outages. Though, a plethora of tools/techniques are available to evaluate the reliability of static networks, there is a lack of such metrics and techniques for TVCNs.

In this thesis, our aim is to address the aforementioned challenges by proposing some reliability models and metrics, and designing novel techniques to assess the reliability of TVCNs. More specifically, first, we extend the concepts of *minimal path sets*, *minimal cut sets* and *spanning arborescences*, commonly used to evaluate the reliability of static networks, to TVCNs by considering device-to-device connection in *store-carry-and-forward* manner. We also present algorithm(s) to enumerate them. Second, we develop methodologies to evaluate the reliability related metrics for *store-carry-and-forward* mechanism reliant TVCNs. We evaluate *2-terminal reliability*, *Expected Hop Counts, Expected Slot Counts* and *broadcast reliability* of TVCNs with the help of well-known Sum-of-Disjoint Products technique. Third, we consider TVCNs' topology control, upgrade and contact plan modification problem. We propose a greedy algorithm for TVCN upgrade. Finally, we estimate the 2-terminal reliability of an *end-to-end* connected TVCN, *viz.*, a MANET, via Monte Carlo simulation. Here, we use the concept of *Link Expiration Time* and propose the notion of *Border Time* to determine more accurately the incremental time at which topology changes would occur.

The performance of the proposed algorithms are evaluated against arbitrary TVCNs. The results suggest that the presented approaches can assist a network planner in reliable topology design and efficient management of TVCNs.

Keywords

Delay tolerant networks, Minimal path/cut set, Mobile ad hoc networks, Network reliability, Spanning arborescences, Sum-of-disjoint products, Time aggregated graph.

Acknowledgements

This Ph.D. thesis would not have been possible without generous support from many people. I take this opportunity to extend my earnest gratitude and appreciation to all those who directly or indirectly gave impetus to me to be perseverant over the past five years.

Foremost, I am indebted to my advisors Prof. Sanjay K. Chaturvedi and Dr. Sieteng Soh for their continuous support, without which this thesis would not have been possible. Their endurance, enthusiasm, motivational sermons, and immense knowledge not only inspired me to push myself for better, but also helped me to overcome all barriers during research and writing of this thesis. For all this, I shall remain grateful to both of them for my entire life. Further, I admit that I am privileged to get an opportunity to complete my Ph. D. under their able supervision.

In addition, I would like to thank the rest of my Doctoral Scrutiny Committee (DSC) members: A/Prof. N. K. Goyal, Dr. M. Sarma and Prof. J. Maiti for their encouragement, insightful comments, and tough questions. I would also like to thank Prof. V. N. A. Naikan and Dr. Rajiv N. Rai for their valuable advices and encouragement throughout my stay at Indian Institute of Technology Kharagpur.

My sincere thanks also goes to Dr. Rajesh Mishra, Prof. Dilip K. Yadav and Dr. Amit Prakash who have always stood beside me through thick and thin. Thank you for patiently listening to my problems and providing me valuable insights to tackle them. I would also like to thank Prof. Kwan-Wu Chin for sacrificing his personal time to review one of my papers, and Prof. Pierluigi Crescenzi for several stimulating discussions through e-mail.

I thank all my fellow lab mates, both seniors and juniors, at Subir Chowdhury School of Quality and Reliability, Indian Institute of Technology Kharagpur for the encouraging discussions, moral support and all the fun which we have had during this course. Also, it is worth to mention the names of some wonderful people whom I met at Curtin University and developed a great bond of friendship: Ms. Lely, Mr. Tony, Ms. Dejalin, Mr. Shashank, Mr. Subhojit and Ms. Rebecca. I will always cherish the warmth shown by them.

A special mention of thanks to my friends Mr. Rahul, Mr. Himanshu, Mr. Akbar, Ms. Madhvi and Ms. Rehnuma who provided distraction and constant support during my breakdown moments. Pals, I am lucky to have you in my life.

My sincere appreciation goes to the administration of Indian Institute of Technology Kharagpur for providing me financial assistance during this Ph. D. Also, thanks to the Curtin University for providing me Curtin International Postgraduate Research Scholarship (CIPRS) to support my stay in Perth for one year.

At last, I would also like to convey a heartfelt thank you to my mother, father and sister for always believing in me, shielding me from any harm and encouraging me to follow my passion and dreams.

Gaurav Khanna

Table of Contents

List of Figures

List of Tables

This Ph. D. dissertation embodies an original research that has culminated either into a published research paper or a book chapter in an edited volume.

Published Works

A. Papers in Refereed Journals-(06)

1. **Khanna G.**, Chaturvedi S. K., Soh S., Chin K.-W. On Enumeration of Spanning Arborescences and Reliability for Network Broadcast in Fixed-Schedule Dynamic Networks. IEEE Trans. on Network Science and Engineering. 2020 Oct-Dec 1: 7(4): 2980-2996.

2. **Khanna G.**, Chaturvedi S. K., Soh S. Two-Terminal Reliability Analysis for Time-Evolving and Predictable Delay-Tolerant Networks. Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering). 2020 Mar 1;13(2):236-50.

3. **Khanna G.**, Chaturvedi S. K., Soh S. On computing the reliability of opportunistic multi-hop networks with Mobile relays. Quality and Reliability Engineering International. 2019 Jun;35(4):870-88.

4. **Khanna G.**, Chaturvedi S. K., Soh S. Reliability evaluation of mobile ad hoc networks by considering link expiration time and border time. International Journal of System Assurance Engineering and Management. 2019 Jun 1;10(3):399-415.

5. **Khanna G.**, Chaturvedi S. K. A comprehensive survey on multi-hop wireless networks: milestones, changing trends and concomitant challenges. Wireless Personal Communications. 2018 Jul 1;101(2):677-722.

6. Chaturvedi S. K., **Khanna G.**, Soh S. Reliability evaluation of time evolving Delay Tolerant Networks based on Sum-of-Disjoint products. Reliability Engineering & System Safety. 2018 Mar 1;171:136-51.

B. Book Chapters-(02)

1. **Khanna G.**, Chaturvedi S. K., Soh S. Time Varying Communication Networks: Modelling, Reliability Evaluation and Optimization. In Advances in Reliability Analysis and its Applications, M. Ram and H. Pham, Eds., 2020 (pp. 1-30). Cham: Springer International Publishing.

2. Chaturvedi S. K., Soh S., **Khanna G**. On Modelling and Performability Evaluation of Time Varying Communication Networks. In Handbook of Advanced Performability Engineering, K. B. Misra, Eds., 2021 (pp. 161-189). Cham: Springer.

Acronyms

Abbreviation	Meaning
npTVCN	Non-predictable TVCN
pTVCN	Predictable TVCN
t-arborescence	Timestamped Spanning Arborescence
tv-arborescence	Timestamped Valid Spanning Arborescence
BDD	Binary Decision Diagrams
BT	Border Time
FS	Free Space Propagation Model
FS-TRG	Free Space-Two Ray Ground Propagation Model
DSN	Deep Space Network
DTNs	Delay Tolerant Networks
ENOF	Expected Number of Node Failure
EHC	Expected Hop Count
ESC	Expected Slot Count
FANETs	Flying Ad hoc Networks
FODAVA	Foundations of Data and Visual Analytics
GPS	Global Positioning System
ICT	Information and Communication Technology
IPN	Interplanetary Networks
LET	Link Expiry Time
LEO	Low Earth Orbiting
MANETs	Mobile Ad hoc Networks
MCS	Minimal Cut Set
MCA	Minimum Cost Arborescence
MCT	Minimum Cost Tree
MPS	Minimal Path Set
MVI	Multi Variable Inversion
OBDD	Ordered BDD
RGG	Random Geometric Graph
SDP	Sum-of-Disjoint Products
SNAP	Stanford Network Analysis Platform
SVI	Single Variable Inversion
TAGs	Time Aggregated Graphs
TRG	Two-Ray Ground Propagation Model
TSE	Timestamped Edge
TS-MPS	Timestamped Valid Minimal Path Set
TS-MCS	Timestamped Minimal Cut Set
TVCNs	Time Varying Communication Networks
UAVs	Unmanned Aerial Vehicles
VANETs	Vehicular Ad hoc Networks
WINNER	Wireless World Initiative New Radio

Notations

Notation	Definition
τ	Maximum value of timestamp/slot, *i.e.*, TVCN period
(η, ξ)	Weibull distribution parameters
$e_{i,j}^{t}$	A TSE from node v_i to v_j at timestamp $t \in [1, \tau]$
$p_{e_{i,j}^{t}}$	Probability of success of the TSE $e_{i,j}^{t}$
(ψ, φ)	A latitude-longitude pair
(s, d)	Source-Destination node pair
r	Transmission range of a mobile node
$K_{\lvert V \rvert}$	Complete network graph of $\lvert V \rvert$ nodes
$G(V, E)$	A loopless directed graph G with set of nodes V and set of bidirectional edges E
$e_{i,j}$	A directed edge in E from node v_i to v_j, where $v_i, v_j \in V$
TS	A $\lvert V \rvert \times \lvert V \rvert$ matrix containing sequence of *timestamps* of each edge $e_{i,j} \in E$
$TS(i, j)$	A sequence (in an ascending order) of timestamps in TS of edge $e_{i,j}$
$TS(i, j, k)$	The k^{th} timestamp between node pair (v_i, v_j)
$G'(V, E^{TS})$	A TAG for graph $G(V, E)$ with set of edges E^{TS}, each of which is associated with a sequence of timestamps in TS
$e_{i,j}^{TS(i,j)}$	An edge in E^{TS} between nodes $v_i, v_j \in V$
$G''(V, E'')$	A directed multigraph of a TAG G' with set of edges E'', each of which is associated with only *one* timestamp, *viz.*, TSE
A_j	Set of all arborescences originating from root node $v_j \in V$ in TAG G'
$\lvert \mathrm{A}_j \rvert$	Number of arborescences in A_j
A_j^i	i^{th} arborescence in set A_j
A_{All}	Set of $\mathrm{A}_j \ \forall \ v_j \in V$
$\lvert \mathrm{A}_{All} \rvert$	Number of arborescences in A_{All}
TA_j	Set of all t-arborescences originating from root node $v_j \in V$ in TAG G'
$\lvert \mathrm{TA}_j \rvert$	Number of t-arborescences in TA_j
TA_j^i	i^{th} t-arborescence in set TA_j
TA_{All}	Set of $\mathrm{TA}_j \ \forall \ v_j \in V$
$\lvert \mathrm{TA}_{All} \rvert$	Number of t-arborescences in TA_{All}
TVA_j	Set of all tv-arborescences originating from root node $v_j \in V$ in TAG G'
$\lvert \mathrm{TVA}_j \rvert$	Number of tv-arborescences in TVA_j
TVA_j^i	i^{th} tv-arborescence in set TVA_j
TVA_{All}	Set of $\mathrm{TVA}_j \ \forall \ v_j \in V$
$\lvert \mathrm{TVA}_{All} \rvert$	Number of tv-arborescences in TVA_{All}
TVA_j^{α}	Earliest tv-arborescence generated from arborescence A_j^i
TVA_j^{β}	Latest tv-arborescence generated from arborescence A_j^i

$\mathrm{TSE}(a,b)$	Set of TSEs corresponding to edge $e_{a,b}$
$L^+(G)$	Laplace in-degree matrix of a graph G
$\hat{L}_j^+$	Reduced $L^+(G)$ obtained by deleting j^{th} row and column
L_i	A set of leaf nodes in TSE_List
N_j^+	Number of spanning arborescences originating from node $v_j \in V$
$\mathrm{TS\text{-}MPS}_{s,d}$	A set of all TS-MPS between (s,d) node pair
$\mathrm{TS\text{-}MPS}_{s,d}^i$	A TS-MPS in $\mathrm{TS\text{-}MPS}_{s,d}$ generated from t-arborescence TA_s^i
$R(s,d)$	2-terminal reliability of device-to-device connected TVCN
$R(v_j)$	Reliability of broadcast from root node $v_j \in V$ of device-to-device connected TVCN, *e.g.*, DTN
$R_1(K)$	Reliability of broadcast from *all* nodes in set $K \subseteq V$ of device-to-device connected TVCN
$R_2(K)$	Reliability of broadcast from *any* node in set $K \subseteq V$ of device-to-device connected TVCN
RQ_j	Set of all *required* TSEs with respect to node $v_j \in V$
$U_j(\mathrm{TSE})$	Percentage utilization of TSEs with respect to root node $v_j \in V$
$U_{All}(\mathrm{TSE})$	Total percentage utilization of TSEs
$U_j(\mathrm{SA})$	Percentage utilization of spanning arborescences with respect to node v_j
$U_{All}(\mathrm{SA})$	Total percentage utilization of spanning arborescences
$\mathbb{N}$	Number of timestamps available for TVCN upgrade
TVA_{min}	Set having the minimum number of tv-arborescences
v_{min}	Root node corresponding to TVA_{min}
TA_{min}	Set of t-arborescences corresponding to v_{min}
$\mathcal{R}$	Radius of Earth
T	Total mission time of a MANET
$\mathrm{LET}_{\mathrm{min}}$	Minimum LET among all LETs
$\mathrm{BT}_{\mathrm{min}}$	Minimum BT among all BTs
$G(N, L)$	A RGG representing a MANET of N nodes and L links
R_{2TR_m}	2-terminal reliability of end-to-end connected TVCN, *i.e.*, a MANET

Chapter 1 Introduction

In the past few years, networks with dynamic connectivity have observed extensive and intensive research interests from engineering and computer research fraternity. The widespread research interest in the domain led to the development of fixed infrastructure less wireless networks like Mobile Ad Hoc Networks (MANETs), Vehicular Ad Hoc Networks (VANETs), Flying Ad Hoc Networks (FANETs) and Delay Tolerant Networks (DTNs) (Soh et al. 2008; Liang and Modiano 2017). Since their topologies change as a function of time, we henceforth refer to all such networks with a common term – Time Varying Communication Networks (TVCNs) in this entire thesis. A topology change in a TVCN can be attributed to a variety of *intrinsic* (predictable and inherent) interruptions such as node mobility and/or *extrinsic* (unpredictable) interruptions such as shadowing that occurs in wireless channel, and hardware failures (Liang and Modiano 2017). Note that topology changes in these networks are not considered as an anomaly, rather regarded as an intrinsic feature (Casteigts et al. 2012). Further, in general, TVCNs exhibit extremely long delays and show intermittent connectivity as nature of the system. More specifically, in practice most TVCNs seldom have *end-to-end* connected multi-hop paths between any node pair and utilize *device-to-device* connection via *store-carry-and-forward* mechanism for data transmission (Casteigts et al. 2011). Thus, TVCNs may actually be disconnected at every time instant; however, data transmission can be made possible via routes available over time and space. For example, a MANET is a TVCN which depends on *end-to-end* connectivity for data communication. On the contrary a DTN is a TVCN which uses *device-to-device* communication.

Primarily TVCNs were designed for providing internet services in the remote areas of developing countries (Pentland et al. 2004), wildlife monitoring (Juang et al. 2002) and battlefield reconnaissance. However, the realization of potential and opportunities of these networks led to more advanced networks like Low Earth Orbiting (LEO) satellite networks, Deep Space Networks (DSN), Interplanetary Networks (IPN) and the networks of

Unmanned Aerial Vehicles (UAVs), *i.e.*, FANETs (Bekmezci et al. 2013; Zhang et al. 2019). Among these applications, movement of animals acting as *data ferry* during wildlife monitoring presents an interesting instance of non-predictable TVCN (*np*TVCN). On the other hand, an LEO satellite network and fleet of buses with predestined trajectories and schedules have somewhat predictable network dynamics over time (Xuan et al. 2003a). Such TVCNs are referred to as *predictable* TVCNs (*p*TVCNs) (a.k.a. Fixed Schedule Dynamic Networks (FSDNs)). Thus, specifically, a TVCN refers to an umbrella term depicting both *np*TVCN and *p*TVCN. Note that this thesis primarily deals with *p*TVCNs. Thus, for the sake of uniformity throughout the text, we use TVCN to signify a *p*TVCN.

It is important to note here that although a plethora of works exist in the literature which deal with topology control (Huang et al. 2013), routing (Raffelsberger and Hellwagner 2014) and trace collection (Baudic et al. 2016) in TVCNs; however, assessment of their reliability is still rarely explored and needs considerable attention. One main reason for the deficient *state-of-the-art* works for the reliability evaluation of TVCNs is due to dynamically changing topology. Further, due to the dynamic topology changes, the definitions and notions of conventional Minimal Path/Cut Sets and Spanning Trees/Arborescences developed for evaluating static networks' reliability become mostly inapplicable in their usual form in TVCNs. Thus, there is a need of substantial modification and extension of these notions to TVCNs as well.

This thesis extends conventional notions of Minimal Path Set, Minimal Cut Set and Spanning Arborescences of static networks to *device-to-device* connected TVCNs by considering *store-carry-and-forward* mechanism for data transmission. More specifically, the research work contained in this thesis extends the conventional terms in network reliability domain to TVCN domain with new terms, *viz.*, Timestamped Valid Minimal Path Set, Timestamped Minimal Cut Set and Timestamped Valid Spanning Arborescences, and definitions, respectively. The thesis also presents novel algorithms to enumerate them. All enumerated Timestamped Valid Minimal Path Set, Timestamped Minimal Cut Set and Timestamped Valid Spanning Arborescences are later utilized to obtain various reliability metrics of a TVCN. In addition, this thesis presents initial insights on the topic of TVCN

design, management and upgrade. The thesis also evaluates the reliability of *end-to-end* connected MANETs via Monte Carlo simulation by using the concept of Link Expiration Time and Border Time.

1.1. Aims and Approaches

The objectives of this thesis are as follows:

Aim 1: To extend the concept of Minimal Path Set and Minimal Cut Set or similar term, commonly used to evaluate the 2-terminal reliability of static networks, to TVCNs, and devise algorithm(s) to enumerate them. The literature survey does not indicate that such concepts, *viz.*, *all* Timestamped Valid Minimal Path Sets and Timestamped Minimal Cut Sets, have earlier been attempted or addressed by the researchers for TVCNs.

In Chapter 3, we propose three different techniques, *viz.*, Cartesian Product based, Connection Matrix based and Line Graph based, to enumerate *all* Timestamped Valid Minimal Path Sets. Besides, we propose two techniques, *viz.*, Timestamped Valid Minimal Path Set inversion based and Line Graph based, to enumerate *all* Timestamped Minimal Cut Sets.

Aim 2: To extend the concept of Spanning Arborescence or similar term, commonly used to evaluate the global reliability of static networks, to TVCNs, and devise algorithm(s) to enumerate them. This concept and its enumeration, *i.e.*, enumerating *all* Timestamped Valid Spanning Arborescences for TVCNs, has also not been addressed in the literature.

In Chapter 4, we adapt Tutte's Matrix Tree Theorem (Aigner 2007) and propose two methods, *viz.*, Method 1 and Method 2, which enumerate *all* Timestamped Valid Spanning Arborescences of a TVCN. In addition to the three algorithms presented in Chapter 3, the Chapter 4 also presents a new technique for enumerating all Timestamped Valid Minimal Path Sets using *all* Timestamped Spanning Arborescences. Note that all Timestamped Spanning Arborescences are intermediate results generated by Method 1 while enumerating all Timestamped Valid Spanning Arborescences.

Aim 3: To develop the methodology to evaluate some reliability related metrics, *e.g.*, 2-terminal reliability, Expected Hop Count, Expected Slot Count and Broadcast reliability, for device-to-device connected TVCNs. Existing literature pertaining to TVCNs' lack reliability metrics.

In Chapter 5, we use the well-known Sum-of-Disjoint Products (SDP) (Rai et al. 1995) technique to evaluate different reliability measures.

Aim 4: To develop technique(s) for TVCN topology control, contact plan modification, and topology upgrade. TVCN design, management and upgrade have not been addressed in the literature.

In Chapter 6, we first present a simple and novel technique for TVCN topology control and contact plan modification. Then, we propose a greedy algorithm for upgrading a TVCN.

Aim 5: To compute the reliability of end-to-end connected TVCNs, *e.g.*, MANETs, by considering Link Expiration Time and Border Time, whereby the changes in topology are predicted accurately. It is worth mentioning here that the existing techniques do not precisely estimate the time instants where the topology of MANET changes; hence, result in imprecise reliability estimation.

In Chapter 7, we consider conditional survival of the mobile nodes executing a mission and utilize the concept of Link Expiration Time and Border Time along with Monte Carlo Simulation to precisely estimate the 2-terminal reliability of MANETs.

1.2. Significance and Contributions

This thesis addresses five intertwined objectives related to reliability modelling, evaluation and optimized design of TVCNs. More specifically, the significance and contributions of this work are as follows:

1) The current state-of-the-art techniques available in the literature for assessing the

performance of TVCNs mainly focus on developing new and efficient routing algorithms. These techniques often try to optimize the cost of a routing path, but, seldom explicitly consider temporal domain into their topology model for studying the topology dynamics at different time intervals and its impact on two-terminal connectivity. Note that a good two-terminal connectivity shows presence of resilient *unicast communication* path, *i.e.*, connection between two specified nodes. Besides, many researchers, working in the domain of TVCNs, consider the terms Minimal Path Sets and walks to be same and use these interchangeably as *journeys* (Casteigts et al. 2012; Santoro 2016). Note that the concept of *walks* is seldom used in the domain of reliability engineering in comparison to Minimal Path Sets. A Minimal Path Set may be a journey but *vice versa* is not true. Thus, to the best of our knowledge, Aim 1 of this thesis is the first to enumerate *all* possible *Minimal Path Sets over time*, which we call as *Timestamped Valid Minimal Path Sets*, of TVCNs by considering temporal information; see Chapter 3. To the best of our knowledge, the chapter also presents the first attempt to enumerate *all Minimal Cut Sets over time* or *Timestamped Minimal Cut Sets* for TVCNs. Its solution is expected to verify whether there exists a duality between Timestamped Valid Minimal Path Sets and Timestamped Minimal Cut Sets for TVCNs as it exists in static networks. Another prospective of this problem is to verify in context of TVCNs the perception that in most practical systems the number of Minimal Cut Sets is much smaller than the number of Minimal Path Sets, because of which it is advisable to address the Capacity Related Reliability evaluation problems with the help of Minimal Cut Sets (Chaturvedi and Mishra 2009). Further, the Timestamped Minimal Cut Sets can also be used to find the weak links in a design or single point failure and reliability bounds in TVCNs.

2) Apart from unicast, the two other important communication models of TVCNs are: network *broadcasting* (Bhadra and Ferreira 2003) and *convergecasting* (Chen et al. 2016). Examples of broadcasting include maneuvering, tracking, software updates and maintenance of satellites from an Earth based ground station (Maini and Agrawal 2014). In contrast, the reception of data gathered by the LEO satellites at a ground station is an instance of convergecasting (Fraire et al. 2017a). In *static* or conventional

networks, the aforementioned communication models are usually accomplished by constructing *spanning arborescences* and *spanning trees* for *directed* and *undirected* networks, respectively; for brevity, here onwards we refer to a spanning arborescence/tree as an/a *arborescence/tree*. However, unlike static networks, each edge of a TVCN has a time window or slots in which it is active. Thus, existing solutions used to generate arborescences cannot be applied in a TVCN as they assume a fixed topology. Hence, arborescences of TVCNs must consider edge schedules. The Aim 2 of this thesis utilizes a simple yet powerful model *viz.*, Evolving Graph often referred to as Time Aggregated Graph (TAG) model to represent TVCNs by including the effect of *time dimension* on the network. The problem is formally defined in Chapter 4, wherein we propose *two* novel methods to generate *all* Timestamped Valid Arborescences of a TVCN. Both methods extend Tutte's Matrix-Tree theorem (Aigner 2007), a classical theorem of graph theory. The proposed methods can be very helpful in identifying all possible broadcast opportunities within a given observation period in a variety of TVCNs like DTNs and VANETs. Note that we only focus on broadcasting as convergecasting is a complementary problem, and our solution can be used for convergecasting with a slight modification, *i.e.*, by reversing the direction of data flow.

3) Aim 3 extends the work done in Aim 1 and Aim 2; see Chapter 5. It suitably conditions the enumerated Timestamped Valid Minimal Path Sets and Timestamped Valid Arborescences to compute the exact 2-terminal and broadcast reliability of device-to-device connected TVCNs with the help of well-known SDP algorithm. It also enumerates the *foremost*, *shortest*, and *fastest* Timestamped Valid Minimal Path Sets from the list of all Timestamped Valid Minimal Path Sets to find the *earliest arrival date*, the *minimum number of hops* and the *minimum time spent* on a path set, respectively and computes the 2-terminal reliability associated with them. Further, this aim introduces and computes a new network performance metric, *i.e.,* Expected Slot Count (to find the expected delay in the TVCNs) in addition to the evaluation of an already existing and well-known metric, *viz.*, Expected Hop Count. The proposed metrics can be very helpful in analyzing the probability of success of data transmission among different pair of mobile nodes, or from a root node or a set of root nodes to all

others within a given observation period of a TVCN. The metrics can also assist a network manager to optimize an effective routing under different objective functions.

4) The *topology control* and design of *contact plans* for TVCNs have received little attention, as it is either assumed that contacts are scarce or that all potential contacts between TVCN nodes can belong to the contact plan (Fraire et al. 2014). Besides, to the best of our knowledge, there is yet no work on TVCN upgrade. In Aim 4, *i.e.*, in Chapter 6, we optimize the contact plan and present a greedy algorithm for addressing TVCN upgrade problem; thereby help in efficient topology design of TVCNs.

5) The earlier works present in the literature (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013) compute the 2-terminal reliability of the end-to-end connected TVCNs, *viz.*, MANETs by observing the changes in the topology after a fixed incremental time. However, under such scenario all the changes occurring at any intermediate instant of time within that interval would be completely missed. Under Aim 5, in Chapter 7, we tackle aforementioned challenge by providing a solution for the evaluation of 2-terminal reliability of MANETs by determining more precisely the incremental time at which topology change(s) would occur.

Figure 1.1 summarizes the contributions of this thesis in augmenting the research in the area of TVCNs.

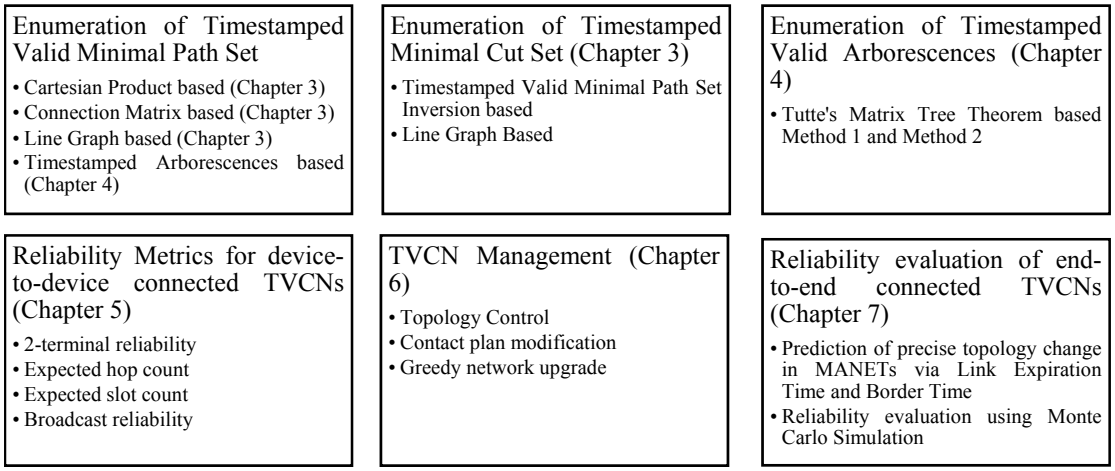


Figure 1.1. Summary of the contributions.

1.3. Structure of the Thesis

This thesis is organized as follows.

Chapter 2 provides a preliminary theoretical background on TVCNs, definitions and notations that are used in the thesis. A review of related works on TVCN modelling is also presented in this chapter. The chapter also discusses available performance evaluation tools for TVCNs, and highlights main differences between static networks and TVCNs. Further, the chapter also briefly describes existing reliability evaluation techniques.

Chapter 3 considers device-to-device connected TVCNs and formally defines the problem of enumerating all Timestamped Valid Minimal Path Sets and Cut Sets between a given source v_s and destination v_d, *i.e.*, (s, d) node pair, of a TVCN. It presents three distinct approaches for enumerating all Timestamped Valid Minimal Path Sets and two for enumerating all Timestamped Minimal Cut Sets. More specifically, it first presents a *Cartesian Product* based approach for the enumeration of all Timestamped Valid Minimal Path Sets between a given (s, d) node pair; however, later it is discovered that this method needs to be run multiple times to compute all Timestamped Valid Minimal Path Sets between every pair of nodes. As a ramification, *Connection Matrix* based method was introduced which enumerates all Timestamped Valid Minimal Path Sets between every (s, d) pair of nodes at once. The last approach first transforms a TAG into a Line Graph, by preserving its temporal information, and then generates its Minimal Path Sets via existing algorithms for static networks. On the contrary, as the first technique for Timestamped Minimal Cut Sets enumeration, we invert all Timestamped Valid Minimal Path Sets using well-known De-Morgan's laws of Boolean algebra. Another approach uses Line Graph and then generates its Minimal Cut Sets via existing algorithms for static networks. The matching simulation results from each technique validate the correctness of each approach.

Chapter 4 considers device-to-device connected TVCNs and formally defines the problem of enumerating all Timestamped Valid Arborescences of a TVCN. Two approaches, namely Method 1 and Method 2, which adapt Tutte's Matrix Tree Theorem for the purpose of enumeration are described. Besides, the chapter presents some propositions

and proofs related to both Timestamped Arborescences and Timestamped Valid Arborescences. As an application of the presented concepts, it generates all Timestamped Valid Minimal Path Sets using the set of all of Timestamped Arborescences. It also provides a set of simulation results to show that Method 2 performs better in comparison to Method 1 in terms of CPU execution time.

Chapter 5 utilizes the results collated in Chapter 3 and Chapter 4 along with the well-known SDP technique to obtain network unicast and broadcast related reliability metrics for TVCNs. More specifically, it computes the 2-terminal reliability, *viz*., $R(s,d)$ associated with a given (s,d) node pair of device-to-device connected TVCNs. Note that $R(s,d)$ depicts a probability that data packets transmitted from source node v_s will be successfully received by the destination node v_d. The chapter also evaluates the 2-terminal reliability associated with the *shortest, fastest* and *foremost* Timestamped Valid Minimal Path Sets segregated from the list of all Timestamped Valid Minimal Path Sets. Besides, the chapter introduces and computes a new network performance metric, *i.e.*, *Expected Slot Count* (to find the expected delay) in addition to an already existing and well-known metric, *viz.*, *Expected Hop Count*. In addition, the chapter introduces and evaluates three different broadcast reliability metrics, *viz.*, $R(v_j)$, $R_1(K)$ and $R_2(K)$. Note that $R(v_j)$ of device-to-device connected TVCNs measures the probability that data packets broadcasted by a root node v_j are successfully received by all other nodes of the network, $R_1(K)$ gives the probability that the data packets broadcasted by all root nodes in a set of nodes K are successfully received by all other nodes of the network and $R_2(K)$ gives the probability that the data packets broadcasted by any node in a set of nodes K are successfully received by all other nodes of the network.

Chapter 6 considers problem related to i) topology control, ii) contact plan modification and iii) TVCN upgrade. The three problems are formally defined in this chapter. Initial insights are provided for the first two problems while a greedy algorithm has been proposed for the third problem. More specifically, we consider broadcast model and present four new metrics namely, U_j(TSE), U_{All}(TSE), U_j(SA) and U_{All}(SA) to assess the utilization of timestamped edges (TSEs) (defined later in Chapter 2) and spanning arborescences. It is

believed that the proposed metrics will assist a network designer to identify the best root location, generate better contact plans and/or select best contact plan among alternatives, thereby will produce an efficient TVCN design. Further, the performance of the proposed greedy algorithm is analyzed with the help of ten arbitrarily generated TAGs of different sizes. The obtained simulation results are compared against random TVCN upgrade. The results clearly show that the proposed greedy algorithm performs better than the average case of random TVCN upgrade.

Chapter 7 considers end-to-end connected TVCNs, *i.e.,* MANETs. It proposes the notion of Border Time, and utilizes Link Expiry Time model and conditional survival of the mobile nodes along with Monte Carlo Simulation for estimating the 2-terminal reliability of MANETs, *i.e.*, R_{2TR_m}. The presented approach is more accurate as the earlier approaches in (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013) overlooked the conditional survival of the mobile nodes executing a mission. Besides, they utilized fixed incremental time to capture the topology changes and hence provided erroneous results and underestimated the value of reliability. Simulation results from the proposed technique are compared against the existing approach in (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013). The results prove the effectiveness of the proposal.

Finally, Chapter 8 concludes this research documentation and provides some presumptive future directions for furthering this research.

Chapter 2 Background

This chapter reviews pertinent and existing literature from the TVCN domain. The chapter is organized as follows.

Section 2.1 provides an overview of TVCNs. Section 2.2 discusses various modelling paradigms developed for representing various features of TVCNs. More specifically, Section 2.2.1 discusses various mobility models from the literature. Section 2.2.2 and Section 2.2.3 describe the connectivity and data communication models, respectively, utilized in TVCNs. Section 2.2.4 is devoted to various radio propagation based link models. Note that, without loss of generality, this thesis uses link and edge, and node and vertex interchangeably. Section 2.2.5 discusses some topological models. It also defines Minimal Path Set (MPS), Minimal Cut Set (MCS) and Arborescence, equivalent models for TVCNs and provides motivation behind the development of these notions. Section 2.3 reviews the literature on methods of topology control and contact plan design. Section 2.4 presents some simulators, emulators and real world experiments developed to analyze the performance of TVCNs. Section 2.5 presents some prominent differences between TVCNs and static networks. In addition, Section 2.6 presents an overview of network reliability of both static networks and TVCNs. At last, Section 2.7 summarizes the chapter.

2.1 TVCN: An Overview

TVCNs are becoming very popular today as they can complement the pre-existing infrastructure based wireless communication networks in sharing their data load. Further, new vistas are evolving each day for their application in disruption tolerant or opportunistic environments. As these networks are, in general, deployed in extreme environmental conditions and/or in inaccessible areas their performance evaluation becomes a matter of great concern. Several models, simulators, testbeds and visualization tools have evolved in the last two decades for analyzing their characteristics.

In upcoming sections, we discuss a number of models, simulators, emulators, testbeds and real world projects implementing such networks. We also discuss an important aspect of these networks, *i.e.*, reliability, as with the growing dependency of many applications on wireless modes, there has been an increasing demand for more reliable wireless networks.

2.2 Modelling Techniques

Formally, a TVCN could be modelled as a network graph wherein its each node and/or edge has a presence schedule defined for it. The edge (node) existence schedule indicates the time instances at which an edge (node) is present, and possibly other parameters such as traversal distance, traversal cost and latency (Ferreira 2003). With the help of the edge and node existence schedules, changes in topology of a TVCN can be studied (Ferreira 2002). A number of models have been proposed in the literature to study and predict the performance characteristic(s) of TVCNs. This section presents some existing models for representing various features of TVCNs. A comprehensive survey on them has been published by the scholar in (Khanna and Chaturvedi 2018). Here, we summarize some significant observations from it.

2.2.1 Mobility Models

In TVCNs, one of the prime network attribute is *node mobility*. A *mobility model* attempts to capture the motion of mobile nodes with the change in their speeds and directions over time. It is well-known that the real-life mobility patterns of human beings and vehicles can be very complex to model (Roy 2011). The complex mobility pattern entails more parameters to be included into the mobility model, thereby, making the model very intricate. Performance assessment of TVCNs in the presence of node mobility, characterized by some mobility model, generally requires an investigation of the impact of change in *node velocity* and *mobility patterns* on network reliability, protocols and application services. Such an analysis is of prime importance for better design and implementation of TVCNs. In a facile classification, mobility models can be categorized as: i) *Synthetic mobility models* (Bai and Helmy 2004), and ii) *Trace based mobility models* (Aschenbruck et al. 2011; Munjal et al. 2012). A *synthetic mobility model* depicts randomly generated movements and creates

synthetic traces while a *trace-based mobility model* is developed by monitoring and extracting features from the real movement patterns of users carrying mobile nodes, thus epitomizes reality. Generally, a synthetic mobility model requires complex mathematical modelling, but it can be easily applied to an arbitrary number of nodes and over a large scale network. Although several synthetic mobility models have been proposed in the literature and many traces have been collected from the real-world experiments to precisely model node movement, yet there is no universal model, which caters all the requirements. Moreover, in the past, mostly the data traces have been collected by deploying the mobile devices in a small region, *viz.*, usually a university campus or a conference room, and require large time overhead (say six months to one year) to collect a good amount of traces to avoid any biased data from appearing in the data set. It is worth mentioning here that many data traces are freely available on repositories like CRAWDAD, Sociopatterns, UMass Trace, Foundations of Data and Visual Analytics (FODAVA), Stanford Network Analysis Platform (SNAP), UC Irvine Machine Learning Repository and MobiLib for further research. Curious readers can refer the recent reviews on mobility models and data-trace repositories in (Batabyal and Bhaumik 2015; Baudic et al. 2016) for more details. Besides, it can be concluded from the information in the cited literature that - at present, trace collection is a difficult job as deploying mobile phones on a large scale for data collection is impractical and costly. It is evident that both techniques have their own pros/cons and challenges. For the readers' benefit, a list of the pros and cons, and associated research challenges for both categories have been tabulated in Table 2.1 and Table 2.2, respectively.

Table 2.1. Pros and Cons of trace based and synthetic mobility models.

Type	**Pros and Cons**					
	Represents Reality	**Deployment Cost**	**Computational Overhead**	**Scalable**	**Complexity**	**Availability of Models**
Trace based models	Yes	High	Low	No	Low	Few
Synthetic models	No	Low	High	Yes	High	Large

Table 2.2. Research challenges in trace based and synthetic mobility models.

Index	Synthetic models	Trace based models
1	Difficult to design a widely acceptable, realistic and simple mobility model	Huge variation in collected trace data with respect to day and time
2	Hard to decide a suitable model for a given scenario	Missing Data
3	Difficulty in validation via traces	Requires filtering and pre-processing
4	Speed decay problem	Sufficient samples need to be acquired
5	Extreme variation in the values of the parameters of simulation	

2.2.2 Connectivity Models

The primitive versions of routing protocols designed for MANETs assumed existence of an *end-to-end* connectivity between (s, d) pair of nodes. However, in reality, such protocols fail to deliver data if an *end-to-end* connectivity/path is not found. In other words, real scenarios encounter situations wherein network gets partitioned and the traditional ad-hoc routing protocols fail to interconnect different partitions (Raffelsberger and Hellwagner 2014). Moreover, with the increasing hop count between v_s and v_d, the end-to-end connected path tends to become unstable due to frequent disconnection of path caused by the movements of intermediate nodes (Matis et al. 2016). Thus, the lack of continuous end-to-end connectivity between devices gave an impetus to the development of *device-to-device* connectivity for opportunistically routing data from one device to another via *store-carry-and-forward* mechanism. More specifically, the DTN group under Internet Research Task Force addressed the issues of intermittent connectivity and partitioned networks via their proposed store-carry-and-forward paradigm. In addition, the group also resolved the needs of the overlay architecture by using an addressing scheme that exploits the late binding of addresses (Zhang 2006). It is worth to note here that in store-carry-and-forward paradigm of data routing a next hop may not be immediately available to the current mobile device for forwarding data packets. Thus, it necessitates the current device to store data packets, maybe for a considerable duration, until it gets an opportunity to forward the

packets to some other device. This renders transmissions between intermediate devices to be independent of each other. Therefore, the notion of store-carry-and-forward, accomplished in *device-to-device* connection, mitigates the effect of hop count to a large extent (Matis et al. 2016).

This thesis considers both device-to-device and end-to-end connectivity, respectively, for TVCN performance evaluation.

2.2.3 Communication Models

The three primarily used communication models in TVCNs are *unicasting*, *broadcasting* and *convergecasting*. Network unicasting deals with data exchange between a single source v_s and destination v_d. Usually in wireless communication, such a connection is established in a single hop manner between (s, d) node pair; however, in multi hop unicast communication, except for (s, d) node pair, all intermediate nodes may be considered as relay nodes. In contrast, network broadcast deals with data or information dissemination from a *root/source* node to all other nodes. Note that if the broadcast involves data dissemination from a root node to only some nodes instead of all, then it is often termed as *multicast*. Some instances which require broadcasting in TVCNs are manoeuvring, tracking, software update and maintenance of satellites from the Earth based ground station (Maini and Agrawal 2014). On the other hand, convergecasting model is for collecting data from some (all) nodes towards a *sink/destination* node (Du and Hu 2008). A convergecasting example is the reception of data gathered by the LEO satellites from different sites across the world, including remote places which are inaccessible for ground-based data acquisition centre, at ground station (Fraire et al. 2017a). It is worth mentioning here that all aforementioned models are equally important. However, between broadcasting and convergecasting most of the works in the literature focus on either of them, mainly because they both represent complementary problems. These models entail generation of efficient path(s) (for unicast) and arborescence(s) (for broadcast/convergecast) which consume minimum energy/cost, ensure collision-free communication and/or provide maximum reliability.

The recent works (Casteigts et al. 2012; Santoro 2016) modelling multi hop unicast communication in TVCNs, consider the terms paths over time, *i.e.*, Timestamped Valid Minimal Path Sets, as equivalent of 'walks' in static networks and use these interchangeably as journeys. However, note that the concept of walks is seldom used in the domain of reliability engineering in comparison to an MPS. Further, the existing works do not aim to find all Timestamped Valid Minimal Path Sets. In contrast, this thesis presents four techniques (three in Chapter 3 and one in Chapter 4) to enumerate all Timestamped Valid Minimal Path Sets of TVCNs. On the other hand, works considering broadcast (Bhadra and Ferreira 2002, 2003) and convergecast (Chen et al. 2016) in TVCNs only aim to find a single Timestamped Valid Arborescence satisfying the constraints on energy/cost and/or time. However, Chapter 4 of this thesis presents in detail two novel approaches to enumerate all Timestamped Valid Arborescences for broadcasting in TVCNs.

2.2.4 Radio Propagation based Link Models

In reality, even if the mobile nodes are in transmission range of each other, the signal strength deteriorates due to a variety of reasons like noise, fading and/or interference etc., thereby, affects the probability of a successful communication. The propagation based link reliability model considers the link reliability as a combination of *Free Space propagation model* (FS) and *Two-Ray Ground propagation model* (TRG) to incorporate the variation in radio signals due to variation in terrain, frequency of operation, speed of mobile nodes, obstacles and other technical factors. Based on this combination, FS-TRG propagation model for representing the link reliability of a MANET as a function of distance has been proposed in (Peiravi and Kheibari 2008), and was later utilized in (Padmavathy and Chaturvedi 2013) for the evaluation of MANET reliability. However, the application of two-ray model in ad hoc networks seems implausible as here both transmitter and receiver are of comparable heights and the inter separation distance between transmitter and receiver is quite less. Recently, the *Wireless World Initiative New Radio* (WINNER) model has been utilized in (Coll-Perales et al. 2015) for urban scenarios with low antenna heights to study the energy consumption in Multi-hop Cellular Networks. Thus, in future studies, WINNER model can find potential applications in other TVCNs as well.

2.2.5 Topological Models

TVCN topology modelling is another crucial facet for understanding the underlying dynamics between nodes and the overall performance of the network. Among other models presented in this section and our contributed book chapter (Khanna et al. 2020a), *Random Geometric Graph* (RGG), *TAG*, *Space-time Graph* and *Line graph* have been extensively utilized by the researchers, and are of particular interest for this thesis. In the upcoming paragraph, we discuss salient features of each of them.

1) *Random Graph model*, originally introduced by Solomonoff and Rapoport in 1951 (Newman et al. 2006), can coherently describe many networks of arbitrary sizes. A random graph/network is formed by adding links having some associated probability between randomly selected node pairs. However, in TVCNs such a modelling is not viable as link existence between nodes is not only distance dependent but also depends on the time of the existence of the link. For instance, in a TVCN any node pair having lesser Euclidean distance between them in comparison to their transmission range will have high probability of link availability in comparison to a node pair having a larger distance between them (Hekmat 2006).

2) Gilbert's model (Díaz et al. 2011) commonly known as RGG is used to study the creation or snapping of edges in wireless networks, thereby their topology changes. The model considers that $|V|$ devices are placed uniformly at random in an area of $[0, 1]^2$; where V is a set of mobile devices and $|V|$ represents the number of devices in set V. Among $|V|$ devices, any two devices in the network can communicate with each other if their Euclidean distance is less than or equal to the transmission range r of the devices. This model is widely utilized as a simplified topological model for wireless sensor networks and MANETs.

3) The *Barabási-Albert model* (Caldarelli 2007) can suitably represent the time growth in many real world networks like social networks, Internet and biological networks, in which random graphs fail (Hekmat 2006). This model depends on two mechanisms, *viz.*, *growth*, as new vertices enter the network at some rate and *preferential attachment*,

as newly added nodes form link *preferentially* with nodes having a large degree. Such a scale-free network model, wherein the network size and node degree distribution are independent of each other (Hekmat 2006), is not appropriate for modelling TVCNs as it seems irrational to assume that some nodes may have a much higher number of neighbors than other in a coverage area with uniformly distributed nodes.

4) The *percolation theory* and *probabilistic epidemic algorithms* based models have been proposed in the literature to model *connectivity* (Chen 2014; Silva et al. 2015). These models can solve two dependent aspects, *viz.*, a good diffusion of information in the network (needed for routing, broadcast and communication) and its connectivity (needed to reach each node) (Shen et al. 2006; Amor et al. 2010). In (Li et al. 2015a), a percolation theory based framework was proposed to calculate the network reliability of random network models and real networks with different nodes and/or links lifetime distributions. To the best of our knowledge, the model has not been utilized for the reliability evaluation of TVCNs like VANETs, DTNs, and OMNs.

5) In *regular lattice graph model,* the radio communications form links between nodes. In this model, the adjacent nodes are connected with a probability p whereas non-adjacent nodes are indirectly connected via intermediate nodes. Besides, the probability of existence of a link varies as a function of the distance between the nodes to capture the decay of radio signal power with the increasing distance between nodes. Thus, this model can be suitable for modelling TVCNs (Hekmat 2006).

6) A *TAG* is used to model the changes in a spatio-temporal network, *e.g.*, road network, over time by collecting the node/edge attributes into a set of time series (George and Shekhar 2008). For example, observe Figure 2.1(i) showing a TVCN of period four time slots. More specifically, G_1 to G_4 in Figure 2.1(i) (a)-(d) respectively represent four sequential snapshots of the network taken at four different slots of time *viz.*, t_1, t_2, t_3 and t_4, where $t_1 < t_2 < t_3 < t_4$. Note that t_0 represents origin of the network evolution, and *slot length* is given by the difference between end time and start time of a slot. Thus, in essence, these snapshots are a sequence of static graphs representing

interactions between four different nodes at four different instants of time. Figure 2.1(ii) shows the TAG representation of the network evolution depicted in Figure 2.1 (i). More specifically, in Figure 2.1(ii) each edge accompanies a time-series representing edge activity schedule, *i.e.*, *contacts*. For example, observe from Figure 2.1(i) that edge between node pair (v_2,v_3) is active in slots 1 and 2. Figure 2.1(ii) shows the same bidirectional edges as $e_{2,3}^{\{1,2\}}$. Here, superscript denotes an ordered set {1, 2} of timestamps or contacts associated with bidirectional edge $e_{2,3}$; thereby, represents edge's activity schedule. Thus, TAG model is an effective medium to compactly depict a TVCN topology, while preserving all temporal information. It is worth mentioning that conventional *static* graph representation cannot effectively model time varying topology conditions, as it will obscure important temporal details of the network.

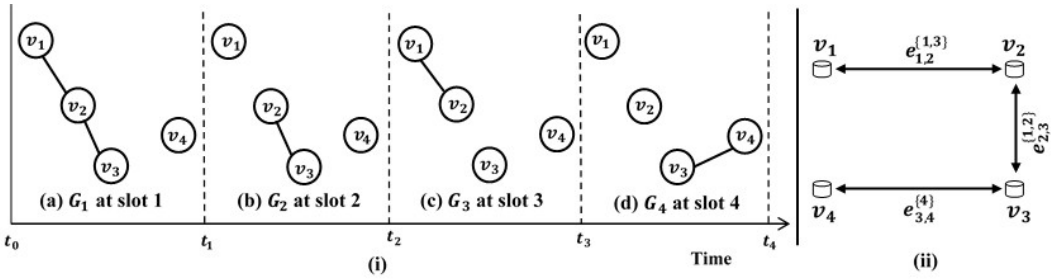


Figure 2.1. (i) A sequence of snapshots at four different time slots representing a TVCN; (ii) TAG representation of the network evolution shown in (i).

For example, the static graph of the TAG shown in Figure 2.1(ii) can be visualized by neglecting the contacts/timestamps over each edge. Thus, from this static graph, one can wrongly adjudge that there exists a path between nodes (v_4,v_2) via node v_3, *i.e.*, ($e_{4,3}.e_{3,2}$). However, when we consider temporal information corresponding to each edge, such a path can never exist. This happens because here each edge accompanying a *single* timestamp, called as *Timestamped Edge* (TSE), will not lead to any time-ordered or Timestamped Valid Minimal Path Set. Here onwards, for brevity, we call a Timestamped Valid Minimal Path Set as a TS-MPS. For example, timestamped path ($e_{4,3}^{4}.e_{3,2}^{1}$) is *invalid* with respect to time or can never exist as TSE $e_{4,3}^{4}$ is active in slot four; however, by that time its successive TSE $e_{3,2}^{1}$, having timestamp 1, ceases to exist, thereby making data transmission impossible between the specified pair of nodes.

Thus, it is important to note that as we cannot traverse backwards in time, the time-order of interaction between nodes is critical in TVCNs to facilitate a TS-MPS. Further, it is worth mentioning here that the notion of TSEs, along with TAG, effectively models *store-carry-and-forward* mechanism of data transmission between mobile devices. To illustrate this, let us consider a TS-MPS, *viz.*, $(e_{1,2}^{1}.e_{2,3}^{1}.e_{3,4}^{4})$ between nodes (v_1,v_4). Observe that data can be transmitted from node v_1 to v_3 via node v_2 in slot 1. The received data is then stored at node v_3 in slots 2 and 3 and is later forwarded to destination node v_4 in slot 4.

As can be seen from the aforementioned examples that TVCNs do not necessarily have an *end-to-end* connectivity at any given instant as it happens in static networks. Thus, they may not have any MPS/MCS and thereby will result in a zero reliability figure. However, it is a false assertion as there may be many opportunistic path sets with respect to time, *i.e.*, TS-MPS, which may result in successful transit of data packets between designated nodes within a finite window of time. Thus, motivated by the above, the scholar in (Chaturvedi et al. 2018; Khanna et al. 2019b, 2020b) extended the notion of an MPS of static networks to TS-MPS of TVCNs.

The formal definitions as proposed and utilized in the scholar's research are given below for better elucidation.

Definition 2.1: ***Timestamped Valid Minimal Path Set*** (TS-MPS) is a sub set of a TVCN's links appearing at different instants of time (in non-decreasing order) during an observational period ensuring and establishing a successful communication between a specified set of nodes. This set is said to be minimal in the sense that every element of this set is needed for ensuring a successful communication. In other words, success of every component in at least one TS-MPS ensures communication or network success. In this definition, the assumption of taking instants of time in non-decreasing order makes sure that we do not move into path sets, which have already occurred in the past.

Similarly, an MCS equivalent in TVCNs is *Timestamped-Minimal Cut Set* (TS-MCS) (Khanna et al. 2020b), whose formal definition is discussed below.

Definition 2.2: ***Timestamped Minimal Cut Set*** (TS-MCS) is a sub set of a TVCN's links appearing at different instants of time during an observational period such that their removal or non-functioning at those particular instants of time ensures network's dis-connectivity for a specified set of nodes, provided that removal of any proper subset of these links does not disconnects the network graph. In other words, the set is minimal because the system does not fail if any one or more of the TSEs do not fail. Thus, failure of every component in at least one TS-MCS ensures communication or network failure.

The idea is extended further by the author in (Khanna et al. 2020c) to model time-ordered arborescences, originating from a source node in TVCNs, for network broadcasting. More specifically, we consider two types of arborescence of TVCNs, *viz.*, *Timestamped Arborescence* or t*-arborescence*, and *Timestamped Valid Arborescence* or tv*-arborescence*. Their definition is given below.

Definition 2.3: A ***Timestamped Arborescence*** or t*-arborescence* is an arborescence where each of its directed edge has a timestamp that denotes the edge's activity or schedule. We call a t*-arborescence* as *valid* t*-arborescence, i.e.,* ***Timestamped Valid Arborescence*** or a tv*-arborescence* if each edge of the arborescence has timestamp no earlier than its predecessor edge, *i.e.*, all edges are *time-ordered*. Otherwise, it is an *invalid* t*-arborescence*.

To illustrate our Definition 2.3, consider Figure 2.2. Observe that without considering timestamps we have only one arborescence rooted at node v_1, *i.e.*, $(e_{1,2}\,.\,e_{2,3}\,.\,e_{3,4})$. However, given timestamps, we have four t*-arborescences*. Out of these, only two are tv*-arborescences*; see solid edges from node v_1 in Figure 2.2(i) and (ii).

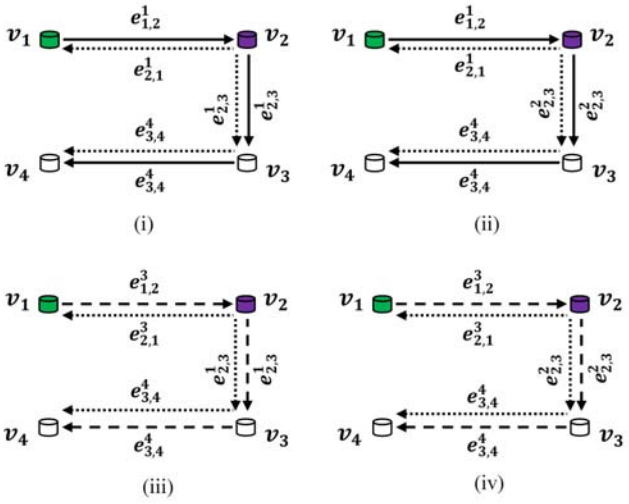


Figure 2.2. Timestamped arborescences from node v_1 and node v_2 for the TAG shown in Figure 2.1(ii); arborescences shown in solid and dotted lines are tv-arborescences, while those in dashed lines are invalid t-arborescences.

Notice that $(e^1_{1,2}.\, e^2_{2,3}.\, e^4_{3,4})$ is a tv-arborescence as edge $e^1_{1,2}$ is active no later than $e^2_{2,3}$. Note that *t-arborescence* $\left(e^3_{1,2}.\, e^1_{2,3}.\, e^4_{3,4}\right)$ of Figure 2.2(iii) is *invalid* because TSE $e^1_{2,3}$ exists at time $t = 1$, which is earlier than its predecessor TSE $e^3_{1,2}$ that is active at time $t = 3$, *i.e.*, the edges are not time-ordered. As shown by the dotted lines in Figure 2.2, there are four tv-arborescences from root node v_2, *i.e.*, each t-arborescence is a tv-arborescence. In Chapter 4, we present two methods for enumerating all tv-arborescences of a TVCN. Thus, owing to the effectiveness of TAG model, we utilize it later for enumerating all TS-MPS, TS-MCS and tv-arborescences, respectively.

7) A *Space-time Graph* (Huang et al. 2013; Chaturvedi et al. 2018) combines a sequence of static network graphs, as shown in Figure 2.1(i), to represent them as a directed graph defined in both spatial and temporal space. Thus, the model adequately captures both space and time dimensions of the TVCN topology and can display *all* TS-MPS, between every pair of nodes, and tv-arborescences. More specifically, to capture *transmission*, *reception* and *storage* at a node $v_i \in V$ in each time slot $t_x \in [1, \tau]$, the model uses two nodes, *viz.*, $v_i^{t_x,\mathcal{T}}$ and $v_i^{t_x,\mathcal{R}}$. Note that τ depicts the period of TVCN and $v_i^{t_x,\mathcal{T}}(v_i^{t_x,\mathcal{R}})$ represents data *transmitting* (*receiving*) node v_i in slot t_x. For convenience, the model also includes two virtual nodes v_i^0 and $v_i^{\tau+1}$ for each node v_i

as the starting point and ending point, respectively, of the time span. Further, two types of edges, *i.e.*, spatial edges and temporal edges, are added in each layer of space-time graph; see dotted box for each time slot in Figure 2.3 and Figure 2.4.

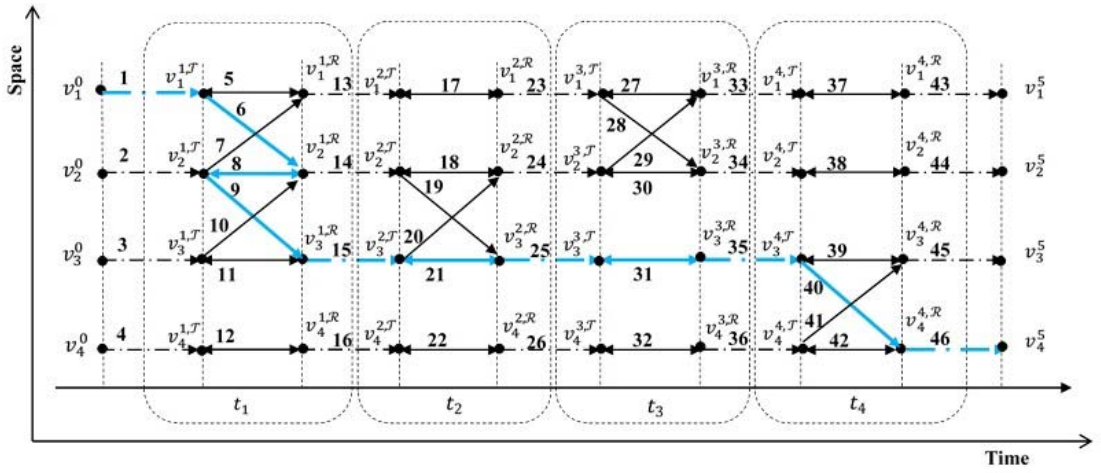


Figure 2.3. Space-time graph, of the TVCN shown in Figure 2.1(i), with multi-hop communication capability.

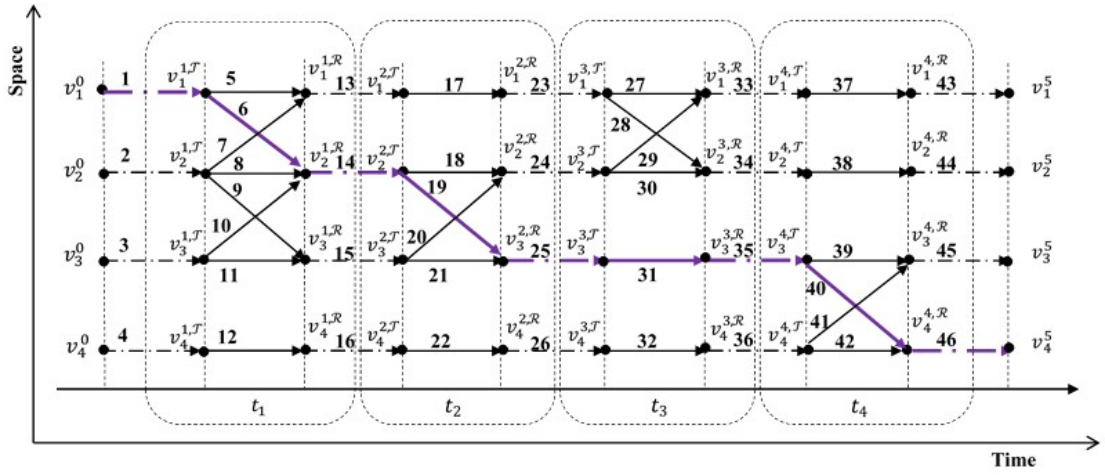


Figure 2.4. Space-time graph, of the TVCN shown in Figure 2.1(i), with single-hop communication capability.

A horizontal line, *i.e.*, temporal edge, either represents storage of packets at a node or a virtual edge connecting two consecutive time slots. On the other hand, a slanted line, *i.e.*, spatial edge, within a slot represents exchange of packets between two nodes at the given time instant. More specifically, in Figure 2.3, a bidirectional horizontal edge, $\overleftrightarrow{v_i^{t_x,\mathcal{T}} v_i^{t_x,\mathcal{R}}}$ within slot t_x is a temporal edge, which represents storage of packets at node

v_i in slot t_x. This bidirectional edge model is helpful in representing multi-hop communication capability amongst nodes because if the model uses unidirectional horizontal temporal edge $\overrightarrow{v_i^{t_x,\mathcal{T}} v_i^{t_x,\mathcal{R}}}$, then only single-hop transmission would be possible within any slot; see Figure 2.4. The reason is that in the latter case any node cannot behave as a transmitter and receiver simultaneously. A horizontal edge between slots t_x and t_{x+1}, *i.e.*, a temporal edge $\overrightarrow{v_i^{t_x,\mathcal{R}} v_i^{t_{x+1},\mathcal{T}}}$ is a virtual edge connecting two consecutive time slots. A non-horizontal edge inside slot t_x is a spatial edge $\overrightarrow{v_i^{t_x,\mathcal{T}} v_k^{t_x,\mathcal{R}}}$, which represents progression of packet(s) from node v_i to its neighbor v_k in time slot t_x. For better understanding of this representation, let us again consider Figure 2.1(ii) and observe the TS-MPS between nodes (v_1,v_4), *viz.*, ($e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4$). Figure 2.3 has an equivalent path as (1-6-8-9-15-21-25-31-35-40-46), wherein edges 6, 9 and 40 correspond to TSEs $e_{1,2}^1$, $e_{2,3}^1$ and $e_{3,4}^4$, respectively. All other edges in the path are either virtual edges or edges representing data storage at node itself. On the other hand, if only single-hop communication is possible within a slot then Figure 2.4 is used to model TS-MPS. For instance, see highlighted path (1-6-14-19-25-31-35-40-46) depicting ($e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4$) in Figure 2.1(ii). The reader can verify by visual inspection that Figure 2.1(ii) and Figure 2.3 can represent both TS-MPS ($e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4$) and ($e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4$), while Figure 2.4 cannot represent TS-MPS ($e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4$). It is worth mentioning here that although space-time graphs can represent TVCNs over time and space; however, as the size of network increases, space-time graphs become quite unwieldy with many virtual nodes and edges in the structure.

8) *Line Graph* (Liang and Modiano 2017) is a useful modelling tool to convert a TAG representation into a conventional static graph without loss of any temporal reachability information. Note that each TSE of the TAG shown in Figure 2.1(ii) is represented as a node in a Line Graph; see Figure 2.5. Observe that the Line Graph successfully models connectivity over time between nodes v_1 and v_4. It is worthy to note that leaf $e_{1,2}^3$ has been pruned and not shown in the diagram as it does not lead to any TS-MPS from node v_1 to node v_4. The algorithm to generate such a Line Graph can be seen from (Liang and Modiano 2017; Khanna et al. 2020b). Note that this graph model is

helpful not only in enumerating TS-MPS, but also TS-MCS (Khanna et al. 2020b), whose failure leads to (s, d) pair disconnection. For example, in Figure 2.5, the two TS-MPS between node pair (v_1, v_4) can be observed as ($e^1_{1,2}.e^1_{2,3}.e^4_{3,4}$) and ($e^1_{1,2}.e^2_{2,3}.e^4_{3,4}$), while the three TS-MCS are $e^1_{1,2}$, $e^4_{3,4}$ and ($e^1_{2,3}.e^2_{2,3}$). Further, without loss of generality, Line Graph model can also be modified to include time latency, *i.e.*, to show only single hop communication capability.

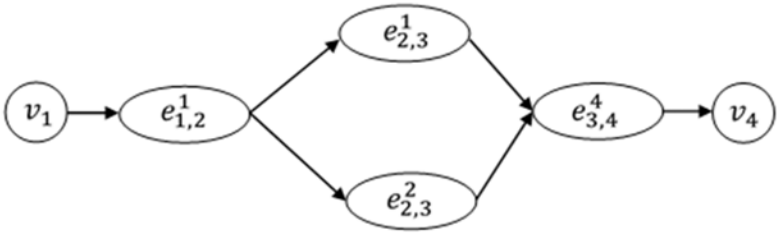


Figure 2.5. Line Graph of the TAG shown in Figure 2.1(ii).

Other alternative representations of TVCNs include *adjacency tensors*, *affine graph*s, and *time line of contacts* (Holme 2015).

2.3 Topology Control and Contact Plan Design

Topology control problem has widely been studied in ad hoc and sensor networks. In TVCNs, topology control problem can be utilized to build a sparsely connected network. Such a sparse network not only reduces the number of edges from the original TVCN, but also helps in reducing radio interference and achieving different objectives like reduction in the total cost of the network. In this regard, authors in (Huang et al. 2013) presented two greedy-based solutions to maintain connectivity over time while reducing the overall cost of the network. In other words, they constructed a sparse topology with at least one directed path between each pair of nodes and deterministic underlying connections. Authors in (Chen and Shi 2015), later modified this by utilizing probability to define the underlying connections between nodes of a TVCN. They presented two heuristic algorithms to find a sub graph to balance the energy cost and data transferring reliability. In (Li et al. 2015b), authors presented heuristic solutions to generate a sparse topology which has a path between any pair of nodes with reliability higher than a given threshold while maintaining the

minimum total cost of the structure. Authors in (Chen et al. 2016) defined the topology control problem specifically for network convergecasting in TVCNs. They proposed to find a directed spanning tree which has the minimum energy cost and simultaneously satisfies the time delay threshold.

The problem of *contact plan design* introduced in (Fraire and Finochietto 2015a, b) for satellite constellations is similar to the topology control problem of TVCNs. In space borne TVCNs, a *feasible contact plan design* is crucial for judiciously utilizing the available resources like energy and number of transponders, and thus *contacts*. Usually, a Mission and Operation Control Centre deterministically computes the expected *contacts* among satellite constellations by using the orbital mechanics and radio models (Fraire and Finochietto 2015b). Authors in (Fraire et al. 2014) presented an optimization problem to design a fair *contact plan* for satellite constellations. In (Fraire et al. 2017b), authors presented an evolutionary algorithm to provide sub-optimal, yet efficient contact plans in bounded time. Similarly, in (Zhou et al. 2017) authors presented an approach to design a feasible contact plan by considering distinctiveness of different missions, energy related issues, and time-varying satellite downlink contacts. They also examined the effect of energy collection rate, battery capacity, and buffer size on the network profit. *Contact plan design* related other similar works can be seen from (Wang et al. 2016; Dai et al. 2018). In Chapter 6 of this thesis, we consider a different type of topology control and contact plan design for TVCNs.

2.4 Simulators, Emulators and Testbeds for TVCNs

Increasing research efforts towards TVCNs require development of new performance evaluation tools. Over the time, the researchers have developed various network simulators/visualization tools and packages, emulators and testbeds for studying the dynamics and complex processes occurring in networks emanating from different fields of study such as social science, biology, citation networks, World Wide Web, Internet, and communication networks. The following sub-sections gives an extensive taxonomy of these simulation/visualization tools, emulators and testbeds, respectively.

2.4.1 Simulation Tools

Simulation is widely used in the domain of wireless networks, where one can find a number of free and commercial simulation platforms. Table 2.3 collates 22 popular simulation and visualization tools from the literature along with a brief description of each of them. Out of these 22 simulators, *ns-2* and *OMNeT++* have widely been used for studying the performance of various routing protocols in MANETs, while *ONE* is a well-known DTN simulator. Further, a number of simulation tools to mimic the *node movement* are available in the literature. These tools can generate and analyze the impact of human or vehicle mobility under various traffic scenarios, and have widely been used by the researchers, consultants and government agencies for some specific applications. Table 2.4 summarizes various mobility simulation tools proposed in the literature.

2.4.2 Emulation Tools and Testbeds

A rich literature is present on the topic of emulators and testbeds (Kropff et al. 2006; Kiess and Mauve 2007; Ivanic et al. 2009; Patel and Jhaveri 2015). Choosing an appropriate emulation/testbed platform for validation of research results is an endeavor on its own. There is no panacea for all possible application scenarios, but a compromise between characteristics, such as scalability, realism, performance, and cost has to be reached. Table 2.5 provides a compilation of 14 emulators and testbeds (along with their features) which are available for investigating and modelling TVCNs. However, from the available literature it seems impossible to identify any ace emulator/testbed.

2.4.3 Real World Project Implementations

This section takes out from archives some real life projects, carried out with prime objective to study TVCNs and to develop new Information and Communication Technology (ICT) based services by addressing the challenges encountered in their implementation. Table 2.6 collates information of 14 projects along with their important features and funding sources. Note that these projects have helped immensely: i) to collect real mobility traces of humans and vehicles, and ii) to provide internet connectivity and communication facility in the remote areas and small towns of developing countries like India, Cambodia and Kenya.

Table 2.3. Simulators for modelling and visualizing TVCNs (Khanna and Chaturvedi 2018).

Sl. No.	Name	License Type	Brief Description
1.	Adyton	Free	A simulation software for opportunistic networks; simulates a plethora of routing protocols and real-world contact traces.
2.	DTNTES	Free	A trace analysis tool used to extract various information from real trace like DieselNet.
3.	Gephi	Open source and Free	A powerful network/graph analysis software which can generate and analyze time varying graphs, and propose a timeline component where a slice of the network can be retrieved.
4.	GlomoSim	Open source	GloMoSim is a library-based sequential and parallel simulator designed as a set of modules in the layered architecture for wireless networks; each module simulates a specific protocol in the protocol stack.
5.	ICONE	Free	Modified ONE simulator is a collection of add-ons that transform the original ONE simulator into a simulator having Named Data Networking (NDN) capability.
6.	Igraph	Open source and Free	A powerful package/library which can be embedded into a higher level programming language (like Python, Perl or GNU R) and is capable of handling huge graphs with millions of vertices and edges. Various igraph libraries can be accessed from Github repository.
7.	JiST/SWANS	Free	JiST is a high performance java based discrete simulator and SWANS is a complete library for simulation of MANETs running on the JiST engine.
8.	J-Sim	Open source	An independent, extensible, and reusable modelling and simulation framework/environment developed in Java for Wireless sensor networks (WSNs).
9.	KUMONOTE	Free for non-commercial use	A powerful network/graph analysis software which can plot degree distribution, generate randomized graph, extract giant connected component, convert the given network in format

Sl. No.	Name	License Type	Brief Description
			which can be analyzed with Pajek and visualize the network connectivity.
10.	MADHOC	Free	Designed for implementing new environments, new mobility schemes and new applications in Delay Tolerant MANETs.
11.	MATLAB	Commercial	A high-performance, multi-paradigm numerical computing environment for computation, visualization, and programming. The MATLAB can be used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more.
12.	NetMiner	Commercial	A network analysis software package which helps to perform data transformation, network analysis, statistical analysis, and network visualization.
13.	ns-2	Open source	Event-driven simulator designed specifically for research in computer communication networks. It now contains modules for numerous network components such as routing, transport layer protocol, application, etc.
14.	ns-3	Open source	A discrete-event network simulator designed to cater the needs of modern networking research for Internet systems; ns-3 is intended to eventually replace the popular ns-2 simulator by a modern simulator targeted primarily for research and educational use.
15.	OMNeT++	Free for non-profit use	Discrete event simulation platform, designed to support the simulation of telecommunication networks and other parallel and distributed systems.
16.	ONE	Free	A well-known DTN simulator designed for evaluating DTN routing and application protocols. It allows users to create scenarios based upon different synthetic movement models and real-world traces.

Sl. No.	Name	License Type	Brief Description
17.	OPNET/ Riverbed Modeler	Commercial version, or Free to qualifying universities worldwide for academic research and teaching	Discrete-event simulator to design, model, and analyze communication networks. It can model various networks and technologies such as VoIP, TCP, OSPFv3, MPLS and IPv6, etc.
18.	Pajek	Free for non-commercial use	Extremely popular program package for analysis and visualization of large networks.
19.	Ptolemy-II	Open source	Can model, simulate and design concurrent, real-time, embedded systems with focus on assembly of concurrent components.
20.	Qualnet	Commercial	A simulation platform which assists in simulating the behaviour of real communications networks, allows user to design, analyze, and optimize various scenarios.
21.	SoNIA	Free	A powerful simulation platform for testing and comparing layouts and techniques, and for browsing attribute-rich network data for animating network dynamics over time.
22.	SSFNet	Open source, Free or available at nominal cost for research purposes.	Java SSF-based discrete-event simulator for modelling and simulation of Internet protocols and of large and complex systems at and above the IP packet level of detail.

DTNTES: Delay **T**olerant **N**etwork **T**race **E**valuator **S**ystem; **GlomoSim: Glo**bal **Mo**bile System **Sim**ulator; **ICONE: I**nformation **C**entric **O**pportunistic **N**etwork **E**nvironment; **JiST/SWANS: J**ava **i**n **S**imulation **T**ime/ **S**calable **W**ireless **A**d hoc **N**etwork **S**imulator; **MADHOC: M**etropolitan **Ad hoc** Network Simulator; **MATLAB: Mat**rix **Lab**oratory; **ONE: O**pportunistic **N**etwork **E**nvironment simulator; **SoNIA: So**cial **N**etwork **I**mage **A**nimator.

Table 2.4. Tools for mobility modelling (Khanna and Chaturvedi 2018).

Sl. No.	Tools	Brief Description	Type	Model Simulation Capability
1.	BonnMotion	A Java based mobility scenario generation and analysis tool. The generated mobility scenarios may also be exported in wireless network simulators like ns-2, GloMoSim, or QualNet.	Free	Random-Waypoint, Random-Walk, Gauss-Markov, Manhattan-Grid, Reference-Point-Group, Disaster Area, Random-Street.
2.	CanuMobiSim	It is a flexible framework for user mobility modelling. It can simulate spatial environment, user trip sequences, and user movement dynamics with the meta-mobility models and vehicular dynamics.	Free	Random-Waypoint, Brownian-Motion, Meta-Mobility
3.	CORSIM/TSIS	Software tool for simulating traffic on surface streets, freeways, and integrated networks. It is a combination of two traffic simulation models, NETSIM for surface streets, and FRESIM for freeways.	Commercial	Vehicle and driver behaviour
4.	HUMsim	A generator of realistic, synthetic trace of human mobility.	--	Behavioural
5.	IMPORTANT	A framework to understand and analyze the impact of mobility pattern on routing performance of MANETs.	--	Random-Waypoint, Reference-Point-Group, Freeway, Manhattan-Grid.
6.	MobiSim	A simulator for performance evaluation of various network routing protocols in MANETs.	Free	Random-Waypoint, Reference-Point-Group, Gauss-Markov, Random-Walk, Freeway, Manhattan-Grid.
7.	PTV Vissim	A powerful tool for the evaluation and planning of urban and extra-urban transport infrastructure with capability to simulate various traffic patterns originating from motorized private transport, goods transport, rail and road related public transport, pedestrians and cyclists.	Commercial	Realistic motion model of all road users

Sl. No.	Tools	Brief Description	Type	Model Simulation Capability
8.	Quadstone Paramics	It is a leading simulator for traffic simulation and transportation infrastructure planning. It can simulate both pedestrian and vehicular traffic and is powerful enough to model an entire city's traffic system.	Commercial	Pedestrian and Vehicular traffic
9.	SUMO	Allows simulation and modelling of intermodal traffic systems. It can be used in route finding, visualization, and network import and emission calculation.	Free and open source	Pedestrian and Vehicular traffic
10.	Toilers-Code-Base	It is a rich collection of a number of published works along with associated codes for the models and services developed thereby.	Available on demand	Random-Waypoint, Random-Walk, Reference-Point-Group, Gauss-Markov, Random-Direction
11.	TraNS	A GUI tool that integrates traffic and network simulators (SUMO and ns-2) to generate realistic simulations of VANETs.	Open source	Vehicular traffic
12.	TRANSIMS	An integrated system of travel forecasting tools for planning transport systems and emissions analysis.	Free and open source	Transport system
13.	VanetMobiSim	It is an upgraded tool similar to CanuMobiSim. It supports new realistic automotive mobility models for studying and understanding, car-to-car and car-to-infrastructure interactions.	Free	New realistic automotive mobility models

CanuMobiSim: CANU Mobility Simulation Environment; **CORSIM/TSIS: Corridor Simulation/Traffic Software Integrated System**; **HUMsim: Human** Urban Mobility **Simulator**; **IMPORTANT: Impact** of **Mobility** on the **Performance Of RouTing** protocols in **Adhoc NeTworks**; **SUMO: Simulation** of **Urban Mobility**; **TraNS: Traffic** and **Network Simulation** Environment; **TRANSIMS: TRansportation ANalysis** and **SIMulation System**.

Table 2.5. Testbeds and Emulators for modelling and visualizing TVCNs (Khanna and Chaturvedi 2018).

Sl. No.	Name	Brief Description	Reported Size	Mobility Modelling		Testbed	Emulator
				Real	Logical Connectivity		
1.	APE	Enables large-scale evaluations of ad hoc routing protocols where experiments are repeatable. It has a non-intrusive measurement system that is fully integrated into the testbed environment.	37 physical nodes	✓		✓	
2.	DAWN	Provides network-layer support for real-time QoS in large and dense mobile ad hoc networks.	10 physical nodes	✓		✓	
3.	EWANT	Allows emulation of RF propagation effects of a huge network spread over hundreds of meters in a very small area, and mobility via an antenna switching mechanism.	4 physical nodes	✓		✓	
4.	HaggleSim	Can replay the mobility traces collected and emulate different forwarding strategies on every contact event.	--	✓			✓
5.	JEmu	Designed to enable the ad hoc routing protocol designer to create and manipulate wireless scenarios with ease.	12 physical nodes		✓		✓
6.	MANE/eMANE	Suite of tools and applications that can be combined and extended as necessary to provide a high-fidelity, scalable emulation environment for mobile wireless networks.	~1000 virtual nodes		✓		✓

Sl. No.	Name	Brief Description	Reported Size	Mobility Modelling		Testbed	Emulator
				Real	Logical Connectivity		
7.	ManTS	Can build large-scale MANETs, and enables researchers to estimate their IP and upper layers' protocols and applications intuitively, as well as by collected data.	--		✓	✓	
8.	MASSIVE	An emulator which follows the distributed concept, supports interactive manipulation of the emulation process and allows the coupled evaluation of routing protocols and applications.	13 physical nodes		✓		✓
9.	MiNT	Platform for testing and evaluating wireless applications and protocol implementations.	8 physical nodes	✓			✓
10.	ORBIT	Aims to provide a flexible, open-access multi-user experimental facility to support research on next-generation wireless networks.	400 physical nodes	✓		✓	
11.	SALT/PC-NETSIM	Allows the actual network protocol and operating system code to be tested quickly.	--		✓	✓	
12.	SALT/PRISM	Permits software control of RF connectivity, supports creation of network topologies such as strings, stars, dense groups, or mobile nodes.	24 physical nodes		✓	✓	
13.	TEALab	Provides an environment for studying network attacks and attack detection in ad hoc networking environments.	10 to 30 physical nodes	✓		✓	

Sl. No.	Name	Brief Description	Reported Size	Mobility Modelling		Testbed	Emulator
				Real	Logical Connectivity		
14.	TrueMobile	Remotely accessible mobile wireless and sensor testbed which can provide accurate positioning and monitoring; can enable automated experiments by both on-site and off-site users.	16 physical nodes	✓		✓	

APE: **A**d hoc **P**rotocol **E**valuation; **DAWN:** **D**ensity-and **A**symmetry-adaptive **W**ireless **N**etwork; **EWANT:** **E**mulated **W**ireless **A**d Hoc **N**etwork **T**estbed; **MANE/eMANE:** **M**obile **A**d hoc **N**etwork **E**mulator/ **E**xtensible **M**obile **A**d-hoc **N**etwork **E**mulator; **ManTS:** **T**est **S**ystem for **MAN**ET; **MASSIVE:** **Ma**net **S**erver **S**uite **I**ncorporating **V**irtual **E**nvironments; **MiNT:** **Mi**niaturized **N**etwork **T**estbed; **ORBIT:** **O**pen Access **R**esearch Test**b**ed for Next-Generation Wi**r**eless Ne**t**works; **SURAN:** **Sur**vivable, **A**daptive **N**etworks; **SALT/PC-NETSIM:** **S**URAN **A**utomated **L**aboratory **T**estbed/**PC**-based **net**work **sim**ulator; **SALT/PRISM:** **S**URAN **A**utomated **L**aboratory **T**estbed/**P**acket **R**adio **I**ntegrated **S**ystem **M**odule; **TEALab:** **T**actical **E**nvironment **A**ssurance **Lab**oratory.

Table 2.6. Some real life projects implementing TVCNs (Khanna and Chaturvedi 2018).

Sl. No.	Project Name	Brief Description	Funded by	Device	Scenario/ Environment	Start	Duration
1.	Challenged Internet Access Network Technology Infrastructure (CHIANTI)	The main objective of this project was to provide services like internet connectivity in fast moving vehicles such as trains, by integrating and expanding existing core internet architecture and developing solutions for mobility support.	European Union	Existing Internet and CHIANTI software	Nomadic, and vehicular	2008	2 Years
2.	Continuous Displacement Plans Oriented Network (CoDPON)	Aims to provide electronic health care for riverine communities of Marajó Archipelago, located in the north of Brazil, through VANETs.	FIDESA	Boats, Peers Base Stations and Hot Spots	Riverine	2010	Ongoing
3.	DakNet	Aims to provide affordable Internet services to citizens in remote areas of developing countries like India and Combodia.	NGOs like American Assistance for Cambodia/Japan Relief Fund (AAfC/JRF) and governments	Mobile Access Points (MAPs)	Remote regions	2003	Ongoing
4.	Haggle	Designed to enable communication in the presence of intermittent network connectivity, which exploits autonomic and opportunistic	European Commission under the Information Society	iMotes or Smart phones	Transportation and remote regions	2006	4 Years

Sl. No.	Project Name	Brief Description	Funded by	Device	Scenario/ Environment	Start	Duration
		communications. Four experimental data sets were gathered by the project, and are referred to as Infocom05, HongKong, Cambridge, and Infocom06.	Technologies Programme				
5.	Invisible Dynamics: Cabspotting	Online system to anonymously track and record the movements of Yellow Cab vehicles throughout the greater San Francisco Bay Area.	The Exploratorium of San Francisco	Global positioning system (GPS), Taxi cabs	Urban street	2008	30 Days
6.	MIT Reality Mining Project	Used mobile phone data to extract patterns that predict future human behaviour. It is first mobile dataset with rich personal behaviour and interpersonal interactions.	(--)	Bluetooth and Mobile phones	Campus and laboratory	2004	9 Months
7.	Sámi Network Connectivity (SNC)	Aimed to provide internet connectivity to reindeer herders in the remote areas of Laponia region in northern Sweden.	Interreg programme	Tablet PCs, laptops, radio, and data relays.	Mountains	2002	5 Years
8.	SARAH	Started with an objective of supporting communication in Delay-Tolerant Mobile Ad Hoc Networks (DT-MANETs). The project also focused on the security issues cropping up in the delay tolerant environment due to	French National Research Agency (ANR)	Laptops, smartphones, vehicles or robots	Rural, urban or wildlife habitat	2005	3.5 Years

Sl. No.	Project Name	Brief Description	Funded by	Device	Scenario/ Environment	Start	Duration
		the lack of end-to-end connectivity.					
9.	Sensor Networking with Delay Tolerance (SeNDT)	A project using sensor nodes for environmental monitoring *viz.*, lake water quality monitoring and roadside noise monitoring; designed for public authorities, NGOs and/or corporates.	Enterprise Ireland	SeNDT nodes	Rural and urban areas and motorways	2002	2 Years
10.	Terminodes	The project identified technical challenges in design of large-scale, self-organized MANETs and primarily focused upon studying, prototyping and finding feasible solutions for them.	Swiss	Terminodes	Wide area (city, region or country)	2000	10 Years
11.	Technology and Infrastructure for Emerging Regions (TIER)	Aims to address the challenges in bringing the Information Technology revolution to the masses of the developing regions of the world. The project utilizes custom-tailored hardware/ software, specific for a region, for providing internet and communication facility.	USAID, Blum Center for Developing Economies and Philippine-California Advanced Research Institutes Project	Custom tailored hardware/ software	Rural and small town	2004	Ongoing
12.	UMass DieselNet	A vehicle based delay tolerant communication network operating in a micro-urban area around Amherst, MA, by the Amherst branch of the	DARPA and National Science -	GPS, HaCom Open Brick computer, and Buses	Campus	2004	3 Years

Sl. No.	Project Name	Brief Description	Funded by	Device	Scenario/ Environment	Start	Duration
		Pioneer Valley Transport Authority (PVTA).	Foundation (NSF)				
13.	Underwater Acoustic networks (UAN)	Aimed to develop and test underwater wireless network for monitoring and protection of critical infrastructures located by the sea.	European Union	Autonomous Underwater Vehicles (UAVs)	Underwater	2008	3 Years
14.	ZebraNet	Focused upon wildlife tracking across large regions of Mpala Research Centre in central Kenya. The project integrated computing, wireless communication, and non-volatile storage along with GPS and other sensors.	National Science Foundation (NSF) Information Technology Research (ITR) grant	GPS, Flash memory, wireless transceivers and a small CPU	Wildlife habitat	2004	~ 1.5 Years [Actual deployment for approximately 17 days]

(--): No information is available

2.5 Difference: TVCN and Static Network

The primary dissimilarities between TVCNs and static networks are:

1) Unlike in static networks, it is possible in TVCNs that some nodes may never ever be able to connect themselves via single (direct) or multiple hops (through relay nodes), yet they are able to communicate with each other *via* device-to-device communications in *store-carry-and-forward* fashion within a finite time frame.

2) In each TS-MPS and tv-arborescence of a TVCN the links have a non-decreasing order of timestamps; but, such a labelling scheme is absent in an MPS and conventional arborescence. It is also important to keep in mind that in TVCNs the timestamps are not necessarily consecutive.

3) A sub path of a shortest MPS is shortest; however, it may not be true for TS-MPS. Thus, existing algorithms for enumerating shortest MPS cannot be directly applied to enumerate TS-MPS (Huang et al. 2014). For illustration, in Figure 2.6, consider a TS-MPS ($e^1_{1,2}.e^1_{2,4}.e^2_{4,3}$), (though not shortest!) from node v_1 to node v_3 via node v_2 and v_4, then its timestamped sub path ($e^1_{1,2}.e^1_{2,4}$) from node v_1 to node v_4 via node v_2 is not the shortest timestamped sub path.

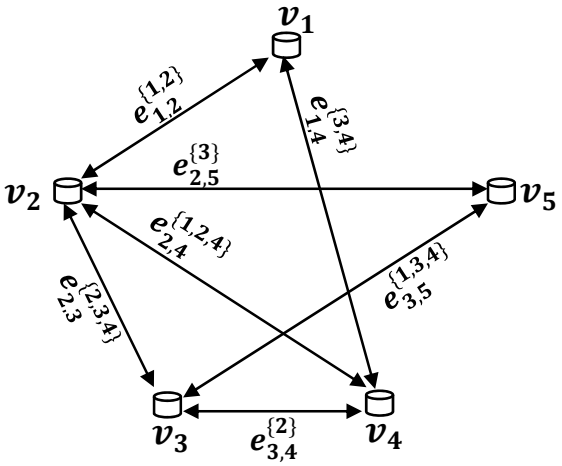


Figure 2.6. A TAG of a TVCN of five nodes at four different time slots.

It is true as TSE $e^3_{1,4}$ uses only one hop from node v_1 to reach node v_4 and is shorter; however, $(e^3_{1,4}.e^2_{4,3})$ is an invalid timestamped path between node v_1 and v_3 as the timestamp of successive edge, *i.e.*, $e^2_{4,3}$ is less than that of the previous edge, *i.e.*, $e^3_{1,4}$. In other words, path $(e^3_{1,4}.\, e^2_{4,3})$ is not a TS-MPS, as its edges do not obey the time order.

4) In TVCNs, if there exists a path from node v_i to node v_j within a designated time window then it is not necessary that the reverse is also true; however, it is by default true in static networks with bidirectional links. This asymmetry, which arises even though each discrete snapshot of a TVCN is symmetric follows from 2) above, and is a result of the direction induced by the inclusion of time explicitly into the network model. For example, consider a hypothetical timestamped path say $(e^a_{s,j}.\, e^b_{j,k}.\, e^c_{k,d})$ between a (s, d) node pair. Recall that the considered path will be valid, *i.e.*, a TS-MPS, if and only if the timestamp a is less than or equal to the timestamp b which in turn should be less than or equal to the timestamp c. Now, if we reverse the direction of data flow then the path shall become invalid if equality does not hold. Therefore, if the considered path is the only path set between the (s, d) node pair or if all other paths also have similar time-ordering of edges then there shall be no TS-MPS to transfer data packets in the reverse direction, *i.e.*, from v_d to v_s.

Due to the above mentioned differences and existing limitations of static networks, because of this dynamic paradigm, the present trends show that researchers are focusing on modelling TVCNs along with time by using TAGs a.k.a. evolving graphs. In summary, it is worthy to appreciate that simply aggregating the snapshots without any information about edge's activity presents a very misleading conclusion. For example, in Figure 2.6, if we do not consider timestamps over edges, then node pair (v_1, v_3) will be connected via node v_4. However, as discussed above, upon considering TSEs such a path can never exist in reality.

2.6 Network Reliability: An Overview

This section presents some important definitions, existing reliability models and techniques for evaluating the network reliability of static networks and some categories of TVCNs.

2.6.1 Important Definitions and Metrics

Network reliability is defined as the probability that a specified set of designated nodes are able to communicate with each other within the intended design features and environments. Depending on the number of nodes in the specified set, the network reliability metric can be termed as *2-terminal*, *k-terminal* or *all-terminal reliability*. The *k-terminal reliability* is more general because $k = 2$ indicates the first metric, while $k = |V|$ (all nodes of the network) yields *all-terminal reliability*. Network survivability, defined as the ability to survive a certain number of failures (Liang and Modiano 2017), is another important metric frequently used to study the fault tolerance of networks. Most works on network reliability and survivability assume that the failures of the units (nodes and/or links) are statistically independent and the failure of a unit does not influence the hazard rate of the surviving units and thereby the subsequent failures (Misra 1992). Further, it is commonly assumed that each unit has a probability of either being operational or failed, *i.e.*, has two-states (Misra 1992). Besides an operational unit may be assumed to be *perfect*, *i.e.*, theoretically 100% reliable or *imperfect*, *i.e.*, having a reliability of less than 100%. It is also worth mentioning here that link and node reliability usually depends on factors like time, environmental interference, transmission power, signal-to-noise ratio and Euclidean distance between the interacting nodes, while node reliability may also additionally include factors like its capacity, battery power, packet size, etc.

2.6.2 Reliability Evaluation – Static Networks

Since 1970, evaluation of reliability measures of static networks has drawn a lot of attention from the researchers (Misra 1992; Chaturvedi 2016). Decades of comprehensive research in the domain has resulted in the evolution of many diverse, well-developed and understood graph-theoretic based methods for the reliability modelling and evaluation of static

networks. These methods have broadly been categorized as: (i) An MPS or MCS based, *e.g.*, *inclusion-exclusion, calculation of bounds, domination theory, SDP*, *binary decision diagrams* (BDD) and *ordered* BDD (OBDD), and (ii) Non-MPS or non-MCS based, *e.g.*, *state enumeration technique*, *factoring theorem* and *network transformation* (Chaturvedi 2016). Both categories have their well-documented merits and demerits, however, first category is simple, comparatively efficient and economical, and less restrictive than the other. It may also be intriguing to note that it has been shown in (Zang et al. 2000; Xing and Amari 2015) that BDD performs better than SDP in case of large networks; however, generation of BDD is highly dependent on the ordering of variables. Note that finding an optimal ordering of the variables is a co NP-complete problem. So, there is no panacea and both SDP and BDD techniques are quite popular amongst researchers.

Numerous methods have been proposed in the literature to enumerate all MPS and MCS (Ahmad 1988; Chaturvedi and Misra 2002; Mishra and Chaturvedi 2009). One can extract from the enumerated MPS, between a pair of nodes, its *foremost*, *shortest*, and the *fastest* MPS to compute *earliest arrival date*, the *minimum number of hops*, and the *minimum delay* (time span), respectively. The enumerated MPS can also be used to compute the Expected Hop Count (EHC) between a (s, d) node pair (Soh et al. 2007). On the other hand, the MCS, whose number is usually less in comparison to the number of MPS in a complex static network, are directly related to the modes of system failure or weak links in the design. Consequently, MCS are preferred over MPS for reliability evaluation, as they can help in direct identification of the distinct and discrete ways in which a system may fail (Billinton and Allan 1992). Further, the MCS concept has been used and extended to compute the maximum flow (Clark and Holton 2005), and performability metrics like capacity related reliability (Soh and Rai 2005) in flow networks.

The techniques utilizing MPS/MCS works in two-steps, *viz.*, enumeration of all MPS or MCS between specified set of nodes of the network, and efficiently combining MPS or MCS (logically using the *laws of probability*) thereafter to obtain a compact form of reliability expression or its estimate. The well-known SDP approach is a morphed and compact version of applying probability *law of unionization of events* or *inclusion-*

exclusion principle on events. There are two versions of SDP technique *viz.*, *Single-Variable Inversion* (SVI) and *Multi-Variable Inversion* (MVI) (Chaturvedi 2016). The SDP-MVI is preferred as it inverts a group of variables rather than a single one at a time, and thus results in a more compact reliability expression as compared to the expression rendered by SVI technique.

2.6.3 Reliability Evaluation – TVCNs

Even though there exists a plethora of methods for assessing the reliability of static networks, very few researchers have attempted to extend the notions from static networks to TVCNs. In the last decade, connectivity, routing protocols, mobility models, energy conservation, security issues and various applications, *etc*., of TVCNs have widely been studied for analyzing their performance. However, their reliability assessment has not been carried out explicitly and possibly because the existing approaches for reliability evaluation of static networks cannot be directly applied to the TVCNs owing to the associated challenges such as frequently changing topology and huge variations in environmental conditions.

The reliability of a TVCN can measure how reliably a packet sent from a source node be delivered to specified destination node(s) within a specified time under defined variable topological conditions. In this regard, recently some concerted efforts have been made by the researchers to model and evaluate the reliability of TVCNs. A MANET is a TVCN in which each node maintains a *routing* table. Using information in the routing table, a source node sends data to a destination node via some other nodes acting as routers (Kawamoto et al. 2013), thereby ensuring *end-to-end* connectivity/communication. References (Abolhasan et al. 2004; Alotaibi and Mukherjee 2012) review various routing schemes developed for MANETs. However, in harsh environmental conditions it is difficult to maintain accurate routing tables owing to the frequent disconnection of *end-to-end* path(s). More specifically, the reliability of MANETs, with end-to-end connectivity, decreases in case of low node density and/or high mobility. The reason is because low density and/or high moving speed of the nodes make it difficult to construct appropriate routing tables

(Kawamoto et al. 2013). In (Cook and Ramirez-Marquez 2008), authors estimated the reliability of MANETs with the help of Monte Carlo simulation technique. Later, their work was extended in (Chaturvedi and Padmavathy 2013) by modifying MANET by making specified set of nodes, ($e.g.$, v_s, v_d) with perfect nodes as their failure will definitely lead to network failure. In (Padmavathy and Chaturvedi 2013), authors utilized FS-TRG propagation model to depict link formation, thus, resulting into further modification of the approach proposed in (Cook and Ramirez-Marquez 2008). Authors in (Padmavathy and Chaturvedi 2015) evaluated the reliability of capacitated MANETs with the help of Log-Normal Shadowing propagation model. Besides, authors in (Soh et al. 2007) utilized SDP to compute reliability and EHC in wireless communication networks, but did not consider the topologies varying with time. In (Migov and Shakhov 2014), authors utilized factoring method to evaluate the reliability of MANETs. Authors in (Rebaiaia and Ait-Kadi 2015) proposed polygon-to-chain and series-parallel reduction based approach for evaluating the reliability of MANETs. In (Ahmad and Mishra 2012), authors proposed a critical node detection based approach for the reliability evaluation of large scale MANETs. In (Singh et al. 2014), authors utilized logistic regression based modelling followed by simulation in ns-2.35 simulation software to evaluate the reliability of MANETs. In (Kharbash and Wang 2007), authors modified the reliability evaluation algorithm for computer networks proposed in (Rai et al. 1986) to evaluate the reliability of MANETs with imperfect nodes. A stochastic link failure model was used in (Egeland and Engelstad 2009) for the reliability evaluation of wireless multi-hop networks. Recently, authors in (Panda and Dash 2017) evaluated the reliability of MANETs under link and node failure model. Besides, in (Meena and Vasanthi 2016a, b), authors proposed a method based on universal generating function for evaluating the reliability of MANETs. In (Eiza and Ni 2013), authors utilized an evolving graph model to find the most reliable journey in VANETs.

Despite the worth noting efforts for the reliability evaluation of MANETs, through *end-to-end* connectivity assessment and using Monte Carlo simulation, the researchers (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013) fail to consider the fact that there are many other TVCNs, which by design have sparse and intermittent connectivity, and mostly depend on *store-carry-and-forward* mechanism for data transfer

among nodes. As discussed in (Raffelsberger and Hellwagner 2014), device-to-device communication methodology increases network robustness in the presence of disruptions. Besides, the mechanism also decreases the impact of number of hops on data transmission (Matis et al. 2016). Moreover, by using device-to-device communication it is possible to find opportunistic routes over time for information exchange. In (Chen et al. 2016), authors utilized device-to-device communication to address topology control problem in network convergecasting applications. More specifically, the work in (Chen et al. 2016) uses space-time graph to model topology changes, and then presents three heuristic algorithms for constructing a sparse TVCN by enumerating tv-arborescence of minimum cost that also satisfies time delay constraint. Among the three polynomial time algorithms presented in (Chen et al. 2016), two are adaptations of well-known Kruskal's and Prim's algorithm (Clark and Holton 2005) used to find a minimum cost tree in static networks. Similarly, topology control problem in (Huang et al. 2013) maintains cost-efficient and connected TVCN topology for supporting data exchange between all pairs of nodes. In contrast, in (Kamiyama and Kawase 2015), authors considered arborescence packing problem in dynamic setting and presented an algorithm to find the desired number of edge-disjoint tv-arborescences in an acyclic temporal network. However, literature lack works that enumerate *all* diverging and/or converging tv-arborescences which can be utilized to evaluate the reliability of broadcasting and/or convergecasting in TVCNs as is evaluated in static networks using arborescences. Similarly, there is no work which considers enumeration of all TS-MPS/TS-MCS between a given (s, d) node pair to evaluate 2-terminal reliability of TVCNs via SDP approach. In this regard, authors in (Mertzios et al. 2019) presented algorithms for finding shortest TS-MPS in a temporal network. Similarly, (Xuan et al. 2003b) presented algorithms for finding shortest, fastest and foremost TS-MPS in TVCNs. Authors in (Coll-Perales et al. 2016) investigated the utility of opportunistic store-carry-and-forward mechanism to conserve energy in Multi-hop Cellular Networks, which integrate both cellular networks and device-to-device communication. In (Liang and Modiano 2017), authors presented a new survivability framework for TVCNs and extended the concept of maxFlow and minCut of static networks to TVCNs. They also showed that Menger's theorem only conditionally holds in TVCNs. Taking inspiration from the recent

developments and gaps in the literature, the works presented in this thesis are efforts to: i) efficiently incorporate s*tore-carry-and-forward* mechanism of data exchange and management of TVCN topology, ii) enumerate all TS-MPS/TS-MCS between a given (s, d) node pair, iii) enumerate all tv-arborescences, and iv) evaluate the reliability of both device-to-device and end-to-end connected TVCNs.

2.7 Summary

Firstly, this chapter has presented various modelling paradigms developed for studying TVCNs. The proposed definitions of Minimal Path Set, Minimal Cut Set and Arborescence equivalent models for TVCNs have also found a place in this chapter and have been discussed succinctly and pellucidly. It was followed by an exhaustive survey of the simulators, emulators and real world experiments developed to analyze the performance of TVCNs. Later part of the chapter highlighted some apparent differences between static networks and TVCNs, and presented their existing reliability evaluation approaches along with some research gaps.

Essentially, this chapter has provided the background and motivational aspects behind the development of algorithms and contributions to the research fraternity of this field. The ensuing chapters of this thesis present in detail the problems which were addressed during the course of this research work.

Chapter 3 Enumeration of Timestamped Minimal Path and Cut Sets

3.1 Introduction

For any arbitrary static network, connectivity (or dis-connectivity) and 2-terminal reliability (or unreliability) between a specified (s, d) pair of nodes can be computed with the help of the MPS (or MCS). Formally, an MPS is a path from v_s to v_d whereby no nodes are traversed more than once and every element of this set is needed for ensuring a successful communication. In contrast, an MCS is a minimal set of edges whose removal disconnects the (s, d) pair/graph. It is worth mentioning here that a number of algorithms exists in the literature for enumerating all MPS and MCS of a static network. On the same note, a similar notion for defining connectivity (or dis-connectivity) between two specified nodes and analyzing their 2-terminal reliability is equally important for TVCNs. However, due to the lack of continuous connectivity, network partitioning, and very long communication delays in TVCNs, this challenging task has not been addressed by the researchers till date. More specifically, literature lacks work which enumerate all MPS and MCS or identical terms for TVCNs by handling time-varying links, long delays, and dynamic topology.

This chapter addresses the aforementioned gaps in the literature. The chapter calls an MPS and MCS equivalent for TVCNs as TS-MPS and TS-MCS, respectively, and presents techniques for enumerating them. Recall that the formal definitions of the respective terms have been provided earlier in Chapter 2. Note that the work in this chapter has been published in (Chaturvedi et al. 2018), (Khanna et al. 2019b), and (Khanna et al. 2020b).

Now, before proceeding further with the chapter organization, we present the network model utilized in this chapter.

3.1.1 Network Model

We represent a TVCN by the TAG model. We also assume that the node set remains unchanged throughout the period τ of the TAG, and only edge set changes with time. Let V be the set of vertices and E is the set of edges (non-inclusive of timestamps) of TAG. Each vertex $v_i \in V$ denotes a mobile node while edge $e_{j,k} \in E$ represents a directed edge from node v_j to v_k, where both v_j, $v_k \in V$. Without loss of generality, we assume set E also contains edge $e_{k,j}$ for each edge $e_{j,k} \in E$. Let TS denote a $|V| \times |V|$ matrix that contains a sequence of *bidirectional contacts* or *timestamps* for each edge $e_{j,k} \in E$. Each entry $TS(j,k) \in TS$ stores the sequence of timestamps of edge $e_{j,k}$ in an ascending order. Thus, we use $G'(V, E^{TS})$ to represent a TAG, where each edge $e_{i,j}^{TS(i,j)} \in E^{TS}$ represents edge $e_{i,j} \in E$ that is associated with a sequence of timestamps $TS(i,j) \in TS$. We call an edge $e_{i,j}$ of the network appended with a single activity timestamp t, *i.e.*, $e_{i,j}^{t}$, as a TSE. Further, let $K_{|V|}$ denote a complete network graph of $|V|$ nodes.

The remaining chapter is organized as follows. Section 3.2 describes Cartesian Product based, Connection Matrix based and Line Graph based approaches for enumerating all TS-MPS. Section 3.3 presents two approaches for enumerating all TS-MCS. The first approach of TS-MCS enumeration inverts all TS-MPS using De Morgan's laws of Boolean algebra, while the second uses Line Graph. Section 3.4 summarizes the chapter.

3.2 TS-MPS Enumeration Techniques

This section presents three novel techniques for enumerating all TS-MPS between a specified (s, d) node pair of TVCNs.

3.2.1 Cartesian Product based Method

To obtain all TS-MPS between a specified (s, d) pair of nodes of a TVCN of $|V|$ nodes, the author in (Chaturvedi et al. 2018) proposed a method based on Cartesian product of timestamps of each edge associated with an MPS in the original complete network graph

$K_{|V|}$. The basis of this work is that any network graph formed subsequently in time, resulting into TAG $G'(V, E^{TS})$, would be a sub graph of the complete graph $K_{|V|}$. We assume that medium is homogenous, so the probability of edge existence would be same between any node pair. Besides, nodes are perfect – 100% reliable and are moderately mobile, such that time window (chosen in time unit) is sufficient enough to capture the changes occurring in the topology. We also assume that each bidirectional contact has an identical transmission power range. In addition, multi-hop transmission between nodes is possible within any time slot instantly. The details of the two-step approach are as follows.

i) Enumerate all (s, d) MPS for the complete static network graph, $K_{|V|}$.

ii) Enumerate all (s, d) TS-MPS for TAG $G'(V, E^{TS})$ from the MPS obtained in step i).

While Step i) is straightforward process for which several algorithms exist in the literature, Step ii) needs a careful consideration to capture the time based edge existence scenario and is one of the major contributions of this research.

We first explain aforementioned Steps i) and ii) through the following illustration.

Illustration 3.1: Let us utilize the TAG $G'(V, E^{TS})$ of five nodes as shown in Figure 2.6. Figure 3.1 is the complete graph (K_5) of Figure 2.6 at t = 0. Here, each bidirectional edge label $e_{i,j}$, without loss of generality, represents two directed edges $e_{i,j}$ and $e_{j,i}$. Assuming the source as node v_1 and destination as node v_5, the objective is to find the all TS-MPS between them.

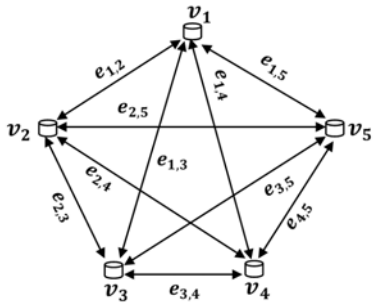


Figure 3.1. Complete graph of five nodes (K_5) representing underlying graph of the TAG shown in Figure 2.6.

In Step i), the MPS between (s, d) node pair (v_1, v_5) are generated for the underlying complete graph K_5 using existing algorithms, *e.g.*, (Chaturvedi and Misra 2002; Chaturvedi 2016). Table 3.1 provides all 16 edge-wise (and corresponding node-wise) MPS between (v_1, v_5).

Table 3.1. MPS between node pair (v_1, v_5) for the underlying complete graph (K_5) as shown in Figure 3.1 and MPS that exist between node pair (v_1, v_5) in the TAG of five nodes as shown in Figure 2.6.

Path No.	**All MPS between (v_1, v_5) for underlying K_5**		**Availability of timestamps to each edge in the corresponding path set**	**Edge-wise MPS that exist between (v_1, v_5) in TAG**
	Node-wise MPS	**Edge-wise MPS**		
1	v_1-v_5	$e_{1,5}$	Edge $e_{1,5}$ was absent throughout.	--
2	v_1-v_2-v_5	$e_{1,2}.e_{2,5}$	{1, 2}-{3}	$e_{1,2}.e_{2,5}$
3	v_1-v_3-v_5	$e_{1,3}.e_{3,5}$	Edge $e_{1,3}$ was absent throughout.	--
4	v_1-v_4-v_5	$e_{1,4}.e_{4,5}$	Edge $e_{4,5}$ was absent throughout.	--
5	v_1-v_2-v_3-v_5	$e_{1,2}.e_{2,3}.e_{3,5}$	{1, 2}-{2, 3, 4}-{1, 3, 4}	$e_{1,2}.e_{2,3}.e_{3,5}$
6	v_1-v_2-v_4-v_5	$e_{1,2}.e_{2,4}.e_{4,5}$	Edge $e_{4,5}$ was absent throughout.	--
7	v_1-v_3-v_2-v_5	$e_{1,3}.e_{3,2}.e_{2,5}$	Edge $e_{1,3}$ was absent throughout.	--
8	v_1-v_3-v_4-v_5	$e_{1,3}.e_{3,4}.e_{4,5}$	Edges $e_{1,3}$ and $e_{4,5}$ were absent throughout.	--
9	v_1-v_4-v_2-v_5	$e_{1,4}.e_{4,2}.e_{2,5}$	{3, 4}-{1, 2, 4}-{3}	$e_{1,4}.e_{4,2}.e_{2,5}$
10	v_1-v_4-v_3-v_5	$e_{1,4}.e_{4,3}.e_{3,5}$	{3, 4}-{2}-{1, 3, 4}	$e_{1,4}.e_{4,3}.e_{3,5}$
11	v_1-v_2-v_3-v_4-v_5	$e_{1,2}.e_{2,3}.e_{3,4}.e_{4,5}$	Edge $e_{4,5}$ was absent throughout.	--
12	v_1-v_2-v_4-v_3-v_5	$e_{1,2}.e_{2,4}.e_{4,3}.e_{3,5}$	{1, 2}-{1, 2, 4}-{2}-{1, 3, 4}	$e_{1,2}.e_{2,4}.e_{4,3}.e_{3,5}$
13	v_1-v_3-v_2-v_4-v_5	$e_{1,3}.e_{3,2}.e_{2,4}.e_{4,5}$	Edges $e_{1,3}$ and $e_{4,5}$ were absent throughout.	--
14	v_1-v_3-v_4-v_2-v_5	$e_{1,3}.e_{3,4}.e_{4,2}.e_{2,5}$	Edge $e_{1,3}$ was absent throughout.	--
15	v_1-v_4-v_2-v_3-v_5	$e_{1,4}.e_{4,2}.e_{2,3}.e_{3,5}$	{3, 4}-{1, 2, 4}-{2, 3, 4}-{1, 3, 4}	$e_{1,4}.e_{4,2}.e_{2,3}.e_{3,5}$
16	v_1-v_4-v_3-v_2-v_5	$e_{1,4}.e_{4,3}.e_{3,2}.e_{2,5}$	{3, 4}-{2}-{2, 3, 4}-{3}	$e_{1,4}.e_{4,3}.e_{3,2}.e_{2,5}$

-- Inactive MPS

Note that each edge-wise (corresponding node-wise) path between (s, d) node pair is a minimal set of edges (corresponding nodes) required to establish connection between nodes v_s and v_d. Observe from Table 3.1 that many MPS comprise of edges which are never active in TAG of Figure 2.6, thereby leads to inactive MPS. For example, edge, also single-hop MPS, $e_{1,5}$ does not exist in the TAG at any time slot. So, we can discard this inactive MPS and all others which utilize edge $e_{1,5}$. Similarly, all other MPS which comprise of edges non-existent in TAG are discarded as they lead to inactive MPS. Last column of Table 3.1 provides a list of path sets that exist in the TAG between node pair (v_1, v_5) without consideration of timestamps. Now, as we have obtained all seven MPS that can exist in TAG between node pair (v_1, v_5) during the four instants of time, in Step ii) we need to check their ability to carry the successful transmission of a packet, *i.e.*, their validity with respect to time-order of edges. This is necessary as the timestamp depicting an edge presence/activity appended with each edge of the paths may render a particular path set invalid or may provide some new paths sets. Thus, now to enumerate all TS-MPS, we must generate TSEs by generating combinations of timestamps, taking one timestamp from each edge of an MPS. For example, the MPS #9: $(e_{1,4}.e_{4,2}.e_{2,5})$ has timestamps of its edges as $TS(1,4)$ = {3, 4}, $TS(4,2)$ = {1, 2, 4} and $TS(2,5)$ = {3}, respectively, indicating the presence of its constituting edges. The six (= 2 × 3 × 1) possible combinations of timestamps for this MPS are {3-1-3}, {3-2-3}, {3-4-3}, {4-1-3}, {4-2-3} and {4-4-3}. The corresponding timestamped path sets, utilizing TSEs, are: $(e^3_{1,4}.e^1_{4,2}.e^3_{2,5})$, $(e^3_{1,4}.e^2_{4,2}.e^3_{2,5})$, $(e^3_{1,4}.e^4_{4,2}.e^3_{2,5})$, $(e^4_{1,4}.e^1_{4,2}.e^3_{2,5})$, $(e^4_{1,4}.e^2_{4,2}.e^3_{2,5})$, and $(e^4_{1,4}.e^4_{4,2}.e^3_{2,5})$. Note that in $(e^3_{1,4}.e^1_{4,2}.e^3_{2,5})$, TSE $e^1_{4,2}$ is present at timestamp 1 and before the appearance of TSE $e^3_{1,4}$, therefore, cannot establish communication between the (s, d) pair. Thus $(e^3_{1,4}.e^1_{4,2}.e^3_{2,5})$ is an invalid timestamped path set. Similarly, timestamped path set $(e^4_{1,4}.e^4_{4,2}.e^3_{2,5})$ is invalid because TSE $e^3_{2,5}$ does not appear at instant 4 to pass the data at the destination node v_5. Thus, none of the above timestamped path sets is able to connect (s, d) node pair (v_1, v_5). We capture the above aspect by simply employing the Cartesian product of the timestamps to generate all combinations of timestamps, thereby timestamped path sets. All invalid combinations with respect to the order of time are discarded. The generated TS-MPS based on this observation are presented in Table 3.2.

Table 3.2. All TS-MPS between node pair (v_1, v_5) in the TAG of Figure 2.6.

Sl. No.	Edge-wise path sets that exist between node pair (v_1, v_5)	Valid combinations of timestamps of available edges	Valid path sets generated based on the edges' timestamps
1	$e_{1,2}.e_{2,5}$	{1}-{3}	$e^1_{1,2}.e^3_{2,5}$
		{2}-{3}	$e^2_{1,2}.e^3_{2,5}$
2	$e_{1,2}.e_{2,3}.e_{3,5}$	{1}-{2}-{3}	$e^1_{1,2}.e^2_{2,3}.e^3_{3,5}$
		{2}-{2}-{3}	$e^2_{1,2}.e^2_{2,3}.e^3_{3,5}$
		{1}-{3}-{3}	$e^1_{1,2}.e^3_{2,3}.e^3_{3,5}$
		{2}-{3}-{3}	$e^2_{1,2}.e^3_{2,3}.e^3_{3,5}$
		{1}-{2}-{4}	$e^1_{1,2}.e^2_{2,3}.e^4_{3,5}$
		{2}-{2}-{4}	$e^2_{1,2}.e^2_{2,3}.e^4_{3,5}$
		{1}-{3}-{4}	$e^1_{1,2}.e^3_{2,3}.e^4_{3,5}$
		{2}-{3}-{4}	$e^2_{1,2}.e^3_{2,3}.e^4_{3,5}$
		{1}-{4}-{4}	$e^1_{1,2}.e^4_{2,3}.e^4_{3,5}$
		{2}-{4}-{4}	$e^2_{1,2}.e^4_{2,3}.e^4_{3,5}$
3	$e_{1,4}.e_{4,2}.e_{2,5}$	--	Generates no TS-MPS
4	$e_{1,4}.e_{4,3}.e_{3,5}$	--	Generates no TS-MPS
5	$e_{1,2}.e_{2,4}.e_{4,3}.e_{3,5}$	{1}-{1}-{2}-{3}	$e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^3_{3,5}$
		{1}-{2}-{2}-{3}	$e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$
		{2}-{2}-{2}-{3}	$e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$
		{1}-{1}-{2}-{4}	$e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^4_{3,5}$
		{1}-{2}-{2}-{4}	$e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5}$
		{2}-{2}-{2}-{4}	$e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5}$
6	$e_{1,4}.e_{4,2}.e_{2,3}.e_{3,5}$	{3}-{4}-{4}-{4}	$e^3_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}$
		{4}-{4}-{4}-{4}	$e^4_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}$
7	$e_{1,4}.e_{4,2}.e_{2,3}.e_{3,5}$	--	Generates no TS-MPS

-- No valid combination obtained via Cartesian product

Notice that the superscript used with each edge encountered in a TS-MPS indicates the respective edge activation time based on proper time ordering such that no past journey is encountered. The enumerated TS-MPS in Table 3.2 indicate that multiple nodes in a (s, d) path set can relay a data packet at the same time instant. From Table 3.2, we can see, *e.g.*, MPS #1, ($e_{1,2}.e_{2,5}$) can be activated for sending data at time t_1-t_3 or t_2-t_3 as per its edge existence; so it represents a single path set ($e_{1,2}.e_{2,5}$) rendering two TS-MPS: ($e^1_{1,2}.e^3_{2,5}$) and ($e^2_{1,2}.e^3_{2,5}$), respectively. The approach can also be modified to consider delay, *i.e.*, only a single-hop transmission in a time slot. In this representation, hidden timestamp due to buffering at any node is not being displayed, *e.g.*, in ($e^2_{1,2}.e^3_{2,5}$) we can see that edge $e_{1,2}$ is

transmitting data at second timestamp, however, edge was available between time slots {1, 2} implying that at timestamp 1, node v_1 kept the data buffered with itself. As per our assumption, nodes are perfect; hence, this representation can sufficiently represent TS-MPS without mentioning about hidden timestamps related with buffering. The validity of our proposition can be easily checked by visual inspection. Next, using the discussed approach, we enumerate the TS-MPS between (s, d) pair (v_5, v_1); see Table 3.3 and Table 3.4.

Table 3.3. MPS between node pair (v_5, v_1) for the underlying complete graph (K_5) as shown in Figure 3.1 and MPS that exist between node pair (v_5, v_1) in the TAG of five nodes as shown in Figure 2.6.

Path No.	**All MPS between (v_5, v_1) for underlying K_5**		**Availability of timestamps to each edge in the corresponding path set**	**Edge-wise MPS that exist between (v_5, v_1) in TAG**
	Node-wise MPS	**Edge-wise MPS**		
1	v_5-v_1	$e_{5,1}$	Edge $e_{5,1}$ was absent throughout.	--
2	v_5-v_2-v_1	$e_{5,2}.e_{2,1}$	{3}-{1, 2}	$e_{5,2}.e_{2,1}$
3	v_5-v_3-v_1	$e_{5,3}.e_{3,1}$	Edge $e_{3,1}$ was absent throughout.	--
4	v_5-v_4-v_1	$e_{5,4}.e_{4,1}$	Edge $e_{5,4}$ was absent throughout.	--
5	v_5-v_3-v_2-v_1	$e_{5,3}.e_{3,2}.e_{2,1}$	{1, 3, 4}-{2, 3, 4}-{1, 2}	$e_{5,3}.e_{3,2}.e_{2,1}$
6	v_5-v_4-v_2-v_1	$e_{5,4}.e_{4,2}.e_{2,1}$	Edge $e_{5,4}$ was absent throughout.	--
7	v_5-v_2-v_3-v_1	$e_{5,2}.e_{2,3}.e_{3,1}$	Edge $e_{3,1}$ was absent throughout.	--
8	v_5-v_4-v_3-v_1	$e_{5,4}.e_{4,3}.e_{3,1}$	Edges $e_{5,4}$ and $e_{3,1}$ were absent throughout.	--
9	v_5-v_2-v_4-v_1	$e_{5,2}.e_{2,4}.e_{4,1}$	{3}-{1, 2, 4}-{3, 4}	$e_{5,2}.e_{2,4}.e_{4,1}$
10	v_5-v_3-v_4-v_1	$e_{5,3}.e_{3,4}.e_{4,1}$	{1, 3, 4}-{2}-{3, 4}	$e_{5,3}.e_{3,4}.e_{4,1}$
11	v_5-v_4-v_3-v_2-v_1	$e_{5,4}.e_{4,3}.e_{3,2}.e_{2,1}$	Edge $e_{5,4}$ was absent throughout.	--
12	v_5-v_3-v_4-v_2-v_1	$e_{5,3}.e_{3,4}.e_{4,2}.e_{2,1}$	{1, 3, 4}-{2}-{1, 2, 4}-{1, 2}	$e_{5,3}.e_{3,4}.e_{4,2}.e_{2,1}$
13	v_5-v_4-v_2-v_3-v_1	$e_{5,4}.e_{4,2}.e_{2,3}.e_{3,1}$	Edges $e_{3,1}$ and $e_{5,4}$ were absent throughout.	--
14	v_5-v_2-v_4-v_3-v_1	$e_{5,2}.e_{2,4}.e_{4,3}.e_{3,1}$	Edge $e_{3,1}$ was absent throughout.	--
15	v_5-v_3-v_2-v_4-v_1	$e_{5,3}.e_{3,2}.e_{2,4}.e_{4,1}$	{1, 3, 4}-{2, 3, 4}-{1, 2, 4}-{3, 4}	$e_{5,3}.e_{3,2}.e_{2,4}.e_{4,1}$
16	v_5-v_2-v_3-v_4-v_1	$e_{5,2}.e_{2,3}.e_{3,4}.e_{4,1}$	{3}-{2, 3, 4}-{2}-{3, 4}	$e_{5,2}.e_{2,3}.e_{3,4}.e_{4,1}$

-- Inactive MPS

Table 3.4. All TS-MPS between node pair (v_5, v_1) in the TAG of Figure 2.6.

Sl. No.	**Edge-wise path sets that exist between node pair (v_5, v_1)**	**Valid combinations of timestamps of available edges**	**Valid path sets generated based on edges' timestamps**
1	$e_{5,2}.e_{2,1}$	--	Generates no TS-MPS
2	$e_{5,3}.e_{3,2}.e_{2,1}$	{1}-{2}-{2}	$e^1_{5,3}.e^2_{3,2}.e^2_{2,1}$
3	$e_{5,2}.e_{2,4}.e_{4,1}$	{3}-{4}-{4}	$e^3_{5,2}.e^4_{2,4}.e^4_{4,1}$
4	$e_{5,3}.e_{3,4}.e_{4,1}$	{1}-{2}-{3}	$e^1_{5,3}.e^2_{3,4}.e^3_{4,1}$
		{1}-{2}-{4}	$e^1_{5,3}.e^2_{3,4}.e^4_{4,1}$
5	$e_{5,3}.e_{3,4}.e_{4,2}.e_{2,1}$	{1}-{2}-{2}-{2}	$e^1_{5,3}.e^2_{3,4}.e^2_{4,2}.e^2_{2,1}$
6	$e_{5,3}.e_{3,2}.e_{2,4}.e_{4,1}$	{1}-{2}-{2}-{3}	$e^1_{5,3}.e^2_{3,2}.e^2_{2,4}.e^3_{4,1}$
		{1}-{2}-{2}-{4}	$e^1_{5,3}.e^2_{3,2}.e^2_{2,4}.e^4_{4,1}$
		{1}-{2}-{4}-{4}	$e^1_{5,3}.e^2_{3,2}.e^4_{2,4}.e^4_{4,1}$
		{1}-{3}-{4}-{4}	$e^1_{5,3}.e^3_{3,2}.e^4_{2,4}.e^4_{4,1}$
		{1}-{4}-{4}-{4}	$e^1_{5,3}.e^4_{3,2}.e^4_{2,4}.e^4_{4,1}$
		{3}-{3}-{4}-{4}	$e^3_{5,3}.e^3_{3,2}.e^4_{2,4}.e^4_{4,1}$
		{3}-{4}-{4}-{4}	$e^3_{5,3}.e^4_{3,2}.e^4_{2,4}.e^4_{4,1}$
		{4}-{4}-{4}-{4}	$e^4_{5,3}.e^4_{3,2}.e^4_{2,4}.e^4_{4,1}$
7	$e_{5,2}.e_{2,3}.e_{3,4}.e_{4,1}$	--	Generates no TS-MPS

Observe that the number of TS-MPS between (v_1, v_5) is 20 and is not equal to the number of TS-MPS between (v_5, v_1), which is equal to 13. It happens because of the directivity induced by the timestamps over edges.

Hence, it can be emphasized here that unlike in static networks, in TVCNs the number of TS-MPS between (s, d) may or may not be equal to the number of TS-MPS between (d, s).

The steps in the illustration are generalized and a flowchart outlining the procedure is shown in Figure 3.2.

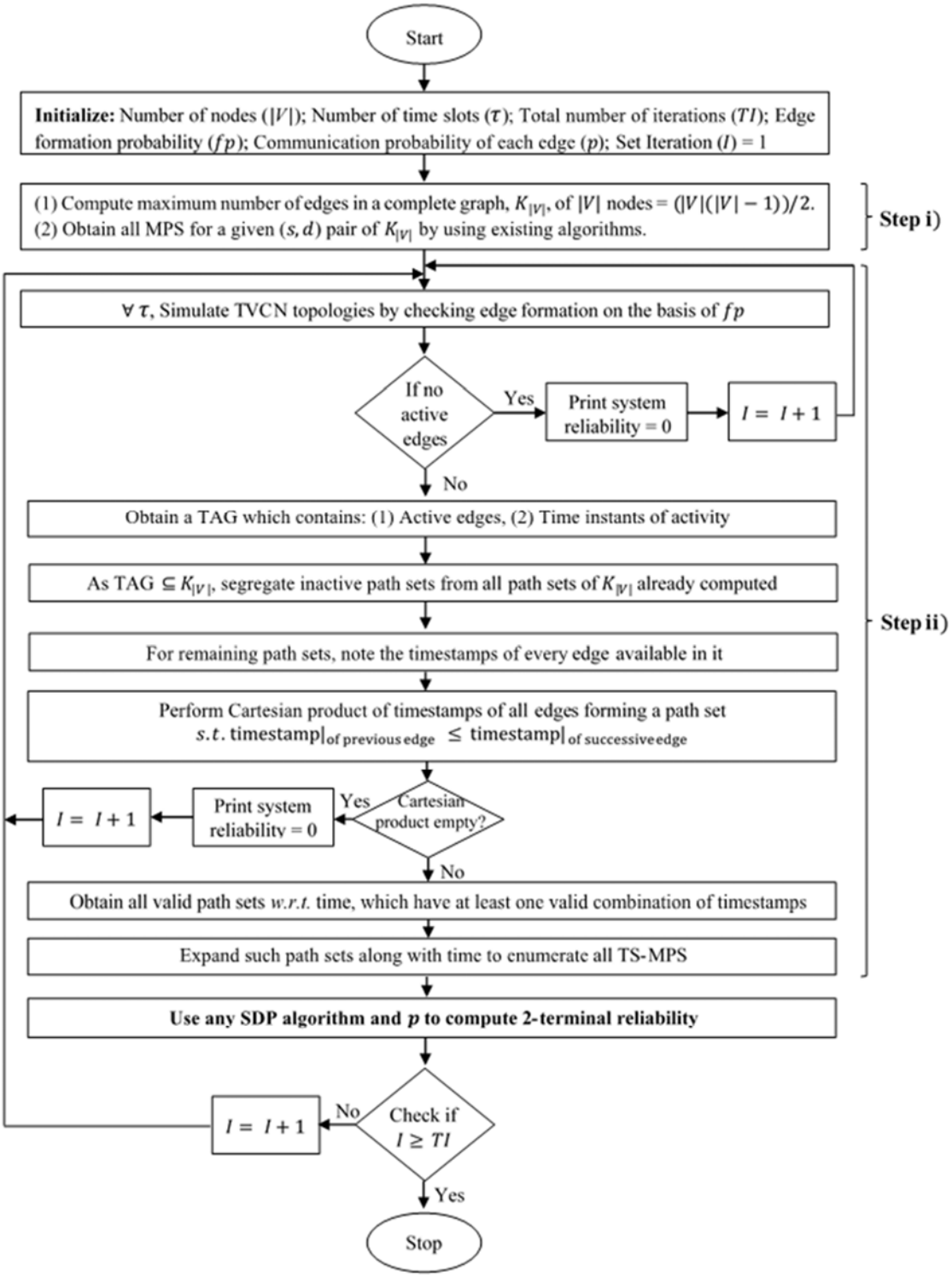


Figure 3.2. Flow chart for TS-MPS enumeration and 2-terminal reliability evaluation of TVCNs using Cartesian product based approach.

In the initialization step of our proposed approach, we consider two probabilities, *i.e.*:

i) *Probability of an edge formation* (fp), which defines the probability of existence of an edge between each pair of nodes. This probability can be linked to an appropriate fading scenario.
ii) *Communication probability of an edge* (p), which defines an edge's capability to successfully transmit the data packets. In other words, it defines the transmission reliability of each edge.

We have used two probabilities because, generally, in practical disruptive environments the edges are not formed deterministically, but are formed in a probabilistic manner. The aforementioned probabilities can be figured out from the field data or past experiences of similar scenarios. Note that the latter probability is used for determining 2-terminal reliability of TVCN - discussed later in Chapter 5, and is shown with bold face in Figure 3.2. In Step i), for a $|V|$ node TVCN, we enumerate all (s, d) MPS of $K_{|V|}$ using existing approaches, *e.g.*, (Chaturvedi 2016). Recall that the main idea behind the proposed algorithm is that as any mobile network at an observation instant would be a sub-graph of complete graph of $|V|$ nodes, therefore, all the MPS are obtained for a complete graph of $|V|$ nodes ($K_{|V|}$), *e.g.*, Table 3.1, Column 2, for a complete graph of five nodes. In Step ii), for all the edges of the complete graph, *i.e.*, $\frac{|V|(|V|-1)}{2}$, we check whether an edge between a pair of nodes forms within time instants of observations. This condition is defined on the basis of edge formation probability, fp. Once the active edges are identified, their respective time instants of activity are noted. This is done to generate an arbitrary TAG similar to one shown in Figure 2.6. Based on the active edges, path sets of $K_{|V|}$ are screened and path sets consisting of inactive edges are discarded. For the remaining MPS, we find Cartesian product of timestamps in the order of their edges constituting each MPS. All non-time-ordered sequences of Cartesian product are discarded as they will not render any TS-MPS. At last, expanding the remaining MPS with all corresponding valid Cartesian products will lead to all TS-MPS.

3.2.1.1 Time Complexity

The total number of edges in a complete network graph of $|V|$ nodes is $\frac{|V|(|V|-1)}{2}$ and since there are τ time slots, the activity status of each edge in all time slots can be determined in $O(|V|^2\,\tau)$. The number of an MPS between a (s, d) node pair in a complete graph $K_{|V|}$ is given by Equation (3.1) (Cook and Ramirez-Marquez 2007).

$$|\text{MPS}| = \sum_{i=0}^{|V|-2} \frac{(|V|-2)!}{(|V|-2-i)!} \tag{3.1}$$

Here, 'i' represents the number of intermediate nodes in a path set. So, the complexity in enumerating all possible MPS would be in the order of $|V|!$. Further, maximum cardinality of any MPS would be $(|V|-1)$. Now, if all the edges involved in the formation of MPS are active in all timestamps, then their Cartesian product will result in $\tau^{|V|-1}$ terms, each having length $(|V|-1)$. The elements within each term obtained after Cartesian product in worst case need $(|V|-2)$ comparisons for checking their validity in time. Hence, the maximum number of comparisons in this step would be $(|V|-2)\,\tau^{|V|-1}$. So, the complexity of this step is $O(|V|\,\tau^{|V|})$. Moreover, as the Cartesian product will be computed at maximum for all possible MPS, the overall time complexity of this step would be $O(|V|!\,(\tau^{|V|}))$. Hence, the worst case complexity of the proposed algorithm for each run is $O(|V|!\,(\tau^{|V|}))$.

3.2.1.2 Simulation Results

We have implemented the algorithm using MATLAB® version 2014a running on PC with the following configurations: Processor: Intel (R) Core (TM) i5-6500 CPU @ 3.20 GHz, RAM: 4.00 GB, System Type: 64-bit Operating System, x64-based processor. Further, to mimic dynamically changing scenarios of five and six node TVCNs, thereby generating time varying topologies by simulating existence/non-existence of edges during an observational period, we used Monte Carlo Simulation. In the Monte Carlo Simulation, we

varied a number of parameters, *viz.*, edge formation probability, number of time slots, number of iterations and edge communication probability to find the relative variation in the average number of generated TS-MPS and the overall average network reliability. As this chapter only deals with enumeration of TS-MPS, results pertaining to average network reliability, *i.e.*, 2-terminal reliability, have been shown later in Chapter 5.

Figure 3.3 provides the variation in the average number of TS-MPS relative to the edge formation probability. Here, Monte Carlo Simulation iterations were set to 10000, time slots to four time units and edge communication probability to 0.7. It is evident from the results that with an increase in the edge formation probability, the overall transmission opportunities increase or we can say environment becomes less disruptive. In such a case, the average number of TS-MPS show increasing trends.

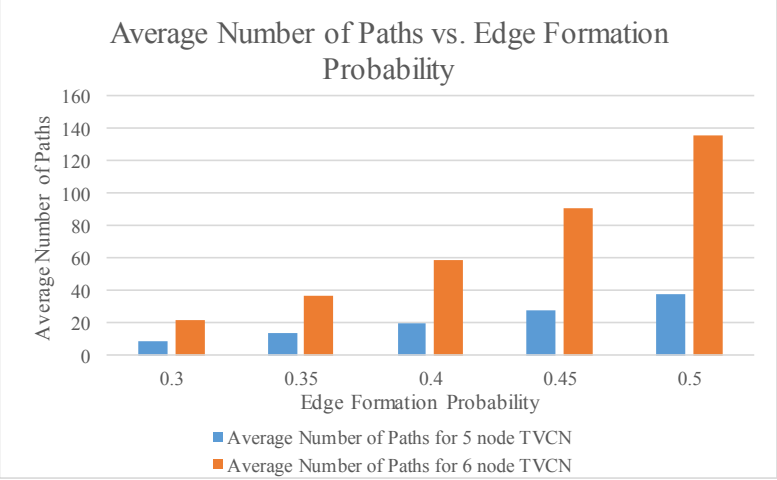


Figure 3.3. Average number of path sets vs. edge formation probability at 10000 iterations, 4 time slots and edge communication probability of 0.7.

Figure 3.4 describes the relationship between average number of TS-MPS and edge communication probability, when the number of iterations is fixed to 10000, TVCN period is fixed to four time units and edge formation probability is fixed as 0.3. Results show that as the edge communication probability increases, the average number of paths remains same for a fixed edge formation probability. This happens because the creation of an edge

is governed by edge formation probability alone. The observations from simulation validate the intuition.

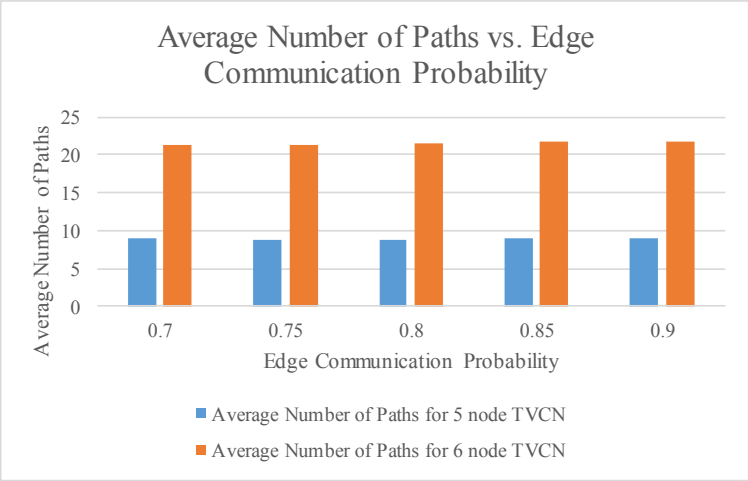


Figure 3.4. Average number of path sets vs. edge communication probability at 10000 iterations, 4 time slots and edge formation probability of 0.3.

Figure 3.5 describes the relationship between average number of path sets relative to the time slots, when the number of iterations is fixed to 10000, edge formation probability is fixed as 0.3 and edge communication probability is fixed as 0.7. It is clear that the number of TS-MPS keep on increasing with an increase in time slots. It is intuitive, as the number of time instants of observation increases the overall opportunities of data transmission will increase.

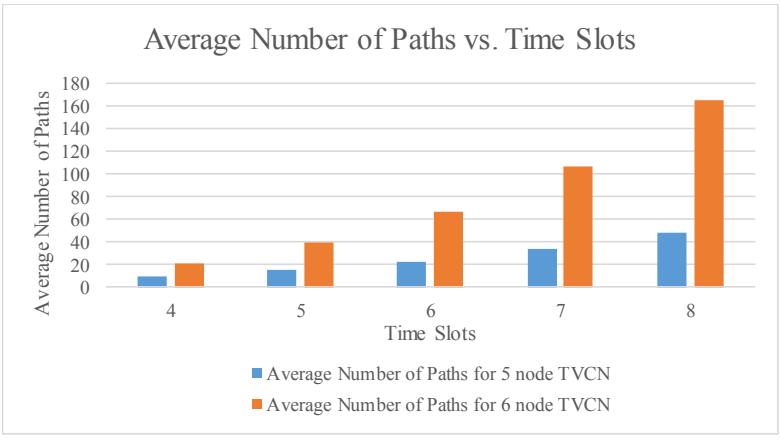


Figure 3.5. Average number of path sets vs. time slots at 10000 iterations, edge formation probability of 0.3 and edge communication probability of 0.7.

Figure 3.6 provides the variation in the average number of path sets relative to the number of iterations, when the time slots are fixed to four, edge formation probability is fixed to 0.5 and edge communication probability is fixed to 0.7.

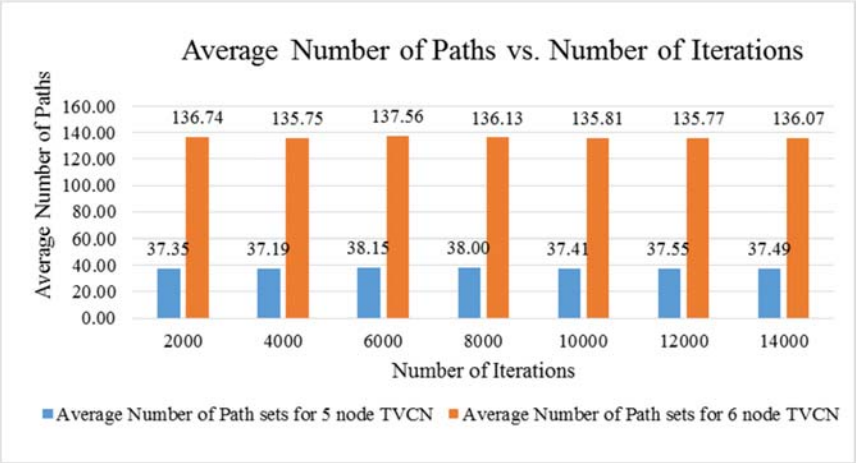


Figure 3.6. Average number of path sets vs. number of iterations at four time slots, edge formation probability of 0.5 and edge communication probability of 0.7.

The graph suggests that the average number of TS-MPS generated are almost the same in each iteration. Therefore, for the selected parameters, 4000-5000 iterations are enough to produce good results. However, in reality, the number of generated path sets are different in each iteration of Monte Carlo Simulation, but after averaging such path sets the result is almost the same in number. Hence, it is important to note from Figure 3.6, that the large number of iterations do not have any remarkable impact on simulation performance.

Limitation: It was later observed by the scholar that with the Cartesian product based approach if in some application one needed to enumerate all TS-MPS between each node pair at one go, then the two steps of the approach had to be repeated for each pair of nodes present in the network – an additional computational burden with associated costs.

The solution proposed and algorithm presented in the next section overcomes this limitation as it generates all TS-MPS between every (s, d) node pair of the TVCN in one go with less computational efforts.

3.2.2 Connection Matrix based Method

The matrix based approach presented in this section and proposed by the scholar in (Khanna et al. 2019b) is capable of enumerating all possible TS-MPS between every pair of nodes of a TAG $G'(V, E^{TS})$. The basic idea behind the development of this technique/algorithm is to exploit all the possible opportunities (in terms of edges and/or nodes), which evolve between different pair of nodes with time and lead to the forwarding of data packets towards a specified destination. This as a result leads to a successful data transmission between a (s, d) node pair via single/multiple hop(s) over distinct nodes in a single/multiple time slot(s) within a given observation duration.

Recall that the TAG $G'(V, E^{TS})$ shown in Figure 2.6 has 20 TS-MPS between (s, d) node pair (v_1, v_5). Similarly, it might have a comparable, larger or smaller number of TS-MPS between other (s, d) node pairs. Hence, for lucid explanation and illustration purpose, we reconsider the small and sparsely connected TAG shown in Figure 2.1(ii). Note here that the same basic assumptions have been followed as in Section 3.2.1. However, we alternatively represent each edge $e_{i,j}$ that is associated with a sequence of timestamps in TAG using a distinct edge number along with the sequence of timestamps. Thus, the edge number remains same in both forward and reverse directions. More specifically, the TAG of Figure 2.1(ii) is alternatively represented as Figure 3.7, wherein a bidirectional edge, say, $e_{1,2}^{\{1,3\}}$ of original TAG is represented as $e_1, \{1,3\}$. This depiction becomes necessary for properly explaining the devised Algorithm 3.1.

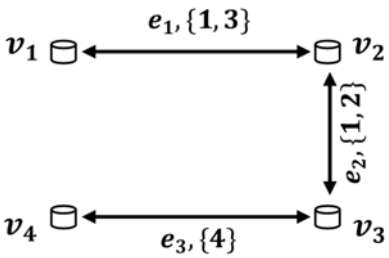


Figure 3.7. Alternative representation of TAG of Figure 2.1(ii) to explain Algorithm 3.1.

In the following, we describe the algorithmic steps of connection matrices based approach of TS-MPS enumeration. More specifically, we now explain each step of the Algorithm 3.1 in detail. In Step 1, this algorithm requires the connection matrix of the network topology at each instance of time, *i.e.*, it requires the connection details in each snapshot as input. Let us denote and initialize such connection matrices by $M_t^{(0)}\ \forall\ t \in [1, \tau]$. Here, $\tau = 4$ time units is the entire observation period, *i.e.*, t = 1, 2, …, 4. Each (i, j)-th entry of the matrix depicts an active edge in snapshot t connecting vertex i to another distinct vertex j. More specifically, entry (i, j) contains TSE e_x^y if edge e_x is active at time slot $y \in [1, \tau]$. If there is no path/connection from i to j, then its (i, j)-th entry is set to zero. The diagonal entries of $M_t^{(0)}$ are set as '1' to reflect the *buffering* (*store-carry-and-forward*) of information at the concerned nodes due to their possible isolation from other nodes or due to absence of opportunistic edges. The connection matrices, *viz.*, $M_1^{(0)},\ M_2^{(0)}, M_3^{(0)},$ and $M_4^{(0)}$ are as shown below.

Algorithm 3.1. Connection_Matrix_TS-MPS

Input: A TAG of $|V|$ nodes.

Output: All TS-MPS between every node pair.

Step 1. Generate the input connection matrices, ${M_t}^{(0)}\ \forall\ t \in [1, \tau]$.

Step 2. Process ${M_t}^{(0)}$ to obtain ${M_t}^{(|V|)}\ \forall\ t \in [1, \tau]$ by using the following procedure.

i) Set $t = 1$,

ii) Set $k = 1,\ S = 1$

iii) Vary $D = 1, 2, \ldots, |V|$. Obtain ${M_t}^{(k)}$ from ${M_t}^{(k-1)}$ whose elements are such that $-\ \boldsymbol{m_{SD}^{k} = m_{Sk}^{k-1}\ \&\ m_{kD}^{k-1}\ |\ m_{SD}^{k-1}}$. Note that we resort to Boolean multiplication and addition, denoted by '**&**' (ampersand) and '|' (vertical bar), respectively, to obtain such elements.

iv) Set $S = S + 1$. Repeat from (iii) while $S \leq |V|$.

v) Set $S = 1$. Set $k = k + 1$, repeat from (iii) while $k \leq |V|$.

vi) Set $t = t + 1$. Repeat from (ii) while $t \leq \tau$

Step 3. Store each ${M_t}^{(|V|)}$ obtained for each time instant t.

Step 4. Compute $MP = {M_1}^{(|V|)} \times {M_2}^{(|V|)} \times \ldots \times {M_\tau}^{(|V|)}$. Simplify its each entry using Boolean algebra laws and set the diagonal elements of MP to 1, (as diagonal elements with value 1 simply show self-connection/buffering).

Step 5. **TS-MPS** $= MP$

$$M_1^0 = \begin{bmatrix} 1 & e_1^1 & 0 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ 0 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M_2^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_3^{(0)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M_4^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$$

In Step 2, each ${M_t}^{(0)}$ is processed to obtain ${M_t}^{(|V|)}$ $\forall\ t \in [1, \tau]$, where $|V|$ is the number of mobile nodes in the network. This involves generation of intermediate matrices ${M_t}^{(k)}$ $\forall\ t \in [1, \tau]$ and $\forall\ k \in [1, |V|]$ whose elements are such that $\boldsymbol{m_{SD}^{k} = m_{Sk}^{k-1}\ \&\ m_{kD}^{k-1}\ |\ m_{SD}^{k-1}}$ obtained from the elements in ${M_t}^{(k-1)}$, where S, k and $D \in [1, |V|]$. Here, '**&**' (ampersand) denotes Boolean multiplication and '|' (vertical bar) denotes addition operation, respectively, to obtain such elements. Note that the Boolean operators used in Step 2 iii) are used to call upon the Boolean laws of idempotence and absorption to absorb duplicate and/or repetitive terms that appear during the aggregation of path sets. Here, Step 2 (i) - (iv) are provided in detail as follows to obtain $M_1^{(1)}$ from $M_1^{(0)}$ by applying $\boldsymbol{m_{SD}^{k} = m_{Sk}^{k-1}\ \&\ m_{kD}^{k-1}\ |\ m_{SD}^{k-1}}$:

for $\boldsymbol{k = 1, S = 1, D = 1,2,3,4}$
$m_{11}^1 = m_{11}^0\ \&\ m_{11}^0\ |\ m_{11}^0 = 1\ \&\ 1\ |\ 1 = 1$
$m_{12}^1 = m_{11}^0\ \&\ m_{12}^0\ |\ m_{12}^0 = 1\ \&\ e_1^1\ |\ e_1^1 = e_1^1$
$m_{13}^1 = m_{11}^0\ \&\ m_{13}^0\ |\ m_{13}^0 = 1\ \&\ 0\ |\ 0 = 0$
$m_{14}^1 = m_{11}^0\ \&\ m_{14}^0\ |\ m_{14}^0 = 1\ \&\ 0\ |\ 0 = 0$

for $\boldsymbol{k = 1, S = 2, D = 1,2,3,4}$
$m_{21}^1 = m_{21}^0\ \&\ m_{11}^0\ |\ m_{21}^0 = e_1^1\ \&\ 1\ |\ e_1^1 = e_1^1$
$m_{22}^1 = m_{21}^0\ \&\ m_{12}^0\ |\ m_{22}^0 = e_1^1\ \&\ e_1^1\ |\ 1 = 1$
$m_{23}^1 = m_{21}^0\ \&\ m_{13}^0\ |\ m_{23}^0 = e_1^1\ \&\ 0\ |\ e_2^1 = e_2^1$
$m_{24}^1 = m_{21}^0\ \&\ m_{14}^0\ |\ m_{24}^0 = e_1^1\ \&\ 0\ |\ 0 = 0$

for $\boldsymbol{k = 1, S = 3, D = 1,2,3,4}$
$m_{31}^1 = m_{31}^0\ \&\ m_{11}^0\ |\ m_{31}^0 = 0\ \&\ 1\ |\ 0 = 0$
$m_{32}^1 = m_{31}^0\ \&\ m_{12}^0\ |\ m_{32}^0 = 0\ \&\ e_1^1\ |\ e_2^1 = e_2^1$
$m_{33}^1 = m_{31}^0\ \&\ m_{13}^0\ |m_{33}^0 = 0\ \&\ 0\ |\ 1 = 1$
$m_{34}^1 = m_{31}^0\ \&\ m_{14}^0\ |m_{34}^0 = 0\ \&\ 0\ |\ 0 = 0$

for $\boldsymbol{k = 1, S = 4, D = 1,2,3,4}$
$m_{41}^1 = m_{41}^0\ \&\ m_{11}^0\ |\ m_{41}^0 = 0\ \&\ 1\ |\ 0 = 0$
$m_{42}^1 = m_{41}^0\ \&\ m_{12}^0\ |\ m_{42}^0 = 0\ \&\ e_1^1\ |\ 0 = 0$
$m_{43}^1 = m_{41}^0\ \&\ m_{13}^0\ |\ m_{43}^0 = 0\ \&\ 0\ |\ 0 = 0$
$m_{44}^1 = m_{41}^0\ \&\ m_{14}^0\ |\ m_{44}^0 = 0\ \&\ 0\ |\ 1 = 1$

Therefore, $M_1^{(1)} = \begin{bmatrix} 1 & e_1^1 & 0 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ 0 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. Following the above procedure on $M_1^{(1)}$ to obtain $M_1^{(2)}$, from $M_1^{(2)}$ to obtain $M_1^{(3)}$ and finally, from $M_1^{(3)}$ to obtain $M_1^{(4)}$, Step 2 produces the matrices below.

$$M_1^{(2)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_1^{(3)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and}$$

$$M_1^{(4)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Table 3.5 shows all input connection matrices and the intermediate matrices generated at various stages of computation. Thus, it also shows each $M_t^{(|V|)}$ obtained for each time instant t.

Table 3.5. Intermediate Matrices generation from connection matrices.

Time slot	Connection Matrix	Intermediate Matrices
$t = 1$	$M_1^0 = \begin{bmatrix} 1 & e_1^1 & 0 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ 0 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$M_1^{(1)} = \begin{bmatrix} 1 & e_1^1 & 0 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ 0 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$M_1^{(2)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$M_1^{(3)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$\boldsymbol{M_1^{(4)} = \begin{bmatrix} 1 & e_1^1 & e_1^1.e_2^1 & 0 \\ e_1^1 & 1 & e_2^1 & 0 \\ e_1^1.e_2^1 & e_2^1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$
$t = 2$	$M_2^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$M_2^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Time slot	Connection Matrix	Intermediate Matrices
		$M_2^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$M_2^{(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$\boldsymbol{M_2^{(4)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & e_2^2 & 0 \\ 0 & e_2^2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$
$t = 3$	$M_3^{(0)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$M_3^{(1)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$M_3^{(2)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$M_3^{(3)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		$\boldsymbol{M_3^{(4)} = \begin{bmatrix} 1 & e_1^3 & 0 & 0 \\ e_1^3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$
$t = 4$	$M_4^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$	$M_4^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$
		$M_4^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$
		$M_4^{(3)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$
		$\boldsymbol{M_4^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}}$

After obtaining matrices ${M_1}^{(|V|)}, {M_2}^{(|V|)}, \ldots, {M_\tau}^{(|V|)}$, respectively, Step 4 enumerates all TS-MPS by multiplying the matrices in order. Further, it also simplifies each entry of the resulting matrix by using the laws of Boolean algebra. At last, it sets the diagonal elements of the resulting matrix to 1 (as diagonal elements with value 1 simply show self-connection/buffering). Step 5 stores this result in **TS-MPS**, *i.e.*,

$$\textbf{TS-MPS} = \begin{bmatrix} 1 & e_1^1 + e_1^3 & e_1^1.e_2^1 + e_1^1.e_2^2 & e_1^1.e_2^1.e_3^4 + e_1^1.e_2^2.e_3^4 \\ e_1^1 + e_1^3 & 1 & e_2^1 + e_2^2 & e_2^1.e_3^4 + e_2^2.e_3^4 \\ e_2^1.e_1^1 + e_2^1.e_1^3 + e_2^2.e_1^3 & e_2^1 + e_2^2 & 1 & e_3^4 \\ 0 & 0 & e_3^4 & 1 \end{bmatrix}$$

For instance, if one intends to obtain all TS-MPS between node pair (1, 4), then there are two TS-MPS available at **TS-MPS**$(\mathbf{1}, \mathbf{4})$, *viz.*, $(e_1^1.e_2^1.e_3^4)$ and $(e_1^1.e_2^2.e_3^4)$. Likewise, for node pair (2, 4), one can look at the matrix cell entries at cell (2, 4) of the **TS-MPS** matrix and so on.

Further, if we intend to consider nodes explicitly in the path sets along with TSEs, then the same steps can be repeated with slightly modified connection matrices for the network. Specifically, for this case, the modified connection matrices would append starting and ending node along with each TSE; rest of the process will remain same. Thus, this method can also be used to enumerate TS-MPS after relaxing the assumption of perfect nodes! *i.e.*, nodes with reliability less than 100%.

It is important to note the following:

- Barring the notation, this approach produces the same results as by the Cartesian product based method, thereby validating the correctness of the two approaches.
- Step 4 used here is similar to the procedure utilized in graph theory wherein the adjacency matrix of the graph is multiplied by itself say k number of times to enumerate all the walks of length k. However, here it needs some further simplification using Boolean laws.

Limitation: The proposed connection matrix based method is effective for a TVCN with few nodes such as an LEO satellite network. The reason is that these networks are: i) not necessarily large enough to warrant the use of asymptotic analysis, and ii) sparse with often unreliable connections; therefore, reflect the inherent unreliability of wireless communications. However, if the number of nodes in such a network is increased manifold, then the matrix operations involving Boolean operators require large computation space and time.

3.2.2.1 Time Complexity

The time complexity of the connection matrix based algorithm is $O(|V|^3\,\tau)$, which is computed as follows. Step 1 takes $O(|V|^2\,\tau)$ since each matrix is of size $|V| \times |V|$ and there are τ such matrices. Step 2 iii) requires $O(|V|^3)$; repeating the step τ times, the time complexity of Step 2 is $O(|V|^3\,\tau)$. In Step 4, each matrix multiplication takes $O(|V|^3)$; with $(\tau - 1)$ multiplications, the step requires time $O(|V|^3\,\tau)$.

3.2.3 Line Graph based Method

The Line Graph based method presented in this section requires two steps for the enumeration of all TS-MPS between a specified (s, d) node pair of TAG $G'(V, E^{TS})$. The steps are: i) Transform the TAG $G'(V, E^{TS})$ of TVCN to an equivalent Line Graph, and ii) Enumerate all TS-MPS from the generated Line Graph (Khanna et al. 2020b). These steps need careful consideration to capture the *time based edge existence* scenario of such TVCNs. More specifically, Step i) transforms a TVCN into a static Line Graph wherein original reachability information is preserved. This technique of graph transformation has recently been utilized in (Liang and Modiano 2017) for analyzing survivability in time-varying networks. Section 3.2.3.1 presents an algorithm defining the steps for constructing a Line Graph. For Step ii), one can use an existing MPS enumeration algorithm, *e.g.*, (Chaturvedi and Misra 2002) to generate all TS-MPS from the generated Line Graph.

3.2.3.1 Generating Line Graph

A Line Graph generation begins by creation of two separate nodes for the source and destination node, respectively. This is followed by the creation of a node for each TSE $e^{t}_{x,y}$ in the TAG $G'(V, E^{TS})$. A newly created node, say $e^{t_i}_{x',y'}$, in Line Graph is connected with a directed edge to another node $e^{t_j}_{x'',y''}$ if and only if $t_i \leq t_j$. Next, a directed edge is added from the node created for the source to $e^{t_i}_{x',y'}$, if $e^{t_i}_{x',y'}$ originates from the source node; similarly a directed edge is added from node $e^{t_j}_{x'',y''}$ to a node created for the destination, if $e^{t_j}_{x'',y''}$ terminates at the destination. Such a transformation from TAG to Line Graph needs careful consideration to capture the *time based edge* (*i.e.*, TSE) *existence* scenario. Note that size of Line Graph is dependent on the number of nodes and edges in its TAG as well as the number of time slots. An example of the Line Graph is shown earlier in Figure 2.5. The algorithmic steps for generating a Line Graph are as below.

Algorithm 3.2. Gen_Line_Graph

Input: Matrix X of size $|V| \times |V|$ representing a TAG or in other words indicating the timestamps at which edges between nodes i and j are active, where $i, j \in [1, |V|]$ and $|V|$ is number of nodes.

Output: Line Graph between source and destination node, in which the nodes represent the TSEs of the TAG.

Step 1. Set all values in the column corresponding to Source node as '0'.

Step 2. Assign a unique numeric value (representing nodes) to the TSEs at which data can be transmitted from node i to j and store them in matrix Y of size $|V| \times |V|$.

Step 3. Using matrix Y, create a list of nodes which can possibly be connected in the next two hops from the current node, to avoid cycles in the Line Graph, by following the steps given below:

(i) Create a list $[Z]$ of node pair i and j, $\forall\ i \neq j, i \neq |V|$ and $Y[i, j] \neq \emptyset$.

(ii) $\forall\ (i, j)$ node pair obtained in Step 3(i), select and store in $[Z]$ all numeric value of the TSEs from i^{th} column leaving all those present in the j^{th} row.

For source node, the numeric value of the TSE is taken as 1. //To find the possible current node(s), *i.e.*, 'i' in the Line Graph.

(iii) Select and store in $[Z]$ the elements in the i^{th} row and j^{th} column for connecting them to all the elements obtained in Step 3(ii). //To find all first hop neighbour node(s), *i.e.*, 'j' in the Line Graph.

(iv) Select and store in $[Z]$ all elements in the j^{th} row leaving all those present in i^{th} column to keep track of second hop neighbors, *i.e.*, 'k' in the Line Graph.

Step 4. Create an empty adjacency matrix of size $R \times R$, where R is the maximum numeric value assigned to the TSEs plus one.

Step 5. For the elements obtained in Step 3(ii) and 3(iii), check their corresponding locations in matrix Y and their respective edge activity time in matrix X.

(i) If the timestamp of preceding edge, i is less than or equal to the timestamp of succeeding edge, j, then only there will be an actual edge formation between the two nodes.

(ii) For all such node pairs, insert '1' at i^{th} row and j^{th} column to fill the empty adjacency matrix created in Step 4. Also, find the edges which terminate at the target node; insert '1' at the rows corresponding to these edges and the last column to complete the creation of Line Graph.

Step 6. Prune the Line Graph to remove irrelevant nodes by using following steps.

(i) Delete all (except source) nodes for which there is no input by removing columns that contain all zero entries and their corresponding rows. Recursively repeat this step until there is no column with all zero entries.

(ii) Delete all (except destination) nodes for which there is no input by removing rows that contain all zero entries and their corresponding columns. Recursively repeat this step until there is no row with all zero entries.

We have implemented the above algorithm for generating Line Graph using MATLAB® version 2016a running on PC with the following configurations: Processor:

Intel (R) Core (TM) i5-6500 CPU @ 3.20 GHz, RAM: 4.00 GB, System Type: 64-bit Operating System, x64-based processor.

Using Algorithm 3.2, we obtain the Line Graph as shown in Figure 3.8 for the TAG shown in Figure 2.1(ii). Observe that barring the notation, the Line graph of Figure 3.8 is equivalent to the one shown in Figure 2.5. The only difference is that in our proposal each TSE, thereby node of the Line Graph, is represented by a distinct decimal number.

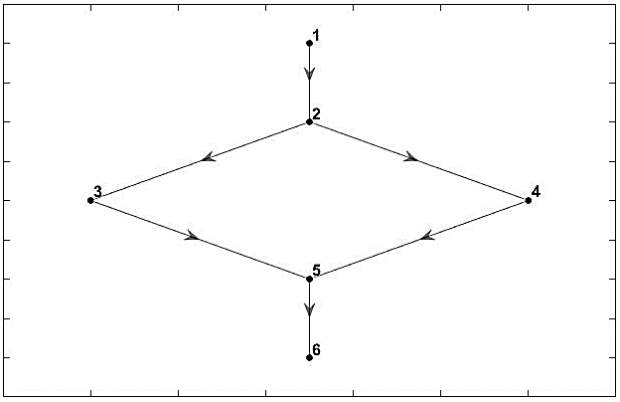


Figure 3.8. Line Graph of TAG shown in Figure 2.1(ii) obtained using Algorithm 3.2.

Notice that nodes 1 and 6 in Figure 3.8 correspond to nodes v_1 and node v_4 in Figure 2.5. Besides, it is important to observe from the Line Graph shown in Figure 2.5 that node v_1 and node v_4 have already been included in the TSEs $e_{1,2}^1$ and $e_{3,4}^4$, respectively, therefore, in the equivalent Line Graph of Figure 3.8, nodes 1 and 6 are redundant and can be deleted from all enumerated TS-MPS between (v_1, v_4), *viz.*, (1-2-3-5-6) and (1-2-4-5-6) obtained in Step ii) using an existing MPS enumeration algorithm, *e.g.*, (Chaturvedi and Misra 2002). The modified TS-MPS of the network under consideration are: (2-3-5) and (2-4-5). The two TS-MPS are equivalent to $(e_{v_1,v_2}^1.e_{v_2,v_3}^1.e_{v_3,v_4}^4)$ and $(e_{v_1,v_2}^1.e_{v_2,v_3}^2.e_{v_3,v_4}^4)$ except difference in the notation. *Note that the performance of this method is dependent on the efficiency of an MPS enumeration algorithm utilized in its Step ii).*

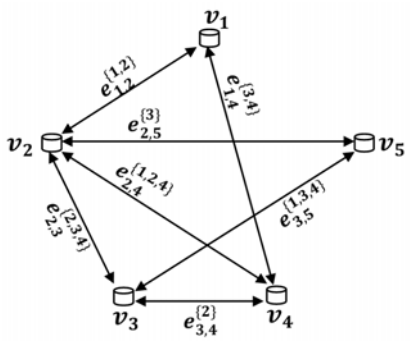
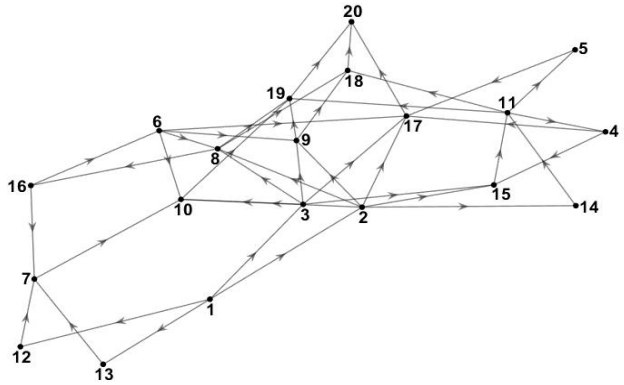
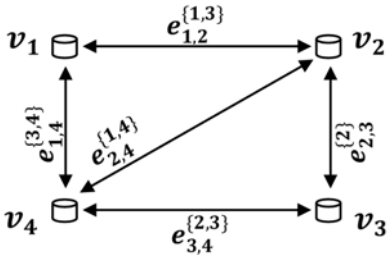
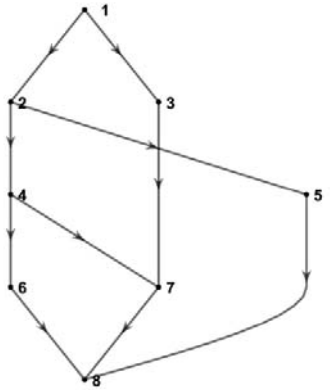
3.2.3.2 Time Complexity

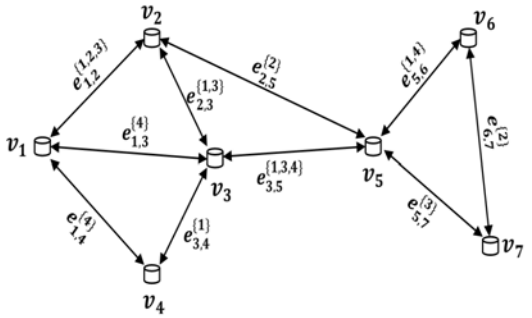
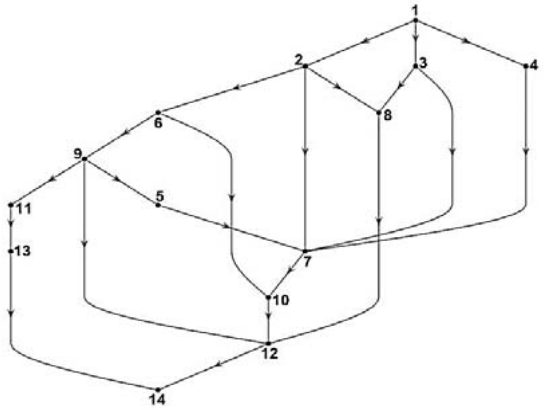
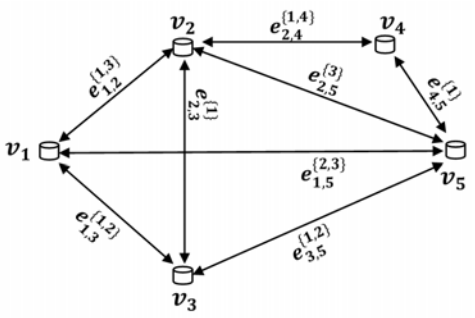
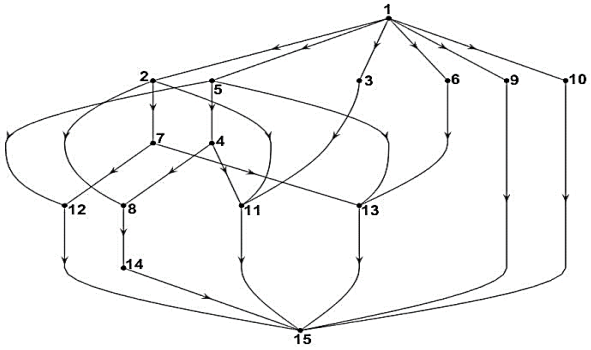
The time complexity of our proposed two-step approach can be calculated from the complexity of Step i) and Step ii). As discussed in Section 3.2.3, one can use any existing approach for Step ii) and therefore the time complexity of this step is dominated by that of the selected algorithm. Hence, in the following, we only analyze the time complexity of Step i) to generate the Line Graph. In worst case, the computational complexity of Step 1 and 2 of the proposed Algorithm 3.2 is $O(|V|^2)$, where $|V|$ is the number of nodes in the TAG. The worst case complexity of Step 3(i) would be $O(|V|)$, and of Step 3(ii), 3(iii) and 3(iv) would be $O(|V|^2)$. So, the overall complexity of Step 3 is $O(3|V|^2 + |V|) \approx O(|V|^2)$. Further, the complexity of Step 4, 5 and 6 would be $O(R^2)$, where R denotes the maximum numeric value assigned to the TSEs plus one. If $|V| \leq R$, then the worst case complexity of the proposed algorithm is $O(R^2)$.

3.2.3.3 Simulation Results

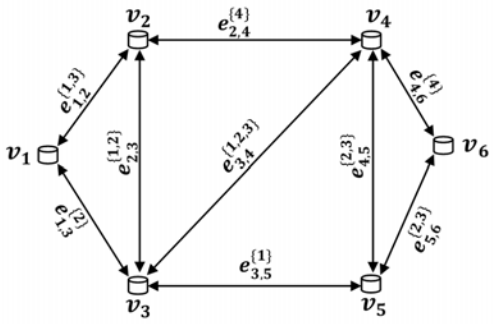
Table 3.6 shows ten Line Graphs generated by using the proposed approach on the ten arbitrary TAGs shown in Column 1. Here, the smallest and the largest indexed nodes denote v_s and v_d, respectively. Observe that TAG #1 is same as TAG shown in Figure 2.6. Further note that these ten TAGs will be considered later in Chapters 4 to 6 for performance analysis. More specifically, the tv-arborescences generated in Chapter 4 from these TAGs will be utilized later in Chapter 5 and 6 for evaluating different reliability metrics and for network management and upgrade, respectively. In addition, for each TAG, Table 3.7 shows the required CPU time (in seconds) to convert each TAG into its Line Graph. It also shows the number of nodes and edges in the generated Line Graph. Further, it collates the number of TS-MPS and TS-MCS for each TAG. Note that TS-MCS will be discussed later in Section 3.3. Table 3.7 also shows that the number of nodes and edges in Line Graph depend on the number of nodes and edges in its TAG as well as the maximum time slot τ or the number of TSEs in the graph. The table shows that our proposed Algorithm 3.2 is efficient, *i.e.*, it generates each Line Graph that contains 6 to 20 nodes and 6 to 42 edges using the maximum of 0.022 seconds CPU time.

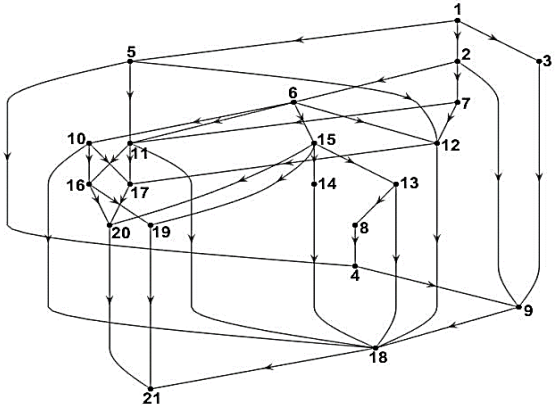
Table 3.6. Line Graphs generated from corresponding TAGs.

Example TAG	Generated Line Graph
TAG 1 	
TAG 2 	

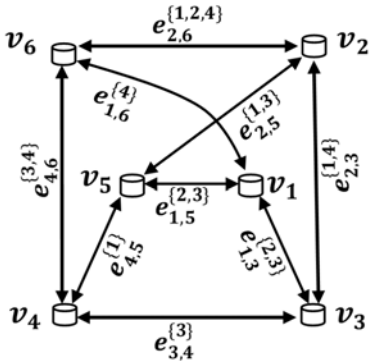
TAG 3  	
TAG 4  	

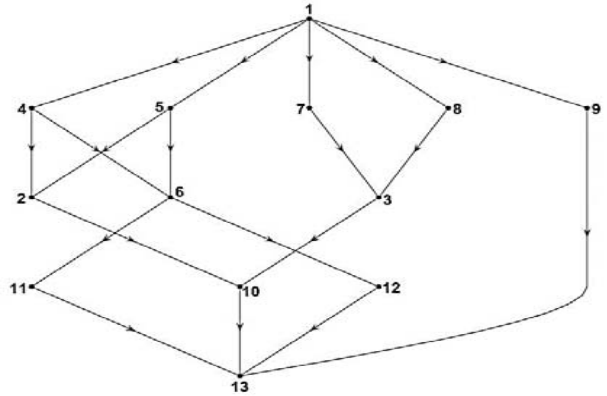
TAG 5

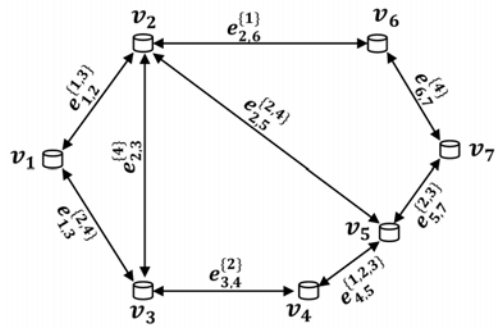
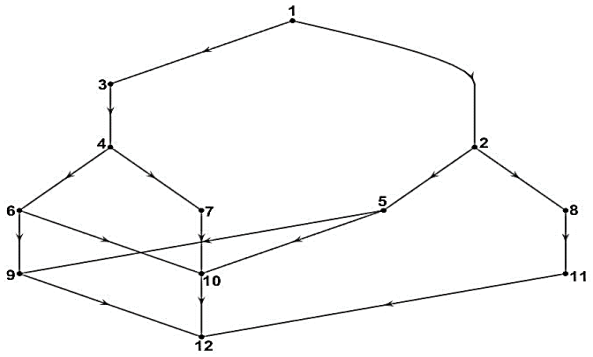
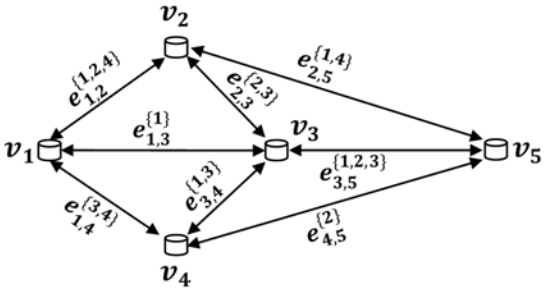
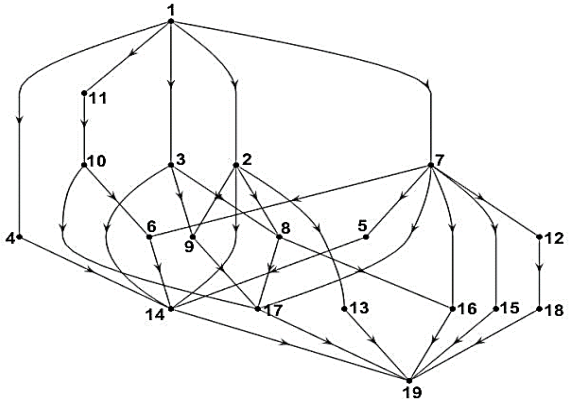

v_1
v_2
v_3
v_4
v_5
v_6
$e^{\{4\}}_{2,4}$
$e^{\{1,3\}}_{1,2}$
$e^{\{1,2\}}_{2,3}$
$e^{\{1,2,3\}}_{3,4}$
$e^{\{2,3\}}_{4,5}$
$e^{\{4\}}_{4,6}$
$e^{\{2\}}_{1,6}$
$e^{\{2,3\}}_{5,6}$
$e^{\{1\}}_{3,5}$

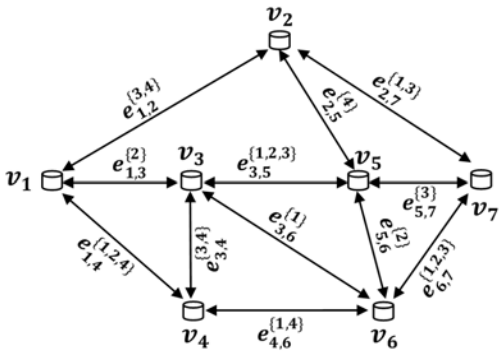
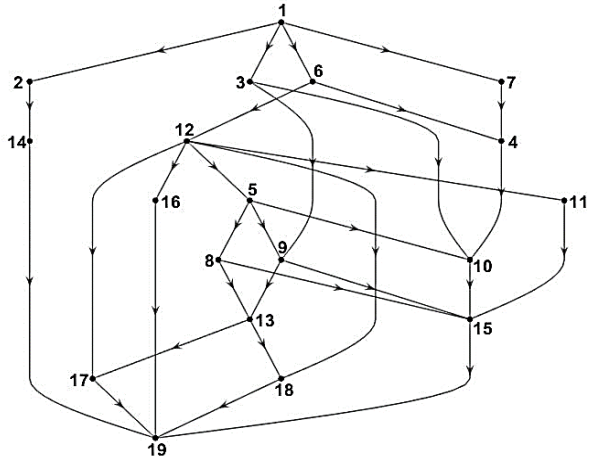
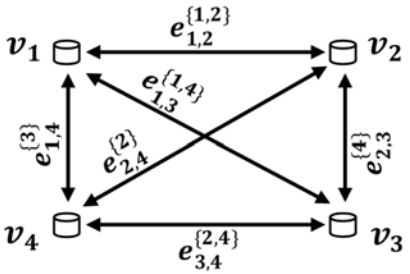
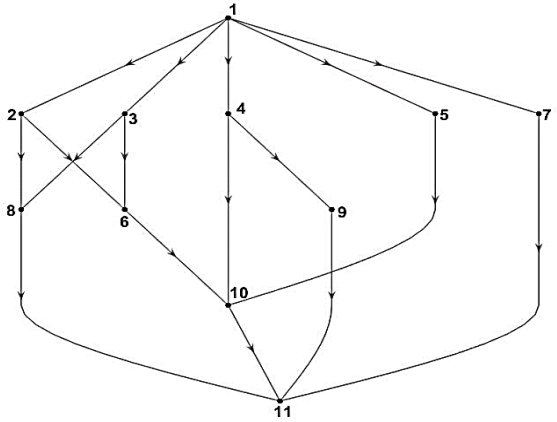

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

TAG 6


v_1
v_2
v_3
v_4
v_5
v_6
$e^{\{1,2,4\}}_{2,6}$
$e^{\{4\}}_{1,6}$
$e^{\{1,3\}}_{2,5}$
$e^{\{3,4\}}_{4,6}$
$e^{\{1,4\}}_{2,3}$
$e^{\{2,3\}}_{1,5}$
$e^{\{1\}}_{4,5}$
$e^{\{2,3\}}_{1,3}$
$e^{\{3\}}_{3,4}$


1
2
3
4
5
6
7
8
9
10
11
12
13

TAG 7	
	
TAG 8	
	

TAG 9  	
TAG 10  	

From the simulation results in Table 3.7, we also observe that the time required to generate Line Graph is insignificant as compared to that required to enumerate TS-MPS.

Table 3.7. Transformation to Line Graph and TS-MPS/MCS.

TAG	Line Graph			Total Enumerated TS-MPS and TS-MCS			
	Number of Nodes	Number of Edges	Time (seconds)	Number of TS-MPS	Time (seconds)	Number of TS-MCS	Time (seconds)
1	20	42	0.014	20	0.206	11	81.470
2	8	10	0.006	4	0.021	5	0.008
3	14	21	0.020	8	0.055	13	0.177
4	15	25	0.014	12	0.038	12	1.526
5	21	40	0.022	29	0.116	26	102.868
6	13	19	0.016	9	0.038	9	0.198
7	12	16	0.020	6	0.036	15	0.066
8	19	34	0.013	18	0.286	20	50.806
9	19	32	0.022	14	0.253	40	18.889
10	11	17	0.012	8	0.044	6	0.069

3.3 TS-MCS Enumeration Techniques

In general, the number of an MCS in most of the infrastructure based complex networks are found to be usually less than the number of an MPS, and it is advisable to use an MCS rather than the an MPS for reliability evaluation (Mishra and Chaturvedi 2009). This strategy reduces the computational time and rounding off errors involved in computations. In this section, we investigate the validity of this contention for TVCNs as well, and extend the notion of MCS of static networks to TS-MCS for TVCNs. More specifically, we devise two methods to enumerate all TS-MCS between a specified (s, d) pair of nodes of a TVCN.

Recall that in the Cartesian product based method of TS-MPS enumeration, we first generated all MPS of the underlying network, which were later assessed and expanded along with timestamps to generate all TS-MPS of the TAG. However, upon application of the same notion on all MCS of the underlying complete network, in order to enumerate all TS-MCS of the same TAG, our logic failed to produce the desired results.

Thus, we infer that the enumeration of TS-MCS is not a straightforward process as it is in the case of TS-MPS enumeration presented earlier in Section 3.2. In the upcoming sections, we present two novel techniques for enumerating all TS-MCS between a specified (s, d) *pair of nodes of a TVCN.*

3.3.1 TS-MPS Inversion based Method

In this method, we applied the well-known De Morgan's law on all TS-MPS in order to invert and subsequently minimize them using Boolean algebra. This was done with an objective to examine the principle of duality between TS-MPS and TS-MCS, and to verify the outcome with manually enumerated (using inspection) TS-MCS (Chaturvedi et al. 2018). For example, inverting and minimizing the two TS-MPS, *viz.*, $(e_{1,2}^1.e_{2,3}^1.e_{3,4}^4)$ and $(e_{1,2}^1.e_{2,3}^2.e_{3,4}^4)$, between node pair (v_1, v_4) of the TAG shown in Figure 2.1(ii) results in three TS-MCS, *i.e.*, $e_{1,2}^1$, $e_{3,4}^4$ and $(e_{2,3}^1.e_{2,3}^2)$. Their correctness can be verified by visual inspection of the Line Graph of Figure 2.5. It is evident, that the TS-MCS enumerated by inverting all TS-MPS, using De Morgan's law, matches exactly with the TS-MCS enumerated by visual inspection; thereby, confirms the existence of duality between TS-MPS and TS-MCS in TVCNs as well. Using the same idea, we enumerated the 11 TS-MCS between node pair (v_1,v_5) of the TAG given in Figure 2.6. However, as this method depends on inversion of all enumerated TS-MPS, which can be exponential in number, it can be regarded as a crude approach of TS-MCS generation with huge computational cost.

3.3.2 Line Graph based Method

In this approach, the first step is same as described earlier in Section 3.2.3., *i.e.*, transforming a TAG to a Line Graph. However, in the second step one can use any known and efficient MCS enumerating algorithm like (Ahmad 1988; Mishra and Chaturvedi 2009) for TS-MCS enumeration. It is important to note here that the performance of this method is highly dependent on the efficiency of the MCS enumeration algorithm utilized in the second step. Table 3.7 collates the number of all TS-MCS between specified (s, d) node pair for each of the ten TAGs of Table 3.6. It is interesting to observe from Table 3.7 that

except for TAG #1, 5 and 10 there are equal or more TS-MCS than TS-MPS, inconsistent with the observation in (Chaturvedi and Mishra 2009) for static graph where the total number of an MCS are expected to be less than that of an MPS. Thus, in contrast to the static networks, we infer that in TVCNs the number of TS-MCS may or may not be less in comparison to the number of TS-MPS. Further, Table 3.7 shows that, except for TAG #2, TS-MCS enumeration takes more time compared to TS-MPS enumeration.

3.4 Summary

The chapter presented three novel methods, *viz.*, Cartesian Product based, Connection Matrix based, and Line Graph based to enumerate all TS-MPS between a specified (s, d) node pair of TVCNs. Note that another novel method which utilizes all t-arborescences rooted at node v_s to enumerate all (s, d) TS-MPS is presented later in Chapter 4 after discussing pertinent prerequisites. In addition, the chapter also presented two novel methods, *viz.*, TS-MPS inversion based and Line Graph based, to enumerate all TS-MCS between a specified (s, d) node pair of TVCNs.

We have exemplified the presented algorithms with relevant illustrative examples. Further, through our analysis, we established the existence of dual relationship between TS-MPS and TS-MCS of TVCNs as it exists for an MPS and MCS in static networks. We also found through our simulations that in TVCNs the number of TS-MCS is not necessarily less than the number of TS-MPS as it exists most often in static networks.

At this juncture, it is worthy to note that as both TS-MPS and TS-MCS are dual of each other, either of them can be used to evaluate the 2-terminal reliability of TVCNs. Further note that among the two proposed *all* TS-MCS enumeration algorithms, one is dependent on inversion of all TS-MPS while the other utilizes Line Graph and enumerates TS-MCS in more time compared to TS-MPS. Therefore, in Chapter 5, we utilize only TS-MPS to demonstrate the technique for evaluating the 2-terminal reliability of device-to-device connected TVCNs.

Now, before moving any further, for more clarity, the pros and cons, applications, and time complexity of each method presented in this chapter is summarized in Table 3.8.

Table 3.8. Summary of the techniques developed for TS-MPS and TS-MCS enumeration.

Sl. No.	Enumerated Entity		Pros and Cons	Applications	Time Complexity
	§	Ɨ			
1	✓		Simple but computationally expensive, as it generates all combinations of timestamps to find all TS-MPS.	In routing algorithms finding paths between a specified (s, d) node pair and 2-terminal reliability evaluation of TVCNs.	$O(\lvert V\rvert!\,(\tau^{\lvert V\rvert}))$
2	✓		Efficiently generates all TS-MPS between each node pair in one go; but, for large networks, use of Boolean operations becomes time consuming.	Suitable for an LEO satellite network to find all routing opportunities, *i.e.*, among each pair of nodes.	$O(\lvert V\rvert^3\,\tau)$
3	✓	✓	Only method which can enumerate both TS-MPS and TS-MCS; however, depends heavily on the performance of an MPS and/or MCS enumeration algorithm used after generation of Line Graph.	Identification of routing opportunities, weak links, survivability analysis and calculation of reliability bounds in TVCNs.	*Step i)*: $O(R^2)$; *Step ii)*: Depends on the complexity of the selected MPS and/or MCS enumeration algorithm.
4		✓	Depends on enumerated TS-MPS; therefore, it is crude and costly.	Identification of weak links and unreliability evaluation of TVCNs.	Not analysed

1: Cartesian product based method; 2: Connection matrix based method; 3: Line Graph based method; 4: TS-MPS Inversion based method; §: TS-MPS; Ɨ: TS-MCS.

Chapter 4 Enumeration of Timestamped Arborescences for Network Broadcast

4.1 Introduction

In *static* or conventional networks, the broadcast communication model is usually accomplished by constructing *arborescences* and *trees* for *directed* and *undirected* networks, respectively. However, unlike static networks, each edge of a TVCN has a time window or slots in which it remains active. Thus, existing solutions used to generate arborescences cannot be applied in a TVCN as they assume a fixed topology. Hence, arborescences of TVCNs must consider edge schedules.

Many algorithms have been proposed to enumerate the Minimum Cost Tree/Arborescence (MCT/MCA) or all trees/arborescences of undirected/directed static networks; see (Chakraborty et al. 2019) for a recent review. In (Huang et al. 2015), the authors presented algorithms to generate two types of MCA in TVCNs that either minimize time or cost of information dissemination. In (Gunturi et al. 2010), the authors presented two algorithms to find a MCT in TVCNs. Similarly, the authors of (Bhadra and Ferreira 2003) presented a modification of the Prim-Dijkstra algorithm to enumerate a rooted MCA for each node of a TVCN. Reference (Chen et al. 2016) presents three heuristic algorithms to generate a directed spanning tree for network convergecasting in TVCNs. In (Coscia 2018), the authors used arborescences to estimate hierarchicalness in directed complex networks. In contrast, as explained in Definition 2.3, this chapter introduces two new concepts on spanning arborescences for TVCNs: (i) t-arborescence – an arborescence in which all edges have contact schedules, and (ii) tv-arborescence – a t-arborescence that requires time-ordered edges. We discuss properties of these arborescences, and propose two novel methods, *viz.*, Method 1 and Method 2, that utilize Tutte's Matrix-Tree theorem (Aigner 2007) to enumerate all tv-arborescences. The results from this work have been reported in (Khanna et al. 2020c).

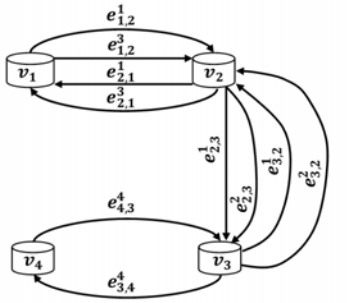
The chapter is organized as follows: Section 4.2 first presents network model used in this chapter and later describes significant properties of arborescences of TVCNs. Section 4.3 starts with preliminaries required for the explanation of our proposed Method 1 and Method 2. It then presents the two methods in detail and provides necessary illustrations for better elucidation. The section also provides time complexity analysis of both methods. Section 4.4 presents an algorithm to enumerate all TS-MPS between a specified (s, d) node pair by using all t-arborescences rooted at node v_s. Section 4.5 presents the simulation results followed by Section 4.6 that summarizes this chapter.

4.2 Network Model and Properties

We use $G(V, E)$ to represent a loopless directed graph where V is a set of nodes, and E is a set of bidirectional edges between nodes $v_i, v_j \in V$. Let $e_{i,j} \in E$ represent a directed edge from node v_i to v_j. Without loss of generality, we assume set E also contains edge $e_{j,i}$ for each edge $e_{i,j} \in E$. Let TS denote a $|V| \times |V|$ matrix that contains a sequence of *bidirectional contacts* or *timestamps* for each edge $e_{i,j} \in E$. Each entry $TS(i,j) \in TS$ stores the sequence of timestamps of edge $e_{i,j}$ in an ascending order. We denote the k^{th} timestamp of edge $e_{i,j}$ as $TS(i,j,k)$, where $k = 1, 2, \ldots, |TS(i,j)|$. We set $TS(i,j) = \{\}$, if edge $e_{i,j} \notin E$. Further, recall that τ is the maximum timestamp value or period of a TVCN. Thus, each $t \in TS(i,j)$ has range $[1, \tau]$. Figure 4.1(i) shows the TS matrix of the TAG shown in Figure 2.1(ii).

$$TS = \begin{bmatrix} \{\} & \{1,3\} & \{\} & \{\} \\ \{1,3\} & \{\} & \{1,2\} & \{\} \\ \{\} & \{1,2\} & \{\} & \{4\} \\ \{\} & \{\} & \{4\} & \{\} \end{bmatrix}$$

(i)



(ii)

Figure 4.1. (i) The TS matrix, and (ii) the directed multigraph representation of the TAG shown in Figure 2.1(ii).

We use $G'(V, E^{TS})$ to represent a TAG, where E^{TS} is a set of edges, each of which has a sequence of timestamps in TS. More specifically, $e_{i,j}^{TS(i,j)} \in E^{TS}$ represents edge $e_{i,j} \in E$ that is associated with a sequence of timestamps $TS(i,j) \in TS$. Formally, we have $E^{TS} = \{e_{i,j}^{TS(i,j)} \mid \forall\, e_{i,j} \in E,\ \forall\, TS(i,j) \in TS\}$. For example, for edge $e_{1,2} \in E$ and $TS(1,2) = \{1,3\}$, the TAG in Figure 2.1(ii) contains edge $e_{1,2}^{\{1,3\}} \in E^{TS}$. We assume that the topology of a TVCN remains static for one time slot but may or may not vary over time slots. This means in any given slot, if a path exists between nodes, then packets may be transmitted to all of them within that slot. We also assume that nodes have sufficient capacity to buffer data packets.

We use $G''(V, E'')$ to represent a directed multigraph of a TAG $G'(V, E^{TS})$. Here, E'' is a set of edges, each of which is associated with only *one* timestamp $t \in TS(i,j)$, for each edge $e_{i,j} \in E$. More specifically, each $e_{i,j}^{t} \in E''$ is a TSE that represents edge $e_{i,j} \in E$ that is active at time $t \in TS(i,j)$. Formally, we have $E'' = \{e_{i,j}^{t} \mid \forall\, e_{i,j} \in E,\ \forall\, t \in TS(i,j)\}$. Thus, each edge $e_{i,j}^{TS(i,j)} \in E^{TS}$ in the TAG corresponds to $|TS(i,j)|$ TSEs $e_{i,j}^{t} \in E''$ in its corresponding directed multigraph, where an edge is created for each $t \in TS(i,j)$. As an example, Figure 4.1(ii) shows the directed multigraph representation of the TAG shown in Figure 2.1(ii). Note that G, G' and G'' contain the same set of nodes V. Further, the number of edges in a directed graph and TAG are equal, *i.e.*, $|E| = |E^{TS}|$. In addition, the number of edges in the directed multigraph is given as $|E''| = \sum |TS(i,j)|$, for all edges $e_{i,j} \in E$.

We use A_j to denote the set of all arborescences originating from root node $v_j \in V$ in TAG $G'(V, E^{TS})$. Let A_j^i be an arborescence i in the set A_j, *i.e.*, $\mathrm{A}_j = \{\mathrm{A}_j^i \mid i = 1, 2, \ldots\}$. Let TA_j be a set of all t-arborescences originating from root node $v_j \in V$ in TAG $G'(V, E^{TS})$, and TA_j^i is the i^{th} t-arborescence in TA_j, *i.e.*, $\mathrm{TA}_j = \{\mathrm{TA}_j^i \mid i = 1, 2, \ldots\}$. Each arborescence $\mathrm{A}_j^i \in \mathrm{A}_j$ can be transformed into one or more t-arborescences in TA_j by assigning different set of timestamps, generated by taking one timestamp from each $TS(a,b)$ corresponding to each edge $e_{a,b} \in \mathrm{A}_j^i$. Note that set TA_j contains both *valid* and *invalid* t-arborescences.

Let TVA_j be a set of all tv-arborescences originating from root node $v_j \in V$ in TAG $G'(V, E^{TS})$, and TVA_j^i is the i^{th} tv-arborescence in TVA_j, *i.e.*, $\text{TVA}_j = \{\text{TVA}_j^i | \, i = 1, 2, \dots\}$. It is important to note here that any TVA_j^i has $(|V| - 1)$ time-ordered TSEs, each of which is between different connected node pairs. Further, we denote sets A_j, TA_j, and TVA_j for each $v_j \in V$ in TAG $G'(V, E^{TS})$ by $\text{A}_{All} = \{\text{A}_j \big| \, \forall \, v_j \in V \}$, $\text{TA}_{All} = \{\text{TA}_j \big| \, \forall \, v_j \in V \}$, and $\text{TVA}_{All} = \{\text{TVA}_j \big| \, \forall \, v_j \in V\}$, respectively.

4.2.1 Properties of Arborescences of TVCNs

This section presents some properties related to tv-arborescences. Specifically, Proposition 4.1 relates to the time complexity required to compute the maximum number of tv-arborescences, while Proposition 4.2 and 4.3 show the relationship among the number of arborescences, t-arborescences and tv-arborescences. Further, Proposition 4.4 and 4.5 give the relationship between tv-arborescences and t-arborescences, respectively, from different root nodes.

Proposition 4.1: The upper bound of $|\text{TVA}_j|$ for each root node $v_j \in V$ can be computed in polynomial time.

Proof: By definition, each tv-arborescence is a t-arborescence, meaning each distinct t-arborescence results in at most one tv-arborescence. Therefore, $|\text{TA}_j|$ gives the upper bound of $|\text{TVA}_j|$. We can compute $|\text{TA}_j|$ in polynomial time as follows. First, we transform a TAG into a directed multigraph $G''(V, E'')$. Note that the transformation can be done in polynomial time. After that, we apply Tutte's Matrix-Tree Theorem (Aigner 2007) on $G''(V, E'')$ to compute $|\text{TA}_j|$. In particular, reference (Miklós 2018) shows that Tutte's Matrix-Tree Theorem (Aigner 2007) can be used to count, in polynomial time, the number of arborescences of a directed multigraph rooted at node v_j, *i.e.*, in our case is $|\text{TA}_j|$ of $G''(V, E'')$. Thus, the upper bound of $|\text{TVA}_j|$ can be computed in polynomial time. ■

Proposition 4.2: The number of t-arborescences rooted at node v_j is always greater than or equal to the number of arborescences rooted at the same node, *i.e.*, $|\mathrm{TA}_j| \geq |\mathrm{A}_j|$.

Proof: For each edge $e_{a,b}$ in an arborescence $\mathrm{A}_j^i \in \mathrm{A}_j$, let $\mathrm{TSE}(a,b)$ be its corresponding set of TSEs, each of which is generated by appending one timestamp t at a time from $TS(a,b)$ for all $t \in TS(a,b)$. Each t-arborescence is formed by taking one TSE from each $\mathrm{TSE}(a,b)$, for each edge $e_{a,b} \in \mathrm{A}_j^i$. When at least one edge $e_{a,b} \in \mathrm{A}_j^i$ has more than one timestamp, *i.e.*, $|TS(a,b)| > 1$, we have $|\mathrm{TA}_j| > |\mathrm{A}_j|$. On the other hand, when each edge $e_{a,b}$ has only one timestamp, *i.e.*, $|TS(a,b)| = 1$, we have $|\mathrm{TA}_j| = |\mathrm{A}_j|$. ■

Proposition 4.3: The number of tv-arborescences rooted at each node $v_j \in V$ is always less than or equal to the number of t-arborescences from the same root node, *i.e.*, $|\mathrm{TVA}_j| \leq |\mathrm{TA}_j|$, for all node $v_j \in V$.

Proof: By definition, a tv-arborescence rooted at a node $v_j \in V$ is a *valid* t-arborescence rooted at the same node. When the root node has at least one *invalid* t-arborescence, the node has $|\mathrm{TVA}_j| < |\mathrm{TA}_j|$. On the other hand, if all t-arborescences from the node are valid, node v_j has the same number of tv-arborescences and t-arborescences, *i.e.*, $|\mathrm{TVA}_j| = |\mathrm{TA}_j|$. ■

From Proposition 4.3 we can infer that in a TVCN although there may be many potential bidirectional contacts amongst different pair of nodes, they are not always beneficial for network broadcast because they can produce many *invalid* t-arborescences. It is worth noting that in general there might be exponential number of arborescences in terms of $|E|$ for a given graph $G(V,E)$ and thus their enumeration cannot be done in polynomial time (Mihalák et al. 2016). Proposition 4.2 implies that a set TA_j for each root $v_j \in V$ in general also cannot be enumerated in polynomial time. While we have $|\mathrm{TA}_j| \geq |\mathrm{A}_j|$ and $|\mathrm{TA}_j| \geq |\mathrm{TVA}_j|$, in general, there is no direct relationship between the number of arborescences $|\mathrm{A}_j|$ and the number of tv-arborescences $|\mathrm{TVA}_j|$ from the root node $v_j \in V$.

The reason is because each arborescence $A_j^i \in A_j$, with timestamps attached to its constituting edges, can result in either zero, one or more than one tv-arborescence $TVA_j^i \in TVA_j$. Recall that a tv-arborescence is a *valid* t-arborescence generated from the combinations of timestamps, taking one timestamp from each edge of A_j^i. Thus, i) if each $A_j^i \in A_j$ results in zero $TVA_j^i \in TVA_j$, then we have $|A_j| > |TVA_j|$ and ii) if each $A_j^i \in A_j$ results in exactly one $TVA_j^i \in TVA_j$, then we have $|A_j| = |TVA_j|$. However, if there exists at least one arborescence $A_j^i \in A_j$ which results in more than one $TVA_j^i \in TVA_j$ and at least one other which does not result in any $TVA_j^i \in TVA_j$, then we cannot conclude whether $|A_j| > |TVA_j|$, $|A_j| = |TVA_j|$ or $|A_j| < |TVA_j|$.

Proposition 4.4: Any tv-arborescence from root node v_i cannot be a tv-arborescence from root node v_j, where $v_i \neq v_j$ and both $v_i, v_j \in V$.

Proof: In a tv-arborescence from root node v_i, every vertex $v_j \neq v_i$ is accessible from the root v_i. However, the root v_i is inaccessible from any vertex $v_j \neq v_i \in V$. The above assertion is true because the root of a tv-arborescence has zero in-degree and all other vertices have in-degree of one. Similarly, in a tv-arborescence from root node v_j, every other vertex $v_i \neq v_j$ is accessible from the root v_j. However, any vertex $v_i \neq v_j$ cannot access the root v_j. Thus, all tv-arborescences from root node v_i and root node v_j are mutually exclusive to each other. ■

Proposition 4.5: Any t-arborescence from root node v_i cannot be a t-arborescence from root node v_j, where $v_i \neq v_j$ and both $v_i, v_j \in V$.

Proof: We can prove this proposition using similar steps as in Proposition 4.4, and thereby, the proof is omitted. ■

From Proposition 4.4, we can compute the total number of tv-arborescences of a TVCN by taking the sum of tv-arborescences from each root node $v_j \in V$, *i.e.*, $|TVA_{All}| =$

$\sum_{j=1}^{|V|}(|\mathrm{TVA}_j|)$. Similarly, using Proposition 4.5, we have $|\mathrm{TA}_{All}| = \sum_{j=1}^{|V|}(|\mathrm{TA}_j|)$. Further, from Proposition 4.3 to 4.5, one can conclude that the total number of tv-arborescences in a TVCN cannot be larger than the total number of t-arborescences, *i.e.*, $|\mathrm{TVA}_{All}| \leq |\mathrm{TA}_{All}|$. Similarly, from Proposition 4.2 and 4.5, we conclude that $|\mathrm{A}_{All}| \leq |\mathrm{TA}_{All}|$.

4.3 Timestamped Valid Arborescences Enumeration Algorithms

This section presents two methods, *i.e.*, Method 1 and Method 2, for enumerating TVA_{All}. Both methods use Tutte's Matrix-Tree Theorem (Aigner 2007) – a classical result of graph theory. We first define some necessary background on Tutte's Matrix-Tree Theorem in Section 4.3.1 before presenting our proposed methods in Section 4.3.2. and Section 4.3.3.

4.3.1 Background

We start with the necessary notations and concepts before formally stating Tutte's Matrix-Tree theorem.

1) Laplace Matrix

Given a directed graph $G = (V, E)$, we denote by $\mathcal{A} = (a_{ij})$ a $|V| \times |V|$ matrix with a_{ij} denoting the number of directed edges from node v_i to v_j. Furthermore, let D^+ be a diagonal matrix of order $|V| \times |V|$, with the (i, i)-th entry representing the in-degree of the i^{th} vertex and its remaining entries are set to zero. The Laplace matrix $L^+(G)$ is then $L^+(G) = D^+ - \mathcal{A}$. Further, any matrix $Q = (q_{ij})$ of size $|V| \times |V|$ can be reduced to a submatrix $Q_{r,c}$ of size $(|V| - 1) \times (|V| - 1)$ by deleting the r^{th} row and the c^{th} column, where $v_r, v_c \in V$.

2) Determinant

For a $|V| \times |V|$ matrix $M = (m_{jl})$, its determinant is $\mathrm{Det}\ M = \sum_{(l_1,\ldots,l_{|V|})} \pm m_{1l_1}\, m_{2l_2} \ldots m_{|V|l_{|V|}}$. Here, the summation is over all permutations $(l_1, \ldots, l_{|V|})$ of $(1, 2, \ldots, |V|)$. Note that a permutation can be transformed to another by interchanging

consecutive pairs of numbers $(1, 2, \ldots, |V|)$. Such an interchange is called as *transposition*. A permutation requiring an even number of transpositions is called *even*, and *odd* otherwise. The above summation contains positive sign for each *even* permutation, and negative sign for each *odd* permutation. For example, if $|V| = 1$, then Det $M = m_{11}$. If $|V| = 2$, then Det $M = m_{11}m_{22} - m_{12}m_{21}$ and if $|V| = 3$, then Det $M = m_{11}m_{22}m_{33} + m_{12}m_{23}m_{31} + m_{13}m_{21}m_{32} - m_{13}m_{22}m_{31} - m_{11}m_{23}m_{32} - m_{12}m_{21}m_{33}$. Thus, there are $|V|!$ summands in the definition of determinant of a $|V| \times |V|$ matrix.

3) Tutte's Matrix-Tree Theorem

Tutte's Matrix-Tree Theorem is a generalization of a classical theorem from graph theory due to Kirchhoff (Aigner 2007). It can be used to determine the number of arborescences diverging from vertex v_j, *i.e.*, N_j^+. We reproduce the theorem here for the sake of completeness.

Theorem 4.1 (Aigner 2007)**.** Let $G = (V, E)$ be a loopless directed graph, $V = \{1, \ldots, |V|\}$. Then, for any $v_r, v_c \in V$ (where $r = c$ is possible),

$$N_r^+ = (-1)^{r+c} \text{ Det } L^+(G)_{c,r} \tag{4.1}$$

$$N_r^- = (-1)^{r+c} \text{ Det } L^-(G)_{r,c} \tag{4.2}$$

Note that Equation (4.2) provides the number of *converging* arborescences rooted at sink node v_r, *i.e.*, N_r^-. The Laplace matrix $L^-(G)$ from which $L^-(G)_{r,c}$ is derived for use in Equation (4.2) utilizes D^- instead of D^+, wherein the (i, i)-th entry of D^- is the out-degree of i^{th} vertex. In our work, our focus is on network *broadcasting* or Equation (4.1) and not on convergecasting or Equation (4.2). It is also worth mentioning here that in Theorem 4.1, multiple edges may be considered between $r \neq c$ in either direction (Aigner 2007; Miklós 2018). Further, Det $L^+(G)_{c,r}$ can be a *positive number*, *negative number* or *zero*. However, as the number of arborescences cannot be negative, the formula has a multiplier $(-1)^{r+c}$ to ensure N_r^+ is positive. Without loss of generality, we may avoid the multiplier by deleting the (r, r)-th row and column from $L^+(G)$ to obtain $L^+(G)_{r,r}$. This

change will always provide a positive determinant. Therefore, here onwards we consider $r = c$ for computing N_r^+ and denote $L^+(G)_{c,r}$ with $r = c$ as $\hat{L}_r^+$. Further, we adapt the method in (Masbaum and Vaintrob 2002) to modify Equation (4.1) for enumerating all arborescences from a directed graph $G(V, E)$ or all t-arborescences from a directed multigraph $G''(V, E'')$. The steps for enumerating all t-arborescences are explained in the next section.

4.3.2 Method 1

Method 1 generates all tv-arborescences from the corresponding t-arborescence of each root node $v_j \in V$. Method 1 consists of three steps:

1) ***Generate a directed multigraph $G''(V, E'')$ from $G'(V, E^{TS})$*:** For each edge $e_{i,j}^{TS(i,j)} \in E^{TS}$ of the TAG $G'(V, E^{TS})$, we generate $|TS(i,j)|$ TSEs $e_{i,j}^t \in E''$ from node v_i to node v_j, for all $t \in TS(i,j)$.

2) ***Enumerate TA_{All} from $G''(V, E'')$*:** We extend Equation (4.1) to enumerate TA_{All} for the directed multigraph $G''(V, E'')$ as follows:

   a) *Generate the Laplace matrix $L^+(G'')$ by filling it with indeterminates/variables*: This is accomplished by filling the (i,i)-th entry of $L^+(G'')$ with variable(s) representing in-degree edge(s). The remaining entries of $L^+(G'')$ are completed by placing a negative sign and filling in variable(s) representing edge(s) from node v_i to v_j. Thus, each entry of $L^+(G'')$ contains variable(s) instead of number of variables to represent both in-degree edge(s) and edge(s) from node v_i to v_j.

   b) *Generate a reduced Laplace matrix $\hat{L}_j^+$ for all root node $v_j \in V$ and calculate its determinant*: Recall that the reduced Laplace matrix $\hat{L}_j^+$ is obtained by deleting the j^{th} row and column from $L^+(G'')$. The method presented in Section 4.3.1. *2)* is then used to compute the determinant of $\hat{L}_j^+$, producing a polynomial function. Each non-

vanishing monomial in this polynomial has coefficient one, and corresponds to a t-arborescence. Thus, the determinant of $\hat{L}_j^+$ for all root node $v_j \in V$ provides TA_{All} of the directed multigraph.

3) Generate TVA_{All} from TA_{All}: This step aims to find each t-arborescence that is tv-arborescence. In particular, this step discards *invalid* t-arborescences from TA_j and yields TVA_j for all root nodes $v_j \in V$, *i.e.*, TVA_{All}.

The details of Step ***3)*** of Method 1 are realized by Algorithm 4.1. Next we explain its working in detail.

Algorithm 4.1. Gen_TVA_{All}fromTA$_{All}$

Input: TA_{All}
Output: TVA_{All}

```
1.  TVA_All ← {}
2.  for (j ← 1 to |V|)
3.     TVA_j ← {}
4.     for (i ← 1 to |TA_j|)
5.        notValid ← FALSE
6.        nodeTime ← [0, ... , 0]
7.        for each e^t_{a,b} ∈ TA^i_j
8.           nodeTime[b] ← t
9.        end for
10.       for each e^t_{a,b} ∈ TA^i_j
11.          if (nodeTime[a] > t)
12.             notValid ← TRUE
13.             break
14.          end if
15.       end for
16.       if (notValid = FALSE)
17.          Insert TA^i_j to TVA_j
18.       end if
19.    end for
20.    Insert TVA_j to TVA_All
21. end for
```

The algorithm uses TVA_j to store all tv-arborescences from root node $v_j \in V$, and TVA_{All} to contain all tv-arborescences in the network. Lines 4-19 check validity of each t-arborescence with respect to time. Line 5 initializes the variable **notValid** to 'FALSE', meaning t-arborescence TA_j^i is first assumed to be valid. Lines 7-9 sets array **nodeTime**[1, 2,…, $|V|$] such that each entry b contains timestamp t of the ending node b of each TSE $e_{a,b}^t \in \text{TA}_j^i$. For example, in a TA_1^i say $(e_{1,2}^3. e_{2,3}^1. e_{3,4}^4)$, we have **nodeTime**[2] = 3, **nodeTime**[3] = 1, **nodeTime**[4] = 4, and **nodeTime** [1] = 0. Note that we have **nodeTime**[j] = 0 for root node v_j. Thus, we have **nodeTime** = (0, 3, 1, 4). Algorithm 4.1 uses array **nodeTime** to discard all invalid t-arborescences. More specifically, using **nodeTime**, Lines 10-15 aim to check if all edges in t-arborescence TA_j^i are time-ordered. If any non-complying edge is found, the algorithm sets **notValid** to TRUE, and stops checking the remaining edges; see Lines 11-14. For above example, we have **nodeTime**[2] = 3, meaning TSE $e_{1,2}^3$ is activated at time t = 3 that is later than its successive edge $e_{2,3}^1$ which is available at time t = 1; thus t-arborescence $(e_{1,2}^3. e_{2,3}^1. e_{3,4}^4)$ is invalid. Otherwise, TA_j^i is a tv-arborescence and thus it is added to set TVA_j; see Lines 16-18. The steps are repeated from Line 2 to obtain all tv-arborescences in the network.

Illustration 4.1: We now use the TVCN shown in Figure 2.1(ii) to explain Method 1. Given the TAG $G'(V, E^{TS})$ shown in Figure 2.1(ii), Step ***1)*** generates the directed multigraph $G''(V, E'')$ shown in Figure 4.1(ii). Recall that the TVCN has a period of four slots and each timestamp is represented as a bidirectional TSE. Step ***2a)*** obtains the following Laplace matrix $L^+(G'')$ for the directed multigraph:

$$L^+(G'') = D^+ - \mathcal{A}$$
$$= \begin{bmatrix} (e_{2,1}^1 + e_{2,1}^3) & -(e_{1,2}^1 + e_{1,2}^3) & 0 & 0 \\ -(e_{2,1}^1 + e_{2,1}^3) & (e_{1,2}^1 + e_{1,2}^3 + e_{3,2}^1 + e_{3,2}^2) & -(e_{2,3}^1 + e_{2,3}^2) & 0 \\ 0 & -(e_{3,2}^1 + e_{3,2}^2) & (e_{2,3}^1 + e_{2,3}^2 + e_{4,3}^4) & -e_{3,4}^4 \\ 0 & 0 & -e_{4,3}^4 & e_{3,4}^4 \end{bmatrix}$$

From $L^+(G'')$, Step ***2b)*** of Method 1 generates $\hat{L}_j^+$, for each root node v_j, for j = 1 to 4. For example, for j = 1, it obtains

$$\hat{L}^{+}_{j=1} = \begin{bmatrix} (e^1_{1,2} + e^3_{1,2} + e^1_{3,2} + e^2_{3,2}) & -(e^1_{2,3} + e^2_{2,3}) & 0 \\ -(e^1_{3,2} + e^2_{3,2}) & (e^1_{2,3} + e^2_{2,3} + e^4_{4,3}) & -e^4_{3,4} \\ 0 & -e^4_{4,3} & e^4_{3,4} \end{bmatrix}$$

As $\hat{L}^{+}_{j=1}$ is a 3×3 matrix, from the discussion in Section 4.3.1.*2)*, it will have a total of six summands in its determinant. Each summand, *e.g.*, $m_{11}m_{22}m_{33}$, represents one or more entries at the specified locations, *viz.*, m_{11}, m_{22} and m_{33} in the matrix $\hat{L}^{+}_{j=1}$. For example, here m_{11} corresponds to $(e^1_{1,2} + e^3_{1,2} + e^1_{3,2} + e^2_{3,2})$, m_{22} corresponds to $(e^1_{2,3} + e^2_{2,3} + e^4_{4,3})$ and m_{33} corresponds to $e^4_{3,4}$. So, in total, the 12 (= $4 \times 3 \times 1$) combinations of the elements in $(e^1_{1,2} + e^3_{1,2} + e^1_{3,2} + e^2_{3,2})$, $(e^1_{2,3} + e^2_{2,3} + e^4_{4,3})$ and $e^4_{3,4}$ will give us $m_{11}m_{22}m_{33}$. Similarly, we can find each of the other five summands. Simplifying the polynomial obtained from the determinant of $\hat{L}^{+}_{j=1}$ by removing duplicate terms will leave us with four unique monomials or t-arborescences from node v_1. Table 4.1 collates the set TA_j of all root node $v_j \in V$. Thus, the directed multigraph shown in Figure 4.1(ii) has $|\text{TA}_{All}|$ = 16.

Table 4.1. Timestamped arborescences of the TAG of Figure 2.1(ii) using Method 1.

<table>
<tr><td colspan="2">$\hat{\boldsymbol{L}}^{+}_{\mathbf{1}} = \begin{bmatrix} (e^1_{1,2} + e^3_{1,2} + e^1_{3,2} + e^2_{3,2}) & -(e^1_{2,3} + e^2_{2,3}) & 0 \\ -(e^1_{3,2} + e^2_{3,2}) & (e^1_{2,3} + e^2_{2,3} + e^4_{4,3}) & -e^4_{3,4} \\ 0 & -e^4_{4,3} & e^4_{3,4} \end{bmatrix}$</td></tr>
<tr><td>$\mathbf{TA_1}$ = $\{(e^1_{1,2}.e^1_{2,3}.e^4_{3,4}), (e^1_{1,2}.e^2_{2,3}.e^4_{3,4}), (e^3_{1,2}.e^1_{2,3}.e^4_{3,4}), (e^3_{1,2}.e^2_{2,3}.e^4_{3,4})\}$</td><td>$\mathbf{TVA_1}$ = $\{(e^1_{1,2}.e^1_{2,3}.e^4_{3,4}), (e^1_{1,2}.e^2_{2,3}.e^4_{3,4})\}$</td></tr>
<tr><td colspan="2">$\hat{\boldsymbol{L}}^{+}_{\mathbf{2}} = \begin{bmatrix} (e^1_{2,1} + e^3_{2,1}) & 0 & 0 \\ 0 & (e^1_{2,3} + e^2_{2,3} + e^4_{4,3}) & -e^4_{3,4} \\ 0 & -e^4_{4,3} & e^4_{3,4} \end{bmatrix}$</td></tr>
<tr><td>$\mathbf{TA_2}$ = $\{(e^1_{2,1}.e^1_{2,3}.e^4_{3,4}), (e^1_{2,1}.e^2_{2,3}.e^4_{3,4}), (e^3_{2,1}.e^1_{2,3}.e^4_{3,4}), (e^3_{2,1}.e^2_{2,3}.e^4_{3,4})\}$</td><td>$\mathbf{TVA_2}$ = $\{(e^1_{2,1}.e^1_{2,3}.e^4_{3,4}), (e^1_{2,1}.e^2_{2,3}.e^4_{3,4}), (e^3_{2,1}.e^1_{2,3}.e^4_{3,4}), (e^3_{2,1}.e^2_{2,3}.e^4_{3,4})\}$</td></tr>
<tr><td colspan="2">$\hat{\boldsymbol{L}}^{+}_{\mathbf{3}} = \begin{bmatrix} (e^1_{2,1} + e^3_{2,1}) & -(e^1_{1,2} + e^3_{1,2}) & 0 \\ -(e^1_{2,1} + e^3_{2,1}) & (e^1_{1,2} + e^3_{1,2} + e^1_{3,2} + e^2_{3,2}) & 0 \\ 0 & 0 & e^4_{3,4} \end{bmatrix}$</td></tr>
<tr><td>$\mathbf{TA_3}$ = $\{(e^4_{3,4}.e^1_{3,2}.e^1_{2,1}), (e^4_{3,4}.e^2_{3,2}.e^1_{2,1}), (e^4_{3,4}.e^1_{3,2}.e^3_{2,1}), (e^4_{3,4}.e^2_{3,2}.e^3_{2,1})\}$</td><td>$\mathbf{TVA_3}$ = $\{(e^4_{3,4}.e^1_{3,2}.e^1_{2,1}), (e^4_{3,4}.e^1_{3,2}.e^3_{2,1}), (e^4_{3,4}.e^2_{3,2}.e^3_{2,1})\}$</td></tr>
</table>

$\hat{\boldsymbol{L}}_{\boldsymbol{4}}^{+} = \begin{bmatrix} (e_{2,1}^{1} + e_{2,1}^{3}) & -(e_{1,2}^{1} + e_{1,2}^{3}) & 0 \\ -(e_{2,1}^{1} + e_{2,1}^{3}) & (e_{1,2}^{1} + e_{1,2}^{3} + e_{3,2}^{1} + e_{3,2}^{2}) & -(e_{2,3}^{1} + e_{2,3}^{2}) \\ 0 & -(e_{3,2}^{1} + e_{3,2}^{2}) & (e_{2,3}^{1} + e_{2,3}^{2} + e_{4,3}^{4}) \end{bmatrix}$	
$\mathbf{TA_4} =$ $\{(e_{4,3}^{4}.e_{3,2}^{1}.e_{2,1}^{1}), (e_{4,3}^{4}.e_{3,2}^{1}.e_{2,1}^{3}), (e_{4,3}^{4}.e_{3,2}^{2}.e_{2,1}^{1}), (e_{4,3}^{4}.e_{3,2}^{2}.e_{2,1}^{3})\}$	$\mathbf{TVA_4} =$ *NULL*

To illustrate Step ***3)***, consider the set TA_1 with four t-arborescences. Let us assess the validity of t-arborescences $(e_{1,2}^{1}.e_{2,3}^{1}.e_{3,4}^{4})$ and $(e_{1,2}^{3}.e_{2,3}^{2}.e_{3,4}^{4})$. For the first t-arborescence, we have **nodeTime** = (0, 1, 1, 4). Line 11 of Algorithm 1 finds that all TSEs, *viz.*, $e_{1,2}^{1}$, $e_{2,3}^{1}$, and $e_{3,4}^{4}$, are time-ordered; thus variable **notValid** remains FALSE and the t-arborescence is a tv-arborescence and is stored in TVA_1. The second t-arborescence has **nodeTime** = (0, 3, 2, 4) and for $e_{2,3}^{2}$, **nodeTime**[2] = 3 is larger than t = 2. Thus, for this case, Line 12 sets variable **notValid** to TRUE and the invalid t-arborescence is discarded. Repeating the above steps, we obtain all tv-arborescences as shown in Table 4.1 with $|\mathrm{TVA}_1| = 2$, $|\mathrm{TVA}_2| = 4$, $|\mathrm{TVA}_3| = 3$ and $|\mathrm{TVA}_4| = 0$. Thus, in this example, a total of nine out of 16 t-arborescences are valid. Note that as $|\mathrm{TVA}_4|$ = 0, root node v_4 would not be able to broadcast packets.

Proposition 4.6: The time complexity of Method 1 is $O(|E||\mathrm{TA}_{All}|)$.

Proof: Step ***1)*** of Method 1 generates $G''(V, E'')$ from $G'(V, E^{TS})$ in $O\left(\sum_{e_{i,j} \in E} |TS(i,j)|\right)$ or $O(|E||TS|)$, where $|TS| = \max(|TS(i,j)|)$ for all $e_{i,j} \in E$. The time complexity of Step ***2)*** of Method 1 depends on $|\mathrm{TA}_j|$, for each root v_j. Further, the time complexity to obtain the determinant of matrix $\hat{L}_j^{+}$ depends not only on its size, but also on its entries. Recall that matrix $\hat{L}_j^{+}$ of size $(|V| - 1) \times (|V| - 1)$ will have $(|V| - 1)!$ summands. A summand say $m_{11}m_{22}\ldots m_{(|V|-1)(|V|-1)}$ represents one or more entries at locations $m_{11}, m_{22}, \ldots, m_{(|V|-1)(|V|-1)}$ of matrix $\hat{L}_j^{+}$. Note that the number of entries at each location of matrix $\hat{L}_j^{+}$ is affected by the number of timestamp(s) over a respective edge. Specifically, when edges have more timestamps, there would be more entries at different

locations of matrix $\hat{L}_j^+$. This results in, possibly, a polynomial comprising of a large number of monomials, *i.e.*, t-arborescences, from the determinant of matrix $\hat{L}_j^+$. Thus, due to the non-deterministic nature of the solution, we opt to represent the time complexity in terms of the total number of t-arborescences generated. Let σ be the time required to generate each t-arborescence TA_j^i. Thus, the time complexity of Step ***2)*** of Method 1 is $O(\sigma|\mathrm{TA}_j|)$ for each root v_j. One can use a depth first search algorithm (Leiserson et al. 2001) to generate an arborescence and assign timestamp to its each edge in time $O(|E|+|V|)$. Therefore, in total, Step ***2)*** takes $O((|E|+|V|)|\mathrm{TA}_{All}|)$. In Step ***3)***, *i.e.*, Algorithm 4.1, for each TA_j^i, Lines 7-9 and Lines 10-15 each takes $O(|V|)$. Lines 16-18 take $O(1)$. Line 4 repeats Lines 5-18 $|\mathrm{TA}_j|$ times. Thus, the time complexity of the loop in Lines 4-19, for checking the validity of all $\mathrm{TA}_j^i \in \mathrm{TA}_j$ is $O(|\mathrm{TA}_j||V|)$. Line 2 repeats Lines 3-20 for each of the $|V|$ root nodes, and thus the time complexity of Lines 2-21 is $O((\sum_{j=1}^{|V|}|\mathrm{TA}_j|)|V|) = O(|\mathrm{TA}_{All}||V|)$ because $|\mathrm{TA}_{All}| = \sum_{j=1}^{|V|}|\mathrm{TA}_j|$. Therefore, the time complexity of Method 1 is $O(|E||TS|+(|E|+|V|)|\mathrm{TA}_{All}|+|\mathrm{TA}_{All}||V|)$. Note that $|\mathrm{TA}_{All}| > |TS|$ and in general $|E| \geq |V|$; hence, the time complexity is reduced to $O(|E||\mathrm{TA}_{All}|)$. ■

As shown in Proposition 4.6, the time complexity of Method 1 is dominated by the complexity of its Step ***2)***, *i.e.*, to generate all t-arborescences.

4.3.3 Method 2

Different from Method 1, Method 2 is a two-step approach that generates all tv-arborescences from their corresponding arborescences, for each root node $v_j \in V$. It has the following two steps:

1) ***Enumerate A_{All} from $G(V,E)$*:** We apply Tutte's Matrix-Tree Theorem on the directed graph $G(V,E)$ of a TAG $G'(V,E^{TS})$ to enumerate all arborescences $\mathrm{A}_j^i \in \mathrm{A}_j$ for all root node $v_j \in V$. We use Tutte's Matrix-Tree Theorem in the same manner as in Method 1. Note that any algorithm to generate all arborescences, *e.g.*, in (Gabow and Myers

1978; Hariharan et al. 1995) can also be used here.

2) Generate TVA_{All} from A_{All}: This step obtains tv-arborescences from all arborescences generated in Step ***1)***. This step is accomplished by Algorithm 4.2.

To explain Algorithm 4.2, we first define order of *active time* or *availabilit*y between tv-arborescences. Consider two tv-arborescences TVA_j^c and TVA_j^d that are generated from arborescence A_j^i. Thus, for each edge $e_{a,b} \in \text{A}_j^i$, tv-arborescences TVA_j^c and TVA_j^d contain TSEs $e_{a,b}^{t_1}$ and $e_{a,b}^{t_2}$, respectively, for any timestamps $t_1, t_2 \in TS(a,b)$. We define tv-arborescence TVA_j^c as active or available *earlier* than TVA_j^d, denoted by $\text{TVA}_j^c < \text{TVA}_j^d$, if each TSE $e_{a,b}^{t_1} \in \text{TVA}_j^c$ is active no later than its corresponding TSE $e_{a,b}^{t_2} \in \text{TVA}_j^d$, *i.e.*, $t_1 \leq t_2$. Further, we call TVA_j^α and TVA_j^β, generated from arborescence A_j^i, as the *earliest* and the *latest* tv-arborescence, respectively, if we have $\text{TVA}_j^\alpha < \text{TVA}_j^z$ and $\text{TVA}_j^z < \text{TVA}_j^\beta$, for any other tv-arborescence TVA_j^z generated from A_j^i. For example, in Figure 2.2, for node v_2, the arborescence in Figure 2.2(i) is the earliest and that in Figure 2.2(iv) is the latest tv-arborescence. The concept of the earliest and the latest tv-arborescence prevents Algorithm 4.2 from assigning all possible timestamps in $TS(a,b)$ to each edge $e_{a,b}$ of arborescence $\text{A}_j^i \in \text{A}_j$, and hence reduces its time complexity. Thus, in essence Algorithm 4.2 discards those TSE(s), corresponding to each edge $e_{a,b} \in \text{A}_j^i$, which do not lead to any tv-arborescence. More specifically, Algorithm 4.2 first generates the earliest and the latest tv-arborescences for each arborescence A_j^i. Then, it enumerates all tv-arborescences whose order of availabilities are between the earliest and latest. The details of Algorithm 4.2 are as follows:

Line 1 of Algorithm 4.2 creates an array TVA_{All} to store TVA_j of all root node $v_j \in V$. For each root node v_j, Lines 2-15 generate a set TVA_j from A_j. For each arborescence $\text{A}_j^i \in \text{A}_j$, Line 5 sorts its edges in increasing order of their distance from the root v_j. Line 6 uses **Gen_MinValues**( ), shown in Algorithm 4.3, to generate array **minValues** that stores the

timestamps of the earliest tv-arborescence, *i.e.*, TVA_j^{α}, for arborescence A_j^i. For each entry $b = 1, 2, \ldots, |V|$ of **minValues**, Algorithm 4.3 replaces it with the *minimum* timestamp from $TS(a, b)$ corresponding to edge $e_{a,b} \in \text{A}_j^i$ such that all edges are time-ordered. In particular, for each arborescence $\text{A}_j^i \in \text{A}_j$, edge $e_{a,b}$ is not time-ordered when **minValues**[b] = 0 for any node $v_b \neq v_j$. Note that we always have **minValues**[j] = 0 for each root node v_j. Thus, Lines 7-9 discard arborescence A_j^i if **minValues** contains more than one entry with value equal to zero, as it will not produce any tv-arborescence $\text{TVA}_j^i \in \text{TVA}_j$. Otherwise, generated **minValues** contains the timestamps of the earliest tv-arborescence corresponding to A_j^i. Line 10 generates array **maxValues** using **Gen_MaxValues**( ), shown in Algorithm 4.4 and 4.5.

Algorithm 4.2. Gen_TVA$_{All}$fromA$_{All}$

Input: A_{All}
Output: TVA_{All}

1. $\text{TVA}_{All} \leftarrow \{\}$
2. **for** ($j \leftarrow$1 to $|V|$)
3. $\text{TVA}_j \leftarrow \{\}$
4. **for** ($i \leftarrow$ 1 to $|\text{A}_j|$)
5. Sort the edges of A_j^i in increasing order of distance from root node v_j
6. **minValues** ← **Gen_MinValues**(A_j^i)
7. **if minValues** contains more than one entry with zero values
8. Repeat Line 4 for the next arborescence A_j^i
9. **end if**
10. **maxValues** ← **Gen_MaxValues**(A_j^i)
11. **TSE_List** ← **Gen_TSE_List**(A_j^i, **minValues**, **maxValues**)
12. Generate all tv-arborescences from **TSE_List** and insert them into TVA_j
13. **end for**
14. Insert TVA_j into TVA_{All}
15. **end for**

Unlike **minValues**, for each entry b = 1, 2, …, $|V|$ of **maxValues**, Algorithm 4.4 and 4.5 replace its entry with the *maximum* timestamp from $TS(a, b)$ corresponding to edge $e_{a,b} \in A_j^i$ such that all edges are time-ordered. Thus, **maxValues** stores the timestamps of the latest tv-arborescence corresponding to A_j^i, *i.e.*, TVA_j^β. Line 11 of Algorithm 4.2 uses function **Gen_TSE_List**( ), shown in Algorithm 4.6, to generate the array **TSE_List** for arborescence A_j^i using **minValues** and **maxValues**. More specifically, for arborescence A_j^i, each entry of **TSE_List** denotes a set of TSE(s) generated from the range of timestamps provided by each entry $b \neq j$ of arrays **minValues** and **maxValues**. Thus, all other timestamps not lying in this range can be discarded as they will not render any tv-arborescence. Therefore, **minValues**, **maxValues** and **TSE_List** together help to reduce the time complexity of Method 2. Finally, Line 12 uses **TSE_List** to generate all tv-arborescences corresponding to arborescence A_j^i, and stores them in TVA_j. Lines 3-13 of Algorithm 4.2 are repeated for each node v_j, and Line 14 stores all generated tv-arborescences.

Now, we describe how function **Gen_MinValues**( ), in Algorithm 4.3, generates the array **minValues** for each arborescence A_j^i. Line 1 initializes each entry of **minValues** to zero. Lines 2-14 use each edge $e_{a,b} \in A_j^i$, sorted in increasing order of distance from root v_j, to set **minValues**[b]. Specifically, Lines 3-4 consider each edge $e_{j,b}$, *i.e.*, all edges from root v_j, and set **minValues**[b] with the smallest value in $TS(j, b)$, *i.e.*, $TS(j,b,1)$. For all other edges, Lines 6-13 compute the remaining entries of **minValues** as follows. Consider a directed edge $e_{a,b}$. If there is *no* timestamp $x \in TS(a, b)$ that has a value at least equal to **minValues**[a], then **minValues**[b] remains zero. Otherwise, we set **minValues**[b] = x, where x is the smallest timestamp in $TS(a, b)$ at some index k, *i.e.*, $TS(a,b,k)$, with value at least equal to **minValues**[a]. Note that this is done to ensure that any two edges in sequence are time-ordered. As an example, for **minValues**[a] = 3 and $TS(a, b)$ = {1, 3, 6}, at index k = 2 of $TS(a, b)$ we have x = 3 which is equal to **minValues**[a], and thus **minValues**[b] = 3. On the other hand, for $TS(a, b)$ = {1, 2}, we have **minValues**[b] = 0, signifying that edge $e_{a,b}$ is not time-ordered.

Algorithm 4.3. Gen_MinValues

Input: A_j^i

Output: minValues

```
1.  minValues ← [0, ..., 0]
2.  for each e_{a,b} ∈ A_j^i
3.    if (a = j)
4.      minValues[b] = TS(a,b,1)
5.    else
6.      k ← 1
7.     while ((minValues[a] > TS(a,b,k) && (k < |TS(a,b)|))
8.       k ++
9.     end while
10.    if (TS(a,b,k) ≥ minValues[a])
11.      minValues[b] ← TS(a,b,k)
12.    end if
13.  end if
14. end for
```

Next, we show how function **Gen_MaxValues**( ) in Algorithm 4.4 generates array **maxValues**. Line 1 initializes each entry of arrays **maxValues** and **parent**[1, … , $|V|$) to zero. The array **parent** stores *parent* of each node. Then, the function generates **maxValues** in two stages. For each edge $e_{a,b} \in A_j^i$, Stage-1 sets **maxValues**[b] with the maximum entry in $TS(a, b)$; see Lines 2-5. Note that the content of **maxValues**[j] remains zero. Lines 2-5 also set each entry b of **parent** to a, *i.e.*, **parent**[b] = a. Note that **parent**[j] = 0. Stage-2, shown in Lines 6-10, searches **maxValues** for any edge $e_{a,b}$ that is not time-ordered. Edge $e_{a,b}$ is not time-ordered if **maxValues**[b], *i.e.*, the timestamp value at $TS(a, b, |TS(a, b)|)$ is less than **maxValues**[a]. For each such edge, function **Update**( ), shown in Algorithm 4.5, backtracks and revises entries in **maxValues** such that all edges become time-ordered. More specifically, it is important that **maxValues**[b] contains the maximum timestamp $x \in TS(a, b)$ such that $x \geq$ **maxValues**[a]. This requirement ensures edge $e_{a,b}$ is active no later than the activation of its preceding edge $e_{\textbf{parent}[a],a}$. Consider **maxValues**[b] = 2 and $TS($**parent**$[a], a)$ = {1, 3, 4}. After Stage-1, **maxValues**[a] contains the maximum

timestamp in $TS(\mathbf{parent}[a], a)$, *i.e.*, **maxValues**$[a]$ = 4. Note that edge $e_{a,b}$ is activated at slot 2, which is earlier than the timestamp for its predecessor edge $e_{\mathbf{parent}[a],a}$ that is available at time 4. For this example, if array **maxValues** filled in Stage-1 is used as such, edge $e_{a,b}$ will not be time-ordered with respect to edge $e_{\mathbf{parent}[a],a}$. Function **Update**( ) addresses this problem, *i.e.*, it sets **maxValues**$[a]$ with timestamp in $TS(\mathbf{parent}[a], a)$ which is no larger than **maxValues**$[b]$. In general, for each entry that is not time-ordered, Stage-2 backtracks and updates, when necessary, the entry in **maxValues** for the preceding edges up to the root node; this task is implemented recursively by **Update**( ) called in Line 8 of Algorithm 4.4.

Algorithm 4.4. Gen_MaxValues

Input: A_j^i

Output: maxValues

1. **maxValues** $\leftarrow [0, \ldots, 0]$ and **parent** $\leftarrow [0, \ldots, 0]$
2. **for** each $e_{a,b} \in \mathrm{A}_j^i$
3. **parent**$[b] \leftarrow a$
4. **maxValues**$[b] \leftarrow TS(a,b,|TS(a,b)|)$
5. **end for**
6. **for** each $e_{a,b} \in \mathrm{A}_j^i$
7. **if** $(TS(a,b,|TS(a,b)|) <$ **maxValues**$[a])$
8. Call **Update**(a, b, **parent**, **maxValues**)
9. **end if**
10. **end for**

The exit condition of **Update**( ) is invoked when it reaches the root node while backtracking and updating the timestamps in **maxValues**; see Lines 1-3 of Algorithm 4.5. Lines 4-7 of Algorithm 4.5 aim to find a suitable timestamp from $TS(\mathbf{parent}[a], a)$ for edge $e_{\mathbf{parent}[a],a}$, so as to make its non-time ordered successive edge $e_{a,b}$ time-ordered. More specifically, Line 4 initializes M as index to the largest timestamp in $TS(\mathbf{parent}[a], a)$ available to edge $e_{\mathbf{parent}[a],a}$. The loop in Lines 5-7 is to find the largest timestamp in $TS(\mathbf{parent}[a], a)$ that is less than or equal to **maxValues**$[b]$. Then, Line 8 sets **maxValues**$[a]$ to the found timestamp, *i.e.*, $TS(\mathbf{parent}[a], a)$ at index M. Note that

$TS(\textbf{parent}[a], a)$ always contains $TS(\textbf{parent}[a], a, M) \leq \textbf{maxValues}[b]$, for some $M > 0$ because we reach this stage only after analyzing **minValues**. Thus, by now we are sure that arborescence A_j^i being considered will produce at least one tv-arborescence.

Algorithm 4.5. Update

Input: a, b, **parent**, **maxValues**
Output: maxValues

1. **if** (**parent**$[a] = 0$)
2. **exit** algorithm
3. **end if**
4. $M \leftarrow |TS(\textbf{parent}[a],a)|$
5. **while** (**maxValues**$[b] < TS(\textbf{parent}[a],a,M)$)
6. $M--$
7. **end while**
8. **maxValues**$[a] \leftarrow TS(\textbf{parent}[a],a,M)$
9. **Update** (**parent**$[a]$, a, **parent**, **maxValues**)

More specifically, in worst case **Update**( ) will produce **maxValues** that has the same contents as **minValues**. Hence, in worst case, an arborescence under consideration generates only one tv-arborescence. The process of backtracking and update is repeated for all preceding edges of $e_{a,b}$ until it reaches root node v_j to make them all time-ordered; see Line 9 of Algorithm 4.5. For example, in Stage-2, for **maxValues**$[b]$ = 2 and $TS(\textbf{parent}[a], a)$ = {1, 3, 4}, we initially have M = 3 and **maxValues**$[a]$ = 4. As we have **maxValues**$[b]$ = 2 < **maxValues**$[a]$ = $TS(\textbf{parent}[a], a, M) =$ 4, value of M is decremented by one, thereby making M = 2; see Lines 5-7 of Algorithm 4.5. Observe that $TS(\textbf{parent}[a], a, 2)$ = 3. Note that if we set **maxValues**$[a]$ = $TS(\textbf{parent}[a], a, 2)$ = 3, the timestamp value will still be larger than **maxValues**$[b]$ = 2. So, we further decrease M to select a suitable timestamp value for **maxValues**$[a]$. For M = 1, we achieve $TS(\textbf{parent}[a], a, 1)$ = 1 ≤ **maxValues**$[b]$ = 2. Thus, we set **maxValues**$[a]$ = 1 which is smaller than **maxValues**$[b]$ = 2 and hence assure that both edges $e_{\textbf{parent}[a],a}$ and $e_{a,b}$ are now time-ordered. Note that for edge $e_{\textbf{parent}[a],a}$, we can discard timestamps 3 and 4 because among three possible combinations of timestamps, *viz.*, {1-2}, {3-2} and {4-2} of

edges $e_{\textbf{parent}[a],a}$ and $e_{a,b}$, only {1-2} makes the two edges time-ordered. The other two combinations make the edges not time-ordered because edge $e_{\textbf{parent}[a],a}$ is active later than $e_{a,b}$. Now, if $TS(\textbf{parent}[a], a)$ = {1, 2} and **maxValues**[b] = 2, we initially have **maxValues**[a] = 2. In this case as **maxValues**[b] = **maxValues**[a], we do not require any amendment in the entries of array **maxValues**.

At this juncture, we discuss how function **Gen_TSE_List**( ), shown in Algorithm 4.6, generates array **TSE_List** from **minValues** and **maxValues** for arborescence A_j^i. Each entry **TSE_List**[k], for k = 1, 2, …, ($|V|-1$) contains a set of TSE(s) for the k^{th} edge of A_j^i; initially, each entry is set to NULL. More specifically, for each edge $e_{a,b} \in A_j^i$, Lines 2-9 of Algorithm 4.6 generate all TSEs $e_{a,b}^t$, for all $t \in TS(a,b)$, such that timestamp t lies in the range of timestamps between **minValues** and **maxValues** at index b, *i.e.*, **minValues**[b] $\leq t \leq$ **maxValues**[b].

Algorithm 4.6. Gen_TSE_List

Input: A_j^i, **minValues**, **maxValues**
Output: TSE_List

```
1.  TSE_List ← {}
2.  for each e_{a,b} ∈ A_j^i
3.      TSE ← {}
4.      for (r ← 1 to |TS(a,b)|)
5.        t ← TS(a,b,r)
6.        if (t ≥ minValues[b] && t ≤ maxValues[b])
7.            Insert e_{a,b}^t to TSE
8.        end if
9.      end for
10.     Insert TSE into TSE_List
11. end for
```

For example, Algorithm 4.6 generates three TSEs *viz*., $e_{a,b}^2$, $e_{a,b}^3$ and $e_{a,b}^4$ for edge $e_{a,b}$ with timestamps {2, 3, 4}, **minValues**[b] = 2 and **maxValues**[b] = 4. However, if the same edge has timestamps {1, 3} and if **minValues**[b] = 1 and **maxValues**[b] = 1, then Algorithm 4.6 creates only $e_{a,b}^1$, thereby avoiding the formation of $e_{a,b}^3$ that will never be

used in generating tv-arborescences. Line 10 stores all TSEs, generated from each edge $e_{a,b} \in \mathrm{A}_j^i$, in **TSE_List**.

After obtaining **TSE_List**, Line 12 of Algorithm 4.2 uses TSEs in **TSE_List** to generate all tv-arborescences corresponding to the arborescence A_j^i and stores them in TVA_j. More specifically, we need to generate combinations of TSEs from **TSE_List** by discarding, if any, combinations which are not time-ordered. We explain this step using an example. Consider a **TSE_List** = $[\{e_{1,2}^1\},\ \{e_{2,3}^1\},\ \{e_{2,4}^4\},\ \{e_{3,5}^1, e_{3,5}^3\},\ \{e_{5,6}^2, e_{5,6}^3\}]$ for an i^{th} arborescence from root v_1; Figure 4.2, shows its arborescence representation. Using the elements in **TSE_List**, we first find all TS-MPS from the root node v_1 to each leaf node. Let L_i be a set of leaf nodes in **TSE_List**, and $\text{TS-MPS}_{j,d}$ be a set of TS-MPS from the root v_j to leaf node v_d. Thus, for Figure 4.2, we have $L_i = \{v_4, v_6\}$. From root node v_1, there is only one path to leaf node v_4, *i.e.*, $(e_{1,2}^1.e_{2,4}^4)$, and the path is time-ordered. Therefore, we have $\text{TS-MPS}_{1,4} = (e_{1,2}^1.e_{2,4}^4)$. On the other hand, there are four paths to leaf node v_6, *viz.*, $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^1.e_{5,6}^2)$, $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^1.e_{5,6}^3)$, $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^3.e_{5,6}^3)$, and $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^3.e_{5,6}^2)$, but only the first three paths are TS-MPS. Thus, we have $\text{TS-MPS}_{1,6} = \{(e_{1,2}^1.e_{2,3}^1.e_{3,5}^1.e_{5,6}^2)$, $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^1.e_{5,6}^3)$, $(e_{1,2}^1.e_{2,3}^1.e_{3,5}^3.e_{5,6}^3)\}$ and we discard the fourth path.

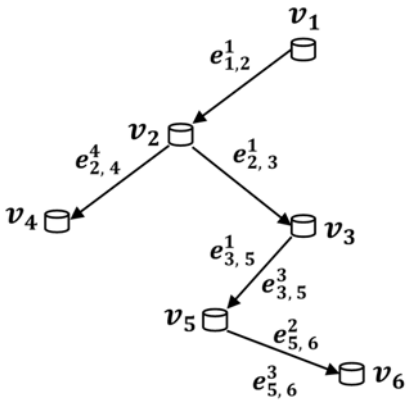


Figure 4.2. A pictorial representation of TSE_List entries.

Next, to enumerate all tv-arborescences corresponding to the i^{th} arborescence from root node v_1, we generate all combinations, having unique elements, of TS-MPS from root

node v_1 to the two leaves. Since |TS-MPS$_{1,4}$| = 1 and |TS-MPS$_{1,6}$| = 3, we obtain $1 \times 3 = 3$ tv-arborescences from root v_1. More specifically, corresponding to i^{th} arborescence, the three tv-arborescences to be added to TVA$_1$ are $(e^1_{1,2}.e^1_{2,3}.e^1_{3,5}.e^2_{5,6}.e^4_{2,4})$, $(e^1_{1,2}.e^1_{2,3}.e^1_{3,5}.e^3_{5,6}.e^4_{2,4})$ and $(e^1_{1,2}.e^1_{2,3}.e^3_{3,5}.e^3_{5,6}.e^4_{2,4})$. In general, for the **TSE_List** derived from A^i_j, with $|L_i|$ leaves, the number of tv-arborescences is given by $\prod_{\forall v_d \in L_i} |\text{TS-MPS}_{j,d}|$. Here i varies from 1 to $|A_j|$. In other words, there can be $|A_j|$ number of A^i_j from root node v_j. Thus, $|TVA_j|$ is given as

$$|\text{TVA}_j| = \sum_{\forall\, A^i_j \in A_j} \left(\prod_{\forall v_d \in L_i} |\text{TS-MPS}_{j,d}|\right) \tag{4.3}$$

For more clarity of the two steps used in Method 2, illustration 4.2 highlights each step in detail.

Illustration 4.2: Consider TAG in Figure 2.1(ii). A directed graph $G(V, E)$ of this TAG is a graph with no timestamps. We generate the Laplace matrix $L^+(G)$ for graph $G(V, E)$ as

$$L^+(G) = D^+ - \mathcal{A} = \begin{bmatrix} e_{2,1} & -e_{1,2} & 0 & 0 \\ -e_{2,1} & (e_{1,2} + e_{3,2}) & -e_{2,3} & 0 \\ 0 & -e_{3,2} & (e_{4,3} + e_{2,3}) & -e_{3,4} \\ 0 & 0 & -e_{4,3} & e_{3,4} \end{bmatrix}$$

In Step ***1)***, we generate $\hat{L}^+_j$ for all $v_j \in V$ from $L^+(G)$ and compute their determinant to find A_{All}. Table 4.2 collates $|A_{All}| = 4$ arborescences of $G(V, E)$. For Step ***2)***, consider the arborescence from node v_1 *viz.*, $A^1_1 = (e_{1,2}.e_{2,3}.e_{3,4})$ wherein the timestamps associated with $e_{1,2}$, $e_{2,3}$, and $e_{3,4}$ are {1, 3}, {1, 2} and {4}, respectively. From the timestamps of each edge, Line 6 of Algorithm 4.2 utilizes Algorithm 4.3 to generate **minValues** = (0, 1, 1, 4). As there exists only one zero, there will be at least one tv-arborescence from node v_1. This condition is checked in Lines 7-9 of Algorithm 4.2. Then, Line 10 of Algorithm 4.2 uses Algorithms 4.4 and 4.5 to generate **maxValues** = (0, 1, 2, 4). Next, Line 11 of Algorithm 4.2 uses Algorithm 4.6 to obtain **TSE_List**. Lines 2-9 of Algorithm 4.6 generate TSEs from

each edge of A_1^1. For example, a TSE generated from edge $e_{1,2}$ is $e_{1,2}^1$. On the other hand, from edge $e_{2,3}$, Lines 2-9 obtain two TSEs, *viz*., $e_{2,3}^1$ and $e_{2,3}^2$. Finally, TSE $e_{3,4}^4$ is generated from edge $e_{3,4}$. Thus, we have **TSE_List** = [$\{e_{1,2}^1\}$, $\{e_{2,3}^1, e_{2,3}^2\}$, $\{e_{3,4}^4\}$]. Observe from the entries in **TSE_List** that v_4 is the only leaf node with respect to root v_1. Hence, Line 12 of Algorithm 4.2 finds TS-MPS$_{1,4}$ = $\{(e_{1,2}^1.e_{2,3}^1.e_{3,4}^4), (e_{1,2}^1.e_{2,3}^2.e_{3,4}^4)\}$ that represents two tv-arborescences from A_1^1, and put them in TVA$_1$. As there is only one arborescence from root v_1, Lines 4-13 of Algorithm 4.2 run only once. Lines 2-15 of Algorithm 4.2 are then repeated for other root nodes.

Table 4.2. Timestamped arborescences of the TAG of Figure 2.1(ii) using Method 2.

$\hat{\boldsymbol{L}}_{\mathbf{1}}^{+} = \begin{bmatrix} (e_{1,2}+e_{3,2}) & -e_{2,3} & 0 \\ -e_{3,2} & (e_{4,3}+e_{2,3}) & -e_{3,4} \\ 0 & -e_{4,3} & e_{3,4} \end{bmatrix}$	
$\mathbf{A_1} = \{(e_{1,2}.e_{2,3}.e_{3,4})\}$ **Timestamps** = {1,3}-{1,2}-{4}	$\mathbf{TVA_1}$ = $\{(e_{1,2}^1.e_{2,3}^1.e_{3,4}^4), (e_{1,2}^1.e_{2,3}^2.e_{3,4}^4)\}$
$\hat{\boldsymbol{L}}_{\mathbf{2}}^{+} = \begin{bmatrix} e_{2,1} & 0 & 0 \\ 0 & (e_{4,3}+e_{2,3}) & -e_{3,4} \\ 0 & -e_{4,3} & e_{3,4} \end{bmatrix}$	
$\mathbf{A_2} = \{(e_{2,1}.e_{2,3}.e_{3,4})\}$ **Timestamps** = {1,3}-{1,2}-{4}	$\mathbf{TVA_2}$ = $\{(e_{2,1}^1.e_{2,3}^1.e_{3,4}^4), (e_{2,1}^1.e_{2,3}^2.e_{3,4}^4), (e_{2,1}^3.e_{2,3}^1.e_{3,4}^4), (e_{2,1}^3.e_{2,3}^2.e_{3,4}^4)\}$
$\hat{\boldsymbol{L}}_{\mathbf{3}}^{+} = \begin{bmatrix} e_{2,1} & -e_{1,2} & 0 \\ -e_{2,1} & (e_{1,2}+e_{3,2}) & 0 \\ 0 & 0 & e_{3,4} \end{bmatrix}$	
$\mathbf{A_3} = \{(e_{3,4}.e_{3,2}.e_{2,1})\}$ **Timestamps** = {4}-{1,2}-{1,3}	$\mathbf{TVA_3}$ = $\{(e_{3,4}^4.e_{3,2}^1.e_{2,1}^1), (e_{3,4}^4.e_{3,2}^1.e_{2,1}^3), (e_{3,4}^4.e_{3,2}^2.e_{2,1}^3)\}$
$\hat{\boldsymbol{L}}_{\mathbf{4}}^{+} = \begin{bmatrix} e_{2,1} & -e_{1,2} & 0 \\ -e_{2,1} & (e_{1,2}+e_{3,2}) & -e_{2,3} \\ 0 & -e_{3,2} & (e_{4,3}+e_{2,3}) \end{bmatrix}$	
$\mathbf{A_4} = \{(e_{4,3}.e_{3,2}.e_{2,1})\}$ **Timestamps** = {4}-{1,2}-{1,3}	$\mathbf{TVA_4}$ = *NULL*

To illustrate the importance of using arrays **minValues**, **maxValues** and **TSE_List**, which prevent generation of some invalid t-arborescences, consider arborescence A_1^1 = $(e_{1,2}.e_{2,3}.e_{3,4})$ from Table 4.2. The timestamps associated with its edges $e_{1,2}$, $e_{2,3}$, and $e_{3,4}$

are {1, 3}, {1, 2} and {4}, respectively. Let us consider a version of Method 2, called Method 2' that assigns *all* possible timestamps to each edge. Thus, Method 2' would (i) enumerate *all* four possible combinations of timestamps, *i.e.*, {1-1-4}, {1-2-4}, {3-1-4} and {3-2-4} by taking one timestamp from each edge; (ii) generate all four corresponding t-arborescences *viz.*, $(e^1_{1,2}.e^1_{2,3}.e^4_{3,4})$, $(e^1_{1,2}.e^2_{2,3}.e^4_{3,4})$, $(e^3_{1,2}.e^1_{2,3}.e^4_{3,4})$ and $(e^3_{1,2}.e^2_{2,3}.e^4_{3,4})$, same as those obtained from Method 1 and shown in Table 4.1; and (iii) check these four t-arborescences to obtain two tv-arborescences in set TVA_1. In contrast, Method 2 uses **minValues**, **maxValues** and **TSE_List** to reduce the computational complexity of Method 2'. For example, as shown in Illustration 4.2, Method 2: (i) does not generate TSE $e^3_{1,2}$; (ii) avoids generating invalid t-arborescences $(e^3_{1,2}.e^1_{2,3}.e^4_{3,4})$ and $(e^3_{1,2}.e^2_{2,3}.e^4_{3,4})$ because they contain TSE $e^3_{1,2}$; and (iii) generates and checks less number of invalid t-arborescences, *i.e.*, zero in this example.

Proposition 4.7: The time complexity of Method 2 is $O(|\mathrm{A}_{All}||V||TS|^{|V|})$.

Proof: Step ***1)*** of Method 2 enumerates A_{All} using Tutte's Matrix-Tree Theorem. Let ρ be the time required to generate each arborescence. Thus, the time complexity to generate $|\mathrm{A}_{All}|$ arborescences is $O(\rho|\mathrm{A}_{All}|)$. One can use a depth first search algorithm (Leiserson et al. 2001) to generate each arborescence in $O(|E| + |V|)$, and thus the time complexity of Step ***1)*** is $O((|E| + |V|)|\mathrm{A}_{All}|)$.

Now we analyze the time complexity of Step ***2)*** of Method 2. Let $|TS|$ denote the maximum $|TS(i,j)|$ for each edge $e_{i,j} \in E$. In Algorithm 4.3, Lines 3-5 take $O(|V|)$, while Lines 6-13 require $O(|V||TS|)$ because for filling each entry in **minValues**, the algorithm in the worst case needs to check for $|TS|$ values. Thus, the time complexity to generate **minValues** is $O(|V||TS|)$.

For Stage-1, *i.e.*, Lines 2-5, Algorithm 4.4 requires $O(|V|)$. In Stage-2, *i.e.*, Lines 6-10, if each edge $e_{a,b} \in \mathrm{A}^i_j$ satisfies the condition in Line 7, Algorithm 4.5 will be called $(|V| - 1)$ times in Line 8 of Algorithm 4.4. Recall that Algorithm 4.5 in the worst case

backtracks and if necessary updates all preceding entries in **maxValues**. Besides, for each update in **maxValues**, Algorithm 4.5 in the worst case will check $|TS|$ timestamps. Thus, the time complexity for filling **maxValues** in Stage-2 is $O\left(\sum_{i=2}^{|V|}|TS|\,(i-1)\right) = O\left(|TS|\frac{|V|(|V|-1)}{2}\right) = O(|V|^2|TS|)$. Therefore, the total time complexity to obtain **maxValues** is $O(|V|^2|TS|)$.

Lines 4-9 of Algorithm 4.6 require $O(|TS|)$. The lines are repeated ($|V|-1$) times in Line 2, and thus the time complexity to generate **TSE_List** is $O(|V||TS|)$.

Line 12 of Algorithm 4.2 requires $O(|V||TS|^{|V|})$ to generate all tv-arborescences from **TSE_List** for each arborescence $\mathrm{A}_j^i \in \mathrm{A}_j$. The details are as follows. It takes $O(|L_i| \prod_{k=1}^{|V|-1} |\mathbf{TSE\_List}[k]|)$ to find all tv-arborescences from **TSE_List** for each A_j^i. Besides, $|L_i|$ cannot be larger than $|V|$ and the number of TSEs at **TSE_List**[k], *i.e.*, |**TSE_List**[k]| can be no larger than $|TS|$. So, the time complexity of this step is $O(|V| \prod_{k=1}^{|V|-1} |TS|) = O(|V||TS|^{|V|})$. Thus, for each A_j^i, the time complexity of Step ***2)*** in Method 2 is $O((|V||TS|) + (|V|^2|TS|) + (|V||TS|) + (|V||TS|^{|V|})) = O(|V||TS|^{|V|})$ because $|V| \leq |TS|^{|V|-1}$. Note that the total number of arborescences from root node v_j is $|\mathrm{A}_j|$. So, the time complexity of Step ***2)*** in Method 2 for enumerating all $\mathrm{TVA}_j^i \in \mathrm{TVA}_j$ from all $\mathrm{A}_j^i \in \mathrm{A}_j$ is $O(|\mathrm{A}_j||V||TS|^{|V|})$. Further, the time complexity to generate all tv-arborescences in the network, *i.e.*, for all A_j is $O(|\mathrm{A}_{All}||V||TS|^{|V|})$. Thus, the time complexity of Method 2 is $O((|E|+|V|)|\mathrm{A}_{All}| + |\mathrm{A}_{All}||V||TS|^{|V|})$. Note that $|E| \leq |V|^2$ and $|TS|^{|V|} \geq |V|$, and therefore the time complexity of Method 2 is $O(|\mathrm{A}_{All}||V||TS|^{|V|})$. ■

As shown in Proposition 4.7, the time complexity of Method 2 is dominated by its Step ***2)***, *i.e.*, to generate tv-arborescences from arborescences.

At this juncture, it appears that the exponential number of t-arborescences generated by Method 1 is a complete waste of computational resources as Method 2 will directly enumerate all tv-arborescences from significantly lesser number of arborescences.

Nevertheless, the generated t-arborescences can be used to solve some other problems related to a TVCN, *e.g.*, optimal design, upgrade, and management of a TVCN topology. These topics are taken up separately in detail in Chapter 6. Another important application of the enumerated t-arborescences rooted at node v_s can be to enumerate all (s, d) TS-MPS. Recall that Chapter 3 covers three other techniques of all TS-MPS enumeration. The description of the developed technique is as follows.

4.4 t-Arborescences based TS-MPS Enumeration

This section presents an algorithm to enumerate a set of all TS-MPS, denoted by TS-MPS$_{s,d}$, between a specified (s, d) node pair of a TVCN from set TA$_s$ for node $v_s \neq v_d$ and $\{v_s, v_d\} \in V$. Recall that TA$_s$ is a set of all t-arborescences originating from root node $v_s \in V$ in TAG $G'(V, E^{TS})$. Note that by definition, there is a timestamped path from a source node v_s to sink node v_d in a timestamped arborescence $\text{TA}_s^i \in \text{TA}_s$. Further, each TA_s^i contains at most one TS-MPS from source v_s to destination node v_d. Let $\text{TS-MPS}_{s,d}^i$ be the TS-MPS generated from TA_s^i. It is important to note here that the set TVA$_s$ should not be used for generating set TS-MPS$_{s,d}$. The reason is because all $\text{TA}_s^i \in \text{TA}_s$ may contain at least one TS-MPS between node pair (s, d), while there may be zero $\text{TVA}_s^i \in \text{TVA}_s$; thereby, making it impossible to generate any TS-MPS from TVA$_s$. Next, we present Proposition 4.8, which is used by our proposed Algorithm 4.7.

Proposition 4.8: A $\text{TS-MPS}_{s,d}^i \in \text{TS-MPS}_{s,d}$ can be generated from $\text{TA}_s^i \in \text{TA}_s$ with time complexity of $O(|V|)$.

Proof: We need two main steps to find a $\text{TS-MPS}_{s,d}^i$ from TA_s^i: (i) search for destination node v_d among $|V|$ nodes in TA_s^i, and (ii) check if all TSE(s) of the path from source v_s to destination node v_d are time-ordered. Using a linear search, one can find node v_d for step (i) in $O(|V|)$. For step (ii), in the worst case, the path contains at most $(|V| - 1)$ number of TSEs. Thus, time-ordering of TSEs can be validated by traversing each TSE

from v_s to v_d in TA_s^i, which requires $O(|V|)$. Hence, the time complexity to generate TS-$\text{MPS}_{s,d}^i$ from TA_s^i is $O(|V| + |V|) = O(|V|)$. ■

Note that a TA_s^i may not contain TS-$\text{MPS}_{s,d}^i$ when the path from v_s to v_d is not time-ordered. For example, consider generating TS-$\text{MPS}_{1,3}$ from the four t-arborescences in TA_1, shown in Table 4.1, *i.e.*, $\text{TA}_1^1 = (e_{1,2}^1.e_{2,3}^1.e_{3,4}^4)$, $\text{TA}_1^2 = (e_{1,2}^1.e_{2,3}^2.e_{3,4}^4)$, $\text{TA}_1^3 = (e_{1,2}^3.e_{2,3}^1.e_{3,4}^4)$ and $\text{TA}_1^4 = (e_{1,2}^3.e_{2,3}^2.e_{3,4}^4)$. The last two t-arborescences render invalid paths, *viz.*, $(e_{1,2}^3.e_{2,3}^1)$ and $(e_{1,2}^3.e_{2,3}^2)$, respectively. Thus, one cannot generate TS-$\text{MPS}_{1,3}^3$ and TS-$\text{MPS}_{1,3}^4$ from TA_1^3 and TA_1^4, respectively. On the other hand, we have TS-$\text{MPS}_{1,3}^2$ = $(e_{1,2}^1.e_{2,3}^2)$ because in this path TSE $e_{1,2}^1$ appears earlier than TSE $e_{2,3}^2$; thus, both TSEs are time-ordered. More specifically, as there are two TS-MPS between node pair (1, 3), *viz.*, TS-$\text{MPS}_{1,3}^1$ = $(e_{1,2}^1.e_{2,3}^1)$ and TS-$\text{MPS}_{1,3}^2$ = $(e_{1,2}^1.e_{2,3}^2)$; thus, |TS-$\text{MPS}_{1,3}$| = 2.

Proposition 4.8 can be used to find all TS-$\text{MPS}_{s,d}^i \in$ TS-$\text{MPS}_{s,d}$ using all $\text{TA}_s^i \in \text{TA}_s$. More specifically, we present the following Algorithm 4.7 to enumerate, for a given (s, d) node pair, all TS-$\text{MPS}_{s,d}^i \in$ TS-$\text{MPS}_{s,d}$ using all $\text{TA}_s^i \in \text{TA}_s$.

Algorithm 4.7. Gen_TS-MPS$_{s,d}$_From_TA$_s$

Input: TA_s, destination node v_d

Output: TS-$\text{MPS}_{s,d}$

```
1.  TS-MPS_{s,d} ← { }
2.  for each TA_s^i ∈ TA_s
3.     if TA_s^i contains TS-MPS_{s,d}^i from v_s to v_d
4.        if TS-MPS_{s,d}^i ∉ TS-MPS_{s,d}
5.           Insert TS-MPS_{s,d}^i into TS-MPS_{s,d}
6.        end if
7.     end if
8.  end for
```

In Line 1, Algorithm 4.7 creates an empty set TS-MPS$_{s,d}$ to store all (s, d) TS-MPS. Lines 2-8 iterate over each $\text{TA}_s^i \in \text{TA}_s$ in order to find all (s, d) TS-MPS. Line 3 uses Proposition 4.8 to find a (s, d) TS-MPS from each TA_s^i. It is worth mentioning here that there can be many duplicate TS-MPS, which may appear from different $\text{TA}_s^i \in \text{TA}_s$. For example, consider generating TS-MPS between node pair (2, 4) of the TAG of Figure 2.1(ii) from arborescences $\text{TA}_2^1 = \left(e_{2,1}^1 . e_{2,3}^1 . e_{3,4}^4\right)$ and $\text{TA}_2^3 = (e_{2,1}^3 . e_{2,3}^1 . e_{3,4}^4)$ rooted at node v_2; see Table 4.1. Then, Line 3 would generate TS-MPS$_{2,4}^1$ = $(e_{2,3}^1 . e_{3,4}^4)$ and TS-MPS$_{2,4}^3$ = $(e_{2,3}^1 . e_{3,4}^4)$, respectively, from the two arborescences. Notice that the generated TS-MPS are identical to each other, *i.e.*, are duplicate. Line 4 checks for duplicating TS-MPS, and Line 5 puts each unique TS-MPS into TS-MPS$_{s,d}$. We use the following illustration to show steps of the algorithm.

Illustration 4.3: Let us find all TS-MPS between node pair (3, 1) of the TAG shown in Figure 2.1(ii). Here source node is v_3, while destination node is v_1. From Table 4.1, $|\text{TA}_3| = 4$, *viz.*, $\text{TA}_3^1 = (e_{3,4}^4 . e_{3,2}^1 . e_{2,1}^1)$, $\text{TA}_3^2 = (e_{3,4}^4 . e_{3,2}^2 . e_{2,1}^1)$, $\text{TA}_3^3 = (e_{3,4}^4 . e_{3,2}^1 . e_{2,1}^3)$ and $\text{TA}_3^4 = (e_{3,4}^4 . e_{3,2}^2 . e_{2,1}^3)$. The algorithm begins with $\text{TA}_3^1 = (e_{3,4}^4 . e_{3,2}^1 . e_{2,1}^1)$; see Line 2. Line 3 finds a $(3, 1)$ TS-MPS, *viz.*, TS-MPS$_{3,1}^1$ = $(e_{3,2}^1 . e_{2,1}^1)$ in $O(|V|)$; see Proposition 4.8. As initially TS-MPS$_{3,1}$ is empty, so Lines 4-6 add TS-MPS$_{3,1}^1$ to TS-MPS$_{3,1}$. Thus, now TS-MPS$_{3,1}$ = $\{(e_{3,2}^1 . e_{2,1}^1)\}$. Next, the algorithm's Line 2 uses $\text{TA}_3^2 = (e_{3,4}^4 . e_{3,2}^2 . e_{2,1}^1)$. Using TA_3^2, Line 3 finds path $(e_{3,2}^2 . e_{2,1}^1)$, which is not time-ordered and hence discarded. So, the algorithm goes back to Line 2 to begin with $\text{TA}_3^3 = (e_{3,4}^4 . e_{3,2}^1 . e_{2,1}^3)$. From TA_3^3, Line 3 finds TS-MPS$_{3,1}^3$ = $(e_{3,2}^1 . e_{2,1}^3)$. The algorithm at this stage checks whether TS-MPS$_{3,1}^3$ is unique or the same TS-MPS already exists in TS-MPS$_{3,1}$. This means that if each TSE of TS-MPS$_{3,1}^3$ is same as of any TS-MPS already in set TS-MPS$_{3,1}$, then TS-MPS$_{3,1}^3$ will be considered as duplicate and will be discarded. For example, the only TS-MPS, up to this stage, in TS-MPS$_{3,1}$, *i.e.*, TS-MPS$_{3,1}^1$ = $(e_{3,2}^1 . e_{2,1}^1)$, is not same as TS-MPS$_{3,1}^3$ = $(e_{3,2}^1 . e_{2,1}^3)$. Thus, TS-MPS$_{3,1}$ is updated to store $\{(e_{3,2}^1 . e_{2,1}^1), (e_{3,2}^1 . e_{2,1}^3)\}$. At last, algorithm uses $\text{TA}_3^4 = (e_{3,4}^4 . e_{3,2}^2 . e_{2,1}^3)$. Line 3 finds TS-MPS$_{3,1}^4$ = $(e_{3,2}^2 . e_{2,1}^3)$ from TA_3^4. Now, it can be observed that TS-MPS$_{3,1}^4$ is unique and therefore Algorithm 4.7 updates TS-MPS$_{3,1}$ to store $\{(e_{3,2}^1 . e_{2,1}^1)$,

$(e^1_{3,2}.e^3_{2,1})$, $(e^2_{3,2}.e^3_{2,1})\}$. Recall that unlike static networks, $|\text{TS-MPS}_{3,1}| = 3$ is not equal to $|\text{TS-MPS}_{1,3}| = 2$ because of the presence of timestamps with edges. Now, let us see Proposition 4.9.

Proposition 4.9: Time complexity to generate set $\text{TS-MPS}_{s,d}$ from set TA_s is $O\left(\frac{|\text{TS-MPS}_{s,d}|(|\text{TS-MPS}_{s,d}|-1)}{2}(|V|)\right)$.

Proof: For each $\text{TA}^i_s \in \text{TA}_s$, Line 3 of Algorithm 4.7 takes $O(|V|)$; see Proposition 4.8. However, the TS-MPS generated in Line 3 needs to be further assessed to verify whether it is unique or duplicate. More specifically, the first TS-MPS is always unique and needs no comparison; thus, is added directly to $\text{TS-MPS}_{s,d}$. The TS-MPS generated next is compared with first TS-MPS stored in $\text{TS-MPS}_{s,d}$. Similarly, the third TS-MPS is compared with the first two, and so on. Note that each comparison needs element by element inspection, which in worst case requires $O(|V|)$. Thus, in total it takes $[0 + 1 + \ldots + (|\text{TS-MPS}_{s,d}| - 1)] \times (|V|)$ comparisons to obtain set $\text{TS-MPS}_{s,d}$. In other words, time complexity to obtain $\text{TS-MPS}_{s,d}$ is $O\left(\frac{|\text{TS-MPS}_{s,d}|(|\text{TS-MPS}_{s,d}|-1)}{2}(|V|)\right)$. ■

4.5 Simulation Results

All algorithms presented in this chapter have been implemented at the facility available at Curtin University, Perth, Australia by using Java code supported on JDK 8 and above. The tests were run on IntelliJ IDEA Community edition version 2018.3.6 running on PC with the following configurations: (1) Processor: Intel (R) Core (TM) i7-7700 CPU @ 3.60 GHz, (2) RAM: 16.00 GB, and (3) System Type: 64-bit Operating System, x64-based processor.

To analyze the performance of our proposed algorithms, we use the ten arbitrarily generated TAGs shown in Table 3.6. It is important to note here that the period of each TAG is assumed to be four slots. Besides, the number of nodes in the aforementioned TAGs vary between four and seven, $|E^{TS}|$ vary between five and twelve, and $|TS|$ vary between two and three.

The upcoming Section 4.5.1, first analyzes the performance of Method 1 against Method 2 and then collates the number of arborescences, t-arborescences and tv-arborescences for each example TAG. Further, in Section 4.5.2 we present the effect of increasing the number of iterations, time slots, and TVCN nodes on the number of arborescences, t-arborescences and tv-arborescences.

4.5.1 Analysis: Method 1 versus Method 2

In this section, we present the salient features of both methods. Table 4.3 shows that Method 1 and Method 2 produce the same number of TVA_{All} for each of the ten TAGs; see column 7. Recall that Method 1 and Method 2 generate TVA_{All} from TA_{All} and A_{All}, respectively. Column 5 shows $|\text{TA}_{All}|$ and column 6 gives $|\text{A}_{All}|$ for each TAG of Table 3.6. Table 4.3 also presents the CPU Time (in seconds) utilized by the two methods for enumerating TVA_{All}; see columns 8 and 9. Note that each CPU Time value is the average over ten runs. Table 4.3 also shows that the number of arborescences generated by Method 2 is significantly less than the number of t-arborescences generated by Method 1, *i.e.*, $|\text{A}_{All}| \ll |\text{TA}_{All}|$ for each TAG; *e.g.*, 2576 versus 89250 for TAG #9. The result supports our analysis in Section 4.2.

Table 4.3. Performance of Method 1 and Method 2 over ten arbitrary TAGs of Table 3.6.

TAG # (1)	$\|V\|$ (2)	$\|E^{TS}\|$ (3)	$\|TS\|$ (4)	$\|\text{TA}_{All}\|$ (5)	$\|\text{A}_{All}\|$ (6)	$\|\text{TVA}_{All}\|$ (7)	CPU Time (in Sec.)	
							Method 1 (8)	Method 2 (9)
1	5	7	3	1820	105	527	0.0783	0.0272
2	4	5	2	176	32	77	0.0513	0.0504
3	7	10	3	4725	441	396	1.5133	0.0940
4	5	8	2	1310	200	457	0.0210	0.0164
5	6	9	3	3234	330	641	0.2395	0.0301
6	6	9	3	7380	486	1246	0.5640	0.0562
7	7	9	3	4536	287	359	1.0286	0.0318
8	5	8	3	3305	225	1001	0.0840	0.0329
9	7	12	3	89250	2576	8699	196.4511	0.4308
10	4	6	2	204	64	97	0.0077	0.0062

Further, it can be observed that the CPU time of Method 1 is considerably longer in comparison to Method 2 for enumerating TVA_{All} of each TAG. As an example, consider TAG #9 with $|V| = 7$, $|E^{TS}| = 12$, $|TS| = 3$. Here, Method 1 requires 196.4511s to enumerate 8699 tv-arborescences. In contrast, Method 2 takes only 0.4308s to enumerate the same TVA_{All}, *i.e.*, Method 2 for this example is 456 times faster than Method 1.

We observe two main reasons for the difference in time complexities of the two methods. Firstly, Method 1 needs to transform a TAG into a directed multigraph wherein each edge denotes a TSE. As there can be a large number of TSEs in a TAG, Method 1 performs a large number of operations for computing the determinant of $\hat{L}_j^+$ for each root $v_j \in V$, *i.e.*, to generate all t-arborescences $\text{TA}_j^i \in \text{TA}_j$ for each $v_j \in V$ using Tutte's Matrix-Tree Theorem. Recall that the time complexity of Method 1 is dominated by Tutte's Matrix-Tree Theorem for generating TA_{All}. On the other hand, Method 2 applies Tutte's Matrix-Tree Theorem on the simple directed graph representation of each TAG to generate all arborescences $\text{A}_j^i \in \text{A}_j$ for each $v_j \in V$. Note that each summand in the determinant computed in Method 2 contains lesser number of terms as compared to that in Method 1. Thus, enumeration of all arborescences in Method 2 is faster as compared to enumeration of all t-arborescences in Method 1. As an example, for TAG #9, Method 2 took only 0.254s to obtain 2576 arborescences, while Method 1 took 168.2611s to enumerate 89250 t-arborescences. Secondly, as discussed in Section 4.3.3, Step ***2)*** of Method 2 utilizes arrays **minValues**, **maxValues** and **TSE_List** to prevent generating all possible combinations of timestamps that correspond to all t-arborescences. Thus, Method 2 prevents generation of some invalid t-arborescences. In contrast, Method 1 uses all combinations of timestamps to obtain TA_{All}, and filters them to generate TVA_{All}. We note that many of the generated t-arborescences are not tv-arborescences, and thus they increase the time complexity of Method 1. As an example, for TAG #9, Method 1 generates 89250 t-arborescences, but only 8699 of which are tv-arborescences.

While Method 2 is computationally more efficient as compared to Method 1 for generating TVA_{All}, it, however, does not generate all t-arborescences of a TVCN that might

be needed for TVCN topology management. As stated before, the generated t-arborescences can be used to solve some other problems related to a TVCN, *e.g.*, optimal design, upgrade, and management of a TVCN topology. Table 4.4 presents the number of arborescences $|A_j|$, t-arborescences $|TA_j|$, and tv-arborescences $|TVA_j|$ for all root node $v_j \in V$ and for each of the ten TAG examples. Table 4.4 also presents the total number of TSEs in the network, *i.e.*, $|E''|$. Observe that for each given TAG, we have the same $|A_j|$ and $|TA_j|$, respectively, for each $v_j \in V$. This occurred because we assume bidirectional edges and contacts. From Table 4.4 also notice that $\sum_{j=1}^{|V|}(|TVA_j|) = |TVA_{All}|$ listed in Table 4.3.

Table 4.4. Simulation results obtained from ten arbitrary TAGs of Table 3.6.

Metric	**Root node**						
	v_1	v_2	v_3	v_4	v_5	v_6	v_7
TAG #1: $\|A_j\| = 21$, $\|TA_j\| = 364$, $\|E''\| = 30$							
$\|TVA_j\|$	107	186	85	118	31	--	--
TAG #2: $\|A_j\| = 8$, $\|TA_j\| = 44$, $\|E''\| = 18$							
$\|TVA_j\|$	11	29	16	21	--	--	--
TAG #3: $\|A_j\| = 63$, $\|TA_j\| = 675$, $\|E''\| = 32$							
$\|TVA_j\|$	21	64	86	49	102	70	4
TAG #4: $\|A_j\| = 40$, $\|TA_j\| = 262$, $\|E''\| = 26$							
$\|TVA_j\|$	71	95	119	87	85	--	--
TAG #5: $\|A_j\| = 55$, $\|TA_j\| = 539$, $\|E''\| = 30$							
$\|TVA_j\|$	67	124	221	76	149	4	--
TAG #6: $\|A_j\| = 81$, $\|TA_j\| = 1230$, $\|E''\| = 32$							
$\|TVA_j\|$	70	319	217	191	354	95	--
TAG #7: $\|A_j\| = 41$, $\|TA_j\| = 648$, $\|E''\| = 30$							
$\|TVA_j\|$	60	94	21	54	56	57	17
TAG #8: $\|A_j\| = 45$, $\|TA_j\| = 661$, $\|E''\| = 32$							
$\|TVA_j\|$	184	175	285	134	223	--	--
TAG #9: $\|A_j\| = 368$, $\|TA_j\| = 12750$, $\|E''\| = 44$							
$\|TVA_j\|$	593	365	1944	1273	822	2589	1113
TAG #10: $\|A_j\| = 16$, $\|TA_j\| = 51$, $\|E''\| = 18$							
$\|TVA_j\|$	33	30	18	16	--	--	--

-- Not Applicable

4.5.2 Analysis of $|A_{All}|$, $|TA_{All}|$ and $|TVA_{All}|$

We consider a 2D-simulation region of (x_{min}, y_{min}) = (0, 0) and (x_{max}, y_{max}) = (100, 100), in distance units, for each TVCN. Within this region, all nodes are arbitrarily distributed, and random topologies are generated using Waxman's (Waxman 1988) edge probability formula, *i.e.*, $p(\{v_i, v_j\}) = \theta\, e^{\frac{-d(v_i,v_j)}{L\gamma}}$. Here, $d(v_i, v_j)$ is the Euclidean distance from node v_i to v_j, L is the maximum distance between two nodes, and $0 < \gamma, \theta \leq 1$. Note that larger values of θ result in graphs having higher densities, while small values of γ increase the density of short edges relative to longer ones. Let each generated topology act as a snapshot of a TVCN. Thus, for a fixed number of nodes and period τ, there are τ randomly generated topologies, which are then aggregated to form a TAG. Finally, Method 1 and Method 2 are used to find average (Avg.) and standard deviation (SD) of $|A_{All}|$, $|TA_{All}|$ and $|TVA_{All}|$ for the generated TVCN. The effects of the *number of iterations*, *time slots*, and *nodes* on $|A_{All}|$, $|TA_{All}|$ and $|TVA_{All}|$ are described as follows.

1) Effect of the number of iterations

This simulation sets $|V|$ = 6, $\gamma = \theta$ = 0.4, τ = 4 in each run and then generates TAG as discussed in Section 4.5.2. It then applies Method 1 and Method 2 on the generated TAG to find $|A_{All}|$, $|TA_{All}|$ and $|TVA_{All}|$. The above explained process is repeated for 2000 to 12000 iterations. Figure 4.3 shows that average and standard deviation of $|TVA_{All}|$ over $|TA_{All}|$ and $|TVA_{All}|$ over $|A_{All}|$ remains almost invariable for each iteration.

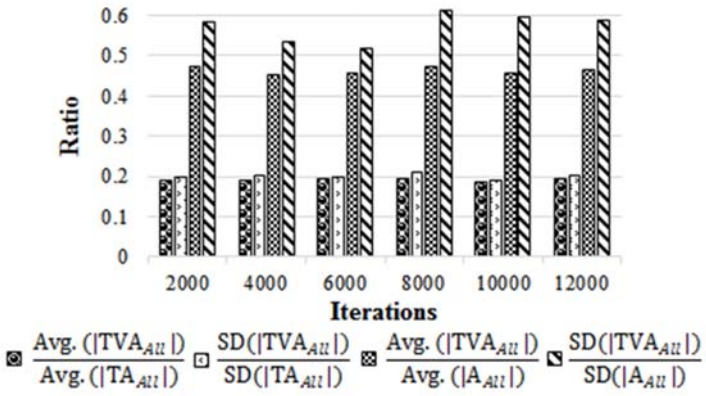


Figure 4.3. A plot showing the impact of number of iterations.

Note that in practice, the number of generated arborescences, t-arborescences and tv-arborescences are different in each iteration, but after averaging, the result is almost the same in number. Hence, the results in Figure 4.3 show that larger number of iterations do not have any remarkable impact on simulation performance.

2) Effect of the number of time slots

This simulation analyzes the impact of increase in the number of time slots on the average and standard deviation of $|A_{All}|$, $|TA_{All}|$ and $|TVA_{All}|$. The simulation considers a TVCN of six nodes, and sets the number of iterations to 10000. The number of time slots is varied from four to eight and all other parameters are set as in the simulation of Section 4.5.2.*1)*.

Figure 4.4 shows that on average the values of $|TA_{All}|$ and $|TVA_{All}|$ increase with more time slots. The figure also shows that the corresponding curve of standard deviation displays a similar increasing trend. The reason for the growth in the value of $|TVA_{All}|$ and $|TA_{All}|$ is intuitive because an increase in time slots gives more opportunities, *i.e.*, more tv-arborescences, for network broadcast. Besides, it also increases the number of t-arborescences exponentially. Thus, Method 1 spends a large computational time to filter all invalid t-arborescences while enumerating TVA_{All}. It is important to observe that the average $|TVA_{All}|$ is less than $|TA_{All}|$ for any time slot. This supports discussion in Section 4.2.1.

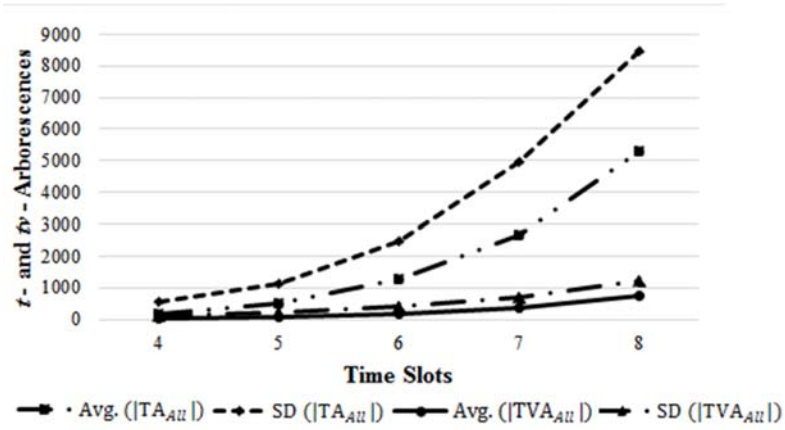


Figure 4.4. The impact of the number of time slots on $|TA_{All}|$ and $|TVA_{All}|$.

Figure 4.5 shows the average and standard deviation of $|TA_{All}|$ and $|A_{All}|$ with increasing number of time slots. It shows that $|TA_{All}|$ is always greater than $|A_{All}|$; this fact is also corroborated by our discussion in Section 4.2.1. Lastly, as there is no relationship between $|TVA_{All}|$ and $|A_{All}|$, we do not present their trend.

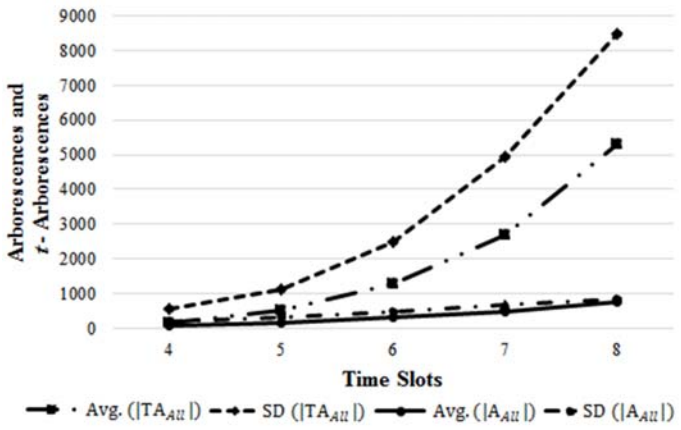


Figure 4.5. The impact of the number of time slots on $|TA_{All}|$ and $|A_{All}|$.

3) Effect of the number of nodes

This simulation considers a TVCN with $\tau = 4$. The number of iterations is 50. All other parameter values are the same as in Section 4.5.2.*1)*. It studies six to nine nodes and their impact on the average $|TA_{All}|$, $|TVA_{All}|$ and $|A_{All}|$ value. Intuitively, with more nodes, the average value of $|TA_{All}|$, $|TVA_{All}|$ and $|A_{All}|$ should increase. This is confirmed in Figure 4.6, where $|TA_{All}|$, $|TVA_{All}|$ and $|A_{All}|$ increases exponentially.

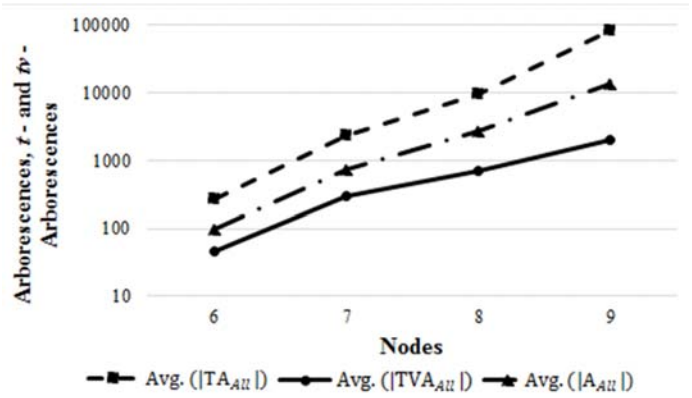


Figure 4.6. The impact of the number of nodes on $|TA_{All}|$, $|TVA_{All}|$ and $|A_{All}|$.

It is important to note that although a nominal edge density, governed by γ and θ, of 0.4 is considered for each of the above simulations, yet it results in a large number of contacts. Consequently, an excessive number of arborescences, t-arborescences and tv-arborescences are generated.

4.6 Summary

The chapter first introduced the notion of two types of arborescences of TVCNs, *viz.*, t-arborescences and tv-arborescences. The chapter also presented the properties of these arborescences. It then presented two novel adaptations of Tutte's Matrix Theorem, *i.e.*, Method 1 and Method 2, for enumerating all tv-arborescences of a TVCN. Using t-arborescences rooted at node v_s, the chapter also presented an approach to enumerate all (s, d) TS-MPS. Our simulation results, obtained using ten arbitrary TAGs, show that Method 2 performed comparatively better than Method 1 in terms of CPU time. Besides, simulations with randomly generated TAGs of variable size also supported our propositions and verified the efficacy of the proposal. In chapters 5 and 6, the enumerated arborescences will be utilized for analyzing some reliability metrics and for studying TVCN management and upgrade techniques.

Chapter 5 Reliability Evaluation of Device-to-Device Connected TVCNs

5.1 Introduction

In this chapter, we evaluate the reliability of device-to-device connected TVCNs. In this regard, the chapter first introduces unicast and broadcast reliability measures for TVCNs and then describes method to evaluate them using SDP-MVI algorithm. More specifically, here we apply SDP-MVI technique on all TS-MPS (enumerated in Chapter 3) and tv-arborescences (enumerated in Chapter 4) of TVCNs to generate reliability expressions, thereby values of the proposed novel reliability measures.

It is important to note that reliability evaluation of networks is an NP-hard problem (Rai et al. 1995). Among several techniques developed for exact reliability evaluation of conventional static networks, SDP approach is very popular. Specifically, SDP is a morphed and compact version of applying probability *law of unionization of events* or *inclusion-exclusion principle* on events. The basic concept behind the SDP approach can be easily described as follows.

Let A and B be two independent but not mutually exclusive events. Then the probability of occurrence of either of these two events would be,

$$\begin{aligned} p(A \cup B) &= p(A) + p(B) - p(A).p(B), \text{ or} \\ &= p(A) + p(\bar{A}).p(B) \text{ (Compact expression)} \end{aligned} \tag{5.1}$$

The bar over a variable in Equation (5.1) denotes its complement. Note that the above expression can be extended to several events consisting of a number of variables. Incidentally, such expression obtained through SDP has one-to-one mapping with the reliability expression.

The techniques based on SDP have been often used effectively and efficiently in reliability evaluation of complex networks to provide a compact reliability expression. Besides, while evaluating the product terms, the laws of probability such as idempotence and absorption play a major role in providing a compact expression. These techniques start with a Boolean polynomial formed by either success terms (*e.g.*, path sets) or failure terms (*i.e.*, cut sets). This class of techniques relies on Boolean algebra manipulation to convert a path/cut sets polynomial, consisting of un-complemented system variable (without bar), into a set of *exclusive and mutually disjoint* (emd) terms.

Recall that there are two versions of SDP, *viz.*, SVI and MVI (Rai et al. 1995). Both the SDP-SVI and SDP-MVI techniques accept a Boolean function of network variables (nodes and/or links) representing an MPS or MCS to compute reliability measures. However, in contrast to SVI approach wherein variables are sequentially inverted one at a time, in MVI approach a group of variables are inverted at once. In SVI/MVI, a minimized expression can be achieved through the ordering of path sets or cut sets. In MVI, not only the ordering of path sets or cut sets but the simultaneous inversion of a group of variables results in a more compact expression as compared to the expression rendered by SVI technique (Soh and Rai 1993). In other words, the MVI technique has built-in feature to provide a more compact reliability expression.

To further clarify the difference in number of terms in a reliability expression obtained through SVI and MVI, consider two path sets, $x_1x_2x_3x_4$ and $x_4x_5x_6x_7$. Using SDP technique, we obtain their disjoint terms, and hence reliability expression as:

By SVI,

$R(G) = p_{x_1}p_{x_2}p_{x_3}p_{x_4} + [(1-p_{x_1}) + (1-p_{x_2})p_{x_1} + (1-p_{x_3})p_{x_1}p_{x_2}]p_{x_4}p_{x_5}p_{x_6}p_{x_7}$, and

By MVI,

$R(G) = p_{x_1}p_{x_2}p_{x_3}p_{x_4} + (1-p_{x_1}p_{x_2}p_{x_3})p_{x_4}p_{x_5}p_{x_6}p_{x_7}$, where p_{x_i} $(1-p_{x_i})$ denotes the probability of success (failure) of an edge x_i, for $0 \leq p_{x_i} \leq 1$.

One can observe that the number of terms in the MVI expression, *i.e.*, two, is less than the number of terms in the SVI expression, *i.e.*, four. For more details, the interested readers may refer (Chaturvedi 2016).

In the context of TVCNs, let $p_{e_{a,b}^t}$ $(1 - p_{e_{a,b}^t})$ be the probability of success (failure) of a TSE $e_{a,b}^t$, for $0 \leq p_{e_{a,b}^t} \leq 1$. We consider that edge states are binary, *i.e.*, an edge is either completely operating or completely failed. Further, we assume that nodes are perfect, *i.e.*, the chances of their failure during the mission duration are nil. Note that one can use the approach presented in (Aggarwal et al. 1975) to also consider node-failure probability and transform an imperfect node to perfect one.

The chapter is organized as follows. In Section 5.2, we consider reliability metrics relevant to unicast communication. Section 5.3 discusses in detail about the broadcast related reliability metrics. In summary, the unicast reliability metrics are computed using the generated TS-MPS (see Chapter 3), while broadcast reliability metrics utilize the tv-arborescences (enumerated in Chapter 4). Section 5.4 presents simulation results, which is followed by chapter summary in Section 5.5.

5.2 Unicast Reliability Metrics

This section discusses about the reliability of communication between a specified (s, d) node pair in a device-to-device connected TVCN. More specifically, in Section 5.2.1, we evaluate the 2-terminal reliability related to (s, d) node pair, *i.e.*, $R(s, d)$, of TVCNs by using all TS-MPS. In addition, we describe methods to extract *foremost*, *shortest*, and *fastest* TS-MPS, respectively. We also explain the steps involved in evaluating the 2-terminal reliability associated with them. Section 5.2.2 describes the EHC metric and an example of its computation. In Section 5.2.3, we propose a new metric, called Expected Slot Count (ESC), and show how to calculate the metric.

5.2.1 2-Terminal Reliability

The 2-terminal reliability related to node pair (s, d), *i.e.*, $R(s, d)$, of device-to-device connected TVCN depicts a probability that data packets transmitted from source node v_s will be successfully received by the destination node v_d. Thus, it depicts success probability of at least one of the TS-MPS between a (s, d) node pair for a successful communication. To evaluate 2-terminal reliability $R(s, d)$ of TVCNs, we follow a two-step approach similar to the one used with static networks. The steps are: i) Enumerate all TS-MPS for a specified (s, d) node pair, such that $v_s \neq v_d$ and $v_s, v_d \in V$, and ii) Apply any SDP approach, *e.g.*, SDP-MVI (Misra 1993; Rai et al. 1995) on all TS-MPS enumerated in i) to make them disjointed and to obtain a compact reliability expression or figure. For node pair (v_1, v_3) of the TAG shown in Figure 2.1(ii), the 2-terminal reliability expression obtained by disjointing the two TS-MPS, *viz.*, $(e_{1,2}^1 . e_{2,3}^1)$ and $(e_{1,2}^1 . e_{2,3}^2)$, is:

$$R(1,3) = p_{e_{1,2}^1} p_{e_{2,3}^1} + \left(1 - p_{e_{2,3}^1}\right) p_{e_{1,2}^1} p_{e_{2,3}^2} \tag{5.2}$$

If each $p_{e_{a,b}^t}$ = 0.9, then Equation (5.2) results in $R(1, 3)$ as 0.891. We can infer from the above discussion that though there may be no direct end-to-end path between a specified (s, d) node pair, yet we may achieve an acceptable value of 2-terminal reliability of TVCNs via device-to-device communication. Moreover, the terminal reliability between node pair (v_1, v_3) turns out to be different from that of between the node pair (v_3, v_1) (which is 0.972). Obviously, as discussed before in Chapter 3, this difference in reliability occurs due to the existence of different TS-MPS, *i.e.*, 02, *viz.*, $\{(e_{1,2}^1 . e_{2,3}^1), (e_{1,2}^1 . e_{2,3}^2)\}$ and 03, *viz.*, $\{(e_{3,2}^1 . e_{2,1}^1), (e_{3,2}^1 . e_{2,1}^3), (e_{3,2}^2 . e_{2,1}^3)\}$, in forward and reverse direction, respectively, between the specified pair of nodes within a given period of time.

At this juncture, it is worth to note that by using all TS-MPS between a specified (s, d) node pair, we can also extract all *foremost*, *shortest*, and the *fastest* TS-MPS to compute the *earliest arrival date*, the *minimum number of hops*, and the *minimum delay* (time span),

respectively. The algorithms for obtaining *foremost*, *shortest*, and the *fastest* TS-MPS are given below.

Algorithm 5.1. Foremost_TS-MPS

1) Select a (s, d) node pair and enumerate all TS-MPS between them.
2) Group the enumerated TS-MPS according to the last timestamp.
3) Store the TS-MPS with minimal last timestamp value.

Algorithm 5.2. Shortest_TS-MPS

1) Select a (s, d) node pair and enumerate all TS-MPS between them.
2) Sort the enumerated TS-MPS in the order of increasing cardinality.
3) Store the TS-MPS with minimal cardinality.

Algorithm 5.3. Fastest_TS-MPS

1) Select a (s, d) node pair and enumerate all TS-MPS between them.
2) From these enumerated TS-MPS, compute the number of timeslots required by the TS-MPS.
3) Store the TS-MPS with minimum number of time slots.

To illustrate the above notions of *foremost*, *shortest*, and the *fastest* TS-MPS, consider the TAG of Figure 2.6. For this TAG, there are 20 TS-MPS between (s, d) node pair (v_1, v_5); see Table 3.2 or Table 5.1. These TS-MPS can now be utilized to find the above defined three types of TS-MPS using Algorithms 5.1, 5.2 and 5.3. The extracted TS-MPS have been collated in Table 5.1.

Table 5.1. Enumerated TS-MPS between node pair (v_1, v_5) of TAG of Figure 2.6.

Sl. No.	(v_1, v_5) TS-MPS	Fastest TS-MPS	Foremost TS-MPS	Shortest TS-MPS
1	$e^1_{1,2}.e^3_{2,5}$	$e^4_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}$	$e^1_{1,2}.e^3_{2,5}$	$e^1_{1,2}.e^3_{2,5}$
2	$e^2_{1,2}.e^3_{2,5}$		$e^2_{1,2}.e^3_{2,5}$	$e^2_{1,2}.e^3_{2,5}$
3	$e^1_{1,2}.e^2_{2,3}.e^3_{3,5}$		$e^1_{1,2}.e^2_{2,3}.e^3_{3,5}$	

Sl. No.	(v_1, v_5) TS-MPS	Fastest TS-MPS	Foremost TS-MPS	Shortest TS-MPS
4	$e^2_{1,2}.e^2_{2,3}.e^3_{3,5}$		$e^1_{1,2}.e^3_{2,3}.e^3_{3,5}$	
5	$e^1_{1,2}.e^3_{2,3}.e^3_{3,5}$		$e^2_{1,2}.e^2_{2,3}.e^3_{3,5}$	
6	$e^2_{1,2}.e^3_{2,3}.e^3_{3,5}$		$e^2_{1,2}.e^3_{2,3}.e^3_{3,5}$	
7	$e^1_{1,2}.e^2_{2,3}.e^4_{3,5}$		$e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^3_{3,5}$	
8	$e^2_{1,2}.e^2_{2,3}.e^4_{3,5}$		$e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$	
9	$e^1_{1,2}.e^3_{2,3}.e^4_{3,5}$		$e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$	
10	$e^2_{1,2}.e^3_{2,3}.e^4_{3,5}$			
11	$e^1_{1,2}.e^4_{2,3}.e^4_{3,5}$			
12	$e^2_{1,2}.e^4_{2,3}.e^4_{3,5}$			
13	$e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^3_{3,5}$			
14	$e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$			
15	$e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5}$			
16	$e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^4_{3,5}$			
17	$e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5}$			
18	$e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5}$			
19	$e^3_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}$			
20	$e^4_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}$			

After enumerating all *foremost*, *shortest*, and the *fastest* TS-MPS, we can apply the SDP-MVI technique, as discussed in the beginning of Section 5.2.1 to obtain Equation (5.2), to obtain the 2-terminal reliability expression, thereby value, associated with them. If the edges are assumed to have same probability of success, say 0.90, then the *foremost* TS-MPS (providing the *earliest arrival date*) yields 2-terminal reliability equal to 0.9800, the *shortest* TS-MPS (providing the *minimum number of hops*) produces a 2-terminal reliability figure of 0.8910 whereas the *fastest* TS-MPS (providing *minimum delay*) results in a 2-terminal reliability of 0.6561. Further, by considering all TS-MPS (*i.e.*, Column 2 of Table 5.1), the 2-terminal reliability would be 0.99620, which is higher than that of the *foremost*, *shortest*, or *fastest* TS-MPS. Note that these TS-MPS can also be used to optimize effective routing under different objective functions.

5.2.2 Expected Hop Count (EHC)

The enumerated TS-MPS between a (s, d) pair of nodes can be used to compute the EHC between them as (Soh et al. 2007):

$$\text{EHC} = \left(\frac{\sum_{l=1}^{|V|-1} l \times p(l)}{\sum_{l=1}^{|V|-1} p(l)} \right) \tag{5.3}$$

where $p(l)$ is the probability that a source node v_s is connected to the destination node v_d with a shortest path of length $1 \leq l \leq |V| - 1$ and $R(s, d) = \sum_{l=1}^{|V|-1} p(l)$.

Note that Equation (5.3) assumes that the routing protocol in the network always finds the available (s, d) shortest path set. When such path set is unavailable, the router finds the next possible shortest path with the same or longer hop count. Algorithm 5.4 presents steps to compute EHC in a TVCN.

Algorithm 5.4. Expected_Hop_Count

1) Select a (s, d) node pair and enumerate all TS-MPS between them.
2) Sort the enumerated TS-MPS in the increasing order of cardinality.
3) Form cumulative group of TS-MPS obtained from Step 2).
4) Generate mutually disjoint terms to compute probability $p(l), \forall\ 1 \leq l \leq |V| - 1$ from the cumulative group of TS-MPS obtained in Step 3).
5) Compute the EHC using Equation (5.3).

To clarify the steps taken to compute EHC, consider the example of Figure 2.6 and Column 2 of Table 5.1. These TS-MPS generate 20 mutually disjoint terms, *i.e.*,

$$\boldsymbol{p(2)} = e_{1,2}^{1}.e_{2,5}^{3} + \overline{e_{1,2}^{1}}.e_{1,2}^{2}.e_{2,5}^{3},$$

$$\boldsymbol{p(3)} = \overline{e^3_{2,5}}.e^1_{1,2}.e^2_{2,3}.e^3_{3,5} + \overline{e^1_{1,2}}.\overline{e^3_{2,5}}.e^2_{1,2}.e^2_{2,3}.e^3_{3,5} + \overline{e^3_{2,5}}.\overline{e^2_{2,3}}.e^1_{1,2}.e^3_{2,3}.e^3_{3,5} +$$
$$\overline{e^1_{1,2}}.\overline{e^3_{2,5}}.\overline{e^2_{2,3}}.e^2_{1,2}.e^3_{2,3}.e^3_{3,5} + \overline{e^3_{2,5}}.\overline{e^3_{3,5}}.e^1_{1,2}.e^2_{2,3}.e^4_{3,5} + \overline{e^1_{1,2}}.\overline{e^3_{2,5}}.\overline{e^3_{3,5}}.e^2_{1,2}.e^2_{2,3}.e^4_{3,5} +$$
$$\overline{e^2_{2,3}}.\overline{e^3_{2,5}}.\overline{e^3_{3,5}}.e^1_{1,2}.e^3_{2,3}.e^4_{3,5} + \overline{e^1_{1,2}}.\overline{e^2_{2,3}}.\overline{e^3_{2,5}}.\overline{e^3_{3,5}}.e^2_{1,2}.e^3_{2,3}.e^4_{3,5} + \overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^1_{1,2}.e^4_{2,3}.e^4_{3,5} +$$
$$\overline{e^1_{1,2}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^2_{1,2}.e^4_{2,3}.e^4_{3,5},$$

$$\boldsymbol{p(4)} = \overline{e^4_{2,3}.e^4_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^3_{3,5} +$$
$$\overline{e^1_{2,4}}.\overline{e^4_{2,3}.e^4_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5} +$$
$$\overline{e^1_{1,2}}.\overline{e^4_{2,3}.e^4_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^3_{3,5} + \overline{e^4_{2,3}}.\overline{e^3_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^1_{1,2}.e^1_{2,4}.e^2_{4,3}.e^4_{3,5} +$$
$$\overline{e^1_{2,4}}.\overline{e^4_{2,3}}.\overline{e^3_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^1_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5} +$$
$$\overline{e^1_{1,2}}.\overline{e^4_{2,3}}.\overline{e^3_{3,5}}.\overline{e^2_{2,3}}.\overline{e^3_{2,3}}.\overline{e^3_{2,5}}.e^2_{1,2}.e^2_{2,4}.e^2_{4,3}.e^4_{3,5} + \overline{e^1_{1,2}}.\overline{e^2_{1,2}}.e^3_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5} +$$
$$\overline{e^1_{1,2}}.\overline{e^2_{1,2}}.\overline{e^3_{1,4}}.e^4_{1,4}.e^4_{4,2}.e^4_{2,3}.e^4_{3,5}.$$

From the 20 disjoint terms, one can obtain the following corresponding reliability expressions:

$$\boldsymbol{p(2)} = p_{e^1_{1,2}}.p_{e^3_{2,5}} + \overline{p_{e^1_{1,2}}}.p_{e^2_{1,2}}.p_{e^3_{2,5}}$$

$$\boldsymbol{p(3)} = \overline{p_{e^3_{2,5}}}.p_{e^1_{1,2}}.p_{e^2_{2,3}}.p_{e^3_{3,5}} + \overline{p_{e^1_{1,2}}}.\overline{p_{e^3_{2,5}}}.p_{e^2_{1,2}}.p_{e^2_{2,3}}.p_{e^3_{3,5}} + \overline{p_{e^3_{2,5}}}.\overline{p_{e^2_{2,3}}}.p_{e^1_{1,2}}.p_{e^3_{2,3}}.p_{e^3_{3,5}}$$
$$+ \overline{p_{e^1_{1,2}}}.\overline{p_{e^3_{2,5}}}.\overline{p_{e^2_{2,3}}}.p_{e^2_{1,2}}.p_{e^3_{2,3}}.p_{e^3_{3,5}} + \overline{p_{e^3_{2,5}}}.\overline{p_{e^3_{3,5}}}.p_{e^1_{1,2}}.p_{e^2_{2,3}}.p_{e^4_{3,5}} +$$
$$\overline{p_{e^1_{1,2}}}.\overline{p_{e^3_{2,5}}}.\overline{p_{e^3_{3,5}}}.p_{e^2_{1,2}}.p_{e^2_{2,3}}.p_{e^4_{3,5}} + \overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,5}}}.\overline{p_{e^3_{3,5}}}.p_{e^1_{1,2}}.p_{e^3_{2,3}}.p_{e^4_{3,5}} +$$
$$\overline{p_{e^1_{1,2}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,5}}}.\overline{p_{e^3_{3,5}}}.p_{e^2_{1,2}}.p_{e^3_{2,3}}.p_{e^4_{3,5}} + \overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^1_{1,2}}.p_{e^4_{2,3}}.p_{e^4_{3,5}} +$$
$$\overline{p_{e^1_{1,2}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^2_{1,2}}.p_{e^4_{2,3}}.p_{e^4_{3,5}}$$

$$\boldsymbol{p(4)} = \overline{p_{e^4_{2,3}.e^4_{3,5}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^1_{1,2}}.p_{e^1_{2,4}}.p_{e^2_{4,3}}.p_{e^3_{3,5}} +$$
$$\overline{p_{e^1_{2,4}}}.\overline{p_{e^4_{2,3}.e^4_{3,5}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^1_{1,2}}.p_{e^2_{2,4}}.p_{e^2_{4,3}}.p_{e^3_{3,5}} +$$
$$\overline{p_{e^1_{1,2}}}.\overline{p_{e^4_{2,3}.e^4_{3,5}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^2_{1,2}}.p_{e^2_{2,4}}.p_{e^2_{4,3}}.p_{e^3_{3,5}} +$$
$$\overline{p_{e^4_{2,3}}}.\overline{p_{e^3_{3,5}}}.\overline{p_{e^2_{2,3}}}.\overline{p_{e^3_{2,3}}}.\overline{p_{e^3_{2,5}}}.p_{e^1_{1,2}}.p_{e^1_{2,4}}.p_{e^2_{4,3}}.p_{e^4_{3,5}} +$$

$\overline{p_{e_{2,4}^1}}.\overline{p_{e_{2,3}^4}}.\overline{p_{e_{3,5}^3}}.\overline{p_{e_{2,3}^2}}.\overline{p_{e_{2,3}^3}}.\overline{p_{e_{2,5}^3}}.p_{e_{1,2}^1}.p_{e_{2,4}^2}.p_{e_{4,3}^2}.p_{e_{3,5}^4}$ +

$\overline{p_{e_{1,2}^1}}.\overline{p_{e_{2,3}^4}}.\overline{p_{e_{3,5}^3}}.\overline{p_{e_{2,3}^2}}.\overline{p_{e_{2,3}^3}}.\overline{p_{e_{2,5}^3}}.p_{e_{1,2}^2}.p_{e_{2,4}^2}.p_{e_{4,3}^2}.p_{e_{3,5}^4}$ + $\overline{p_{e_{1,2}^1}}.\overline{p_{e_{1,2}^2}}.p_{e_{1,4}^3}.p_{e_{4,2}^4}.p_{e_{2,3}^4}.p_{e_{3,5}^4}$ +

$\overline{p_{e_{1,2}^1}}.\overline{p_{e_{1,2}^2}}.\overline{p_{e_{1,4}^3}}.p_{e_{1,4}^4}.p_{e_{4,2}^4}.p_{e_{2,3}^4}.p_{e_{3,5}^4}$.

Setting the probability of success of each TSE to 0.90, one can compute $R(1,5) = p(2) + p(3) + p(4) = 0.891 + 0.09783 + 0.00737 = 0.9962$. Further, using Equation (5.3), EHC = $(2\,p(2) + 3\,p(3) + 4\,p(4))/\ R(1,5)$ = 2.1130 hops. Notice that the minimum (maximum) hop count is 2 (4), and $2 \leq$ EHC = 2.1130 ≤ 4.

5.2.3 Expected Slot Count (ESC)

Similar to EHC, the enumerated TS-MPS between a (s,d) pair of nodes can be used to compute the ESC between them. Algorithm 5.5 describes the steps required to compute ESC. Note that, in this algorithm, l' is the last time slot available in a particular TS-MPS, and $p(l')$ is the probability that a source node v_s is connected to the destination node v_d with an earliest TS-MPS (in terms of time) of length $1 \leq l' \leq \tau$.

Algorithm 5.5: Expected_Slot_Count

1) Select a (s,d) node pair and enumerate all TS-MPS between them.
2) Sort the TS-MPS according to the last time slot available in each TS-MPS.
3) Form cumulative group of TS-MPS obtained from Step 2).
4) Generate mutually disjoint terms to compute probability $p(l')$, $\forall\ 1 \leq l' \leq \tau$ from the cumulative group of TS-MPS obtained in Step 3).
5) Compute ESC using Equation (5.3), wherein l is replaced by l'.

For the example considered, we sort the 20 TS-MPS in the order of their ending time slot (*i.e.*, Step 2)). Note that there are nine and 11 TS-MPS that end after three time slots and four slots, respectively. Further, there are two cumulative groups of TS-MPS, *viz.*, i) ending in time slot 3, and ii) ending in time slot 4 including all TS-MPS ending in time slot 3. These groups of TS-MPS will be used to evaluate probability $p(3)$ by generating

mutually disjoint terms for TS-MPS ending at time slot 3, while probability $p(4)$ is computed by disjointing all TS-MPS ending in time slot 4 along with all TS-MPS ending in time slot 3. Setting the probability of success of each TSE to 0.90, we obtain $R(1,5) = p(3) + p(4) = 0.9800 + 0.0162 = 0.9962$. Thus, ESC = $(3\ p(3) + 4\ p(4))/R(1,5)$ = 3.0162 slots. Notice that the minimum (maximum) slot count is 3 (4), and $3 \leq$ ESC = $3.0163 \leq 4$.

5.3 Broadcast Reliability Metrics

This section discusses about the reliability of network broadcast in a device-to-device connected TVCN. More specifically, in Section 5.3.1, we discuss about the reliability of broadcast from root node v_j, *i.e.*, $R(v_j)$ and define its evaluation steps. Section 5.3.2 presents two other broadcast reliability metrics, *viz.*, $R_1(K)$ and $R_2(K)$ for TVCNs that consider a set of root nodes $K \subseteq V$. The section also explains the methods to compute them.

5.3.1 Reliability of Broadcast from Node v_j of TVCN

Reliability metric $R(v_j)$ is the probability that data packets broadcasted by a root node v_j are successfully received by all other nodes of the network. To illustrate reliability $R(v_j)$, consider Figure 2.1(ii) with root node v_1; thus, $R(v_1)$ is the probability of at least one of the two tv-arborescences, *i.e.*, $(e^1_{1,2}.\, e^1_{2,3}.\, e^4_{3,4})$ and $(e^1_{1,2}.\, e^2_{2,3}.\, e^4_{3,4})$, being operational.

Similar to the approach used to evaluate $R(s,d)$, we follow a two-step approach for computing the reliability $R(v_j)$. Step 1 requires enumeration of all tv-arborescences rooted at node v_j. For Step 2, we propose using any SDP technique, *e.g.*, SDP-MVI (Rai et al. 1995; Chaturvedi 2016), to disjoint and convert the obtained set of tv-arborescences into its reliability expression. For example, the TAG shown in Figure 2.1(ii) has two tv-arborescences rooted at node v_1 (see Table 4.1). Disjointing the two tv-arborescences results in the $R(v_1)$ expression as:

$$R(v_1) = p_{e^1_{1,2}} p_{e^1_{2,3}} p_{e^4_{3,4}} + (1 - p_{e^1_{2,3}}) p_{e^1_{1,2}} p_{e^2_{2,3}} p_{e^4_{3,4}} \quad (5.4)$$

Assuming each $p_{e^t_{a,b}}$ has an equal communication probability of 0.90, then the above expression results in $R(v_1)$ = 0.8019. Similarly, by applying SDP-MVI technique on four and three tv-arborescences from root node v_2 and v_3, respectively (see Table 4.1), we have $R(v_2)$ = 0.88209 and $R(v_3)$ = 0.8748. As $|\text{TVA}_4| = 0$, so, $R(v_4)$ = 0.

Note that the general expression of $R(v_1)$ in Equation (5.4) can have dissimilar value of $p_{e^t_{a,b}}$ for different pair of nodes and also for different values of timestamp t between a specified pair of nodes. For example, $p_{e^1_{1,2}}$ may not be equal to $p_{e^1_{2,3}}$. Besides, $p_{e^1_{2,3}}$ and $p_{e^2_{2,3}}$ may also be different as they appear at different instants of time between a given node pair (v_2, v_3).

The following proposition shows that TSE reliability and the number of tv-arborescences $|\text{TVA}_i|$ from any source node v_i are not the only parameters that affect the value of broadcast reliability $R(v_i)$.

Proposition 5.1: Consider a TAG $G'(V, E^{TS})$ and two source nodes $v_i, v_j \in V$. The broadcast reliability metric $R(v_i)$ from source node $v_i \in V$ can be larger than the reliability $R(v_j)$ from another node $v_j \in V$ even when the number of tv-arborescences $|\text{TVA}_i|$ from node v_i is less than the number of tv-arborescences $|\text{TVA}_j|$ from node v_j and each TSE has the same reliability.

Proof: Let TVA_i^k represent the k^{th} tv-arborescence from source node $v_i \in V$ in a TAG G', for k = 1, 2, …, m; so, $|\text{TVA}_i| = m$. The SDP technique (Rai et al. 1995; Lucet and Manouvrier 1999) uses the following formula to produce the broadcast reliability expression from node v_i: $R(v_i) = \sum_{k=1}^{m} \text{TVA}_i^k = \text{TVA}_i^1 + \sum_{k=2}^{m} [\text{TVA}_i^k \cdot \prod_{a=1}^{k-1} \overline{\text{TVA}_i^a}]$, where $\overline{\text{TVA}_i^a}$ is failure event of a^{th} tv-arborescence from source node v_i. Similarly, the broadcast reliability from source $v_j \in V$ is expressed as: $R(v_j) = \sum_{k=1}^{n} \text{TVA}_j^k = \text{TVA}_j^1 +$

$\sum_{k=2}^{n}[\text{TVA}_j^k . \prod_{a=1}^{k-1} \overline{\text{TVA}_j^a}]$, for $|\text{TVA}_j| = n$. One then computes the reliability value $R(v_i)$ ($R(v_j)$) by substituting every element in each TVA_i^k (TVA_j^k) and $\overline{\text{TVA}_i^a}$ $\left(\overline{\text{TVA}_j^a}\right)$ with its corresponding TSE reliability. Thus the reliability value of $R(v_i)$ and $R(v_j)$ depends on the following metrics: (i) the total number of terms, $|R(v_i)|$ and $\left|R\left(v_j\right)\right|$, in the reliability expressions for $R(v_i)$ and $R(v_j)$, respectively, and (ii) the constituting edges, *i.e.*, TSEs, in each TVA_i^k and TVA_j^k as the disjunction must be applied for each elementary variables of the sum. For (i), following (Rai et al. 1995; Lucet and Manouvrier 1999), $|R(v_i)|$ can be larger than $|R(v_j)|$. The reason is because total disjoint terms depend on the constituting edges in each tv-arborescence. Thus, in general, one can have $|R(v_i)| \geq |R(v_j)|$ or $|R(v_i)| < |R(v_j)|$. For (ii), due to laws of probability, *viz*., *absorption* and *idempotency*, a network having more *disjoint* tv-arborescences will result in higher reliability compared to the other in which several edges are repeated in multiple arborescences. Thus, it is possible to have $R(v_i) > R(v_j)$. ■

5.3.2 Reliability of Broadcast from Nodes in Set *K* of TVCN

We propose two variants of the reliability of broadcast from nodes in set K, *viz.*, $R_1(K)$ and $R_2(K)$. Reliability metric $R_1(K)$ is the probability that the data packets broadcasted by *all* nodes in set K are successfully received by all other nodes of the network. In contrast, $R_2(K)$ is the probability that the data broadcasted by *any* node in set K will be received by all others nodes of the network. Note that the number of nodes in set K ranges from 1 to $|V|$, and thus $R_1(K) = R_2(K) = R(v_j)$ when $|K| = 1$. Further, in general, $R_1(K)$ and $R_2(K)$ for $|K| = |V|$ are larger than $R_1(K)$ and $R_2(K)$ for $|K| \subset |V|$, respectively. To illustrate $R_1(K)$ and $R_2(K)$ consider Figure 2.1(ii) with $K = \{v_1, v_2\}$. As shown in Figure 2.2, there are two tv-arborescences (in solid edges) from node v_1 and four tv-arborescences (in dotted edges) from node v_2. For this example, $R_1(K)$ computes the probability that at least one of the two tv-arborescences from v_1 *and* one of the four tv-arborescences from v_2 are operational, *e.g.*, $(e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4)$ and $(e_{2,1}^3 . e_{2,3}^2 . e_{3,4}^4)$. Equivalently, for this example, $R_1(K)$ gives the probability that one of eight ($= 2 \times 4$) possible combinations of tv-arborescences of v_1 and v_2 is operational. On the other hand, $R_2(K)$ is the probability of any of the six tv-

arborescences rooted at node v_1 and v_2 being operational. In the following, we describe our proposed methods to compute $R_1(K)$ and $R_2(K)$.

5.3.2.1 Computing $R_1(K)$

We propose a two-step approach to evaluate $R_1(K)$. Step 1 finds all possible *irredundant* and *minimal* combinations of tv-arborescences using one spanning arborescence from each $\text{TVA}_K \; \forall \; K \in [1,2, \ldots, |V|]$. Recall that TVA_j is a set of all tv-arborescences from root node v_j. Let us assume that $\text{TVA}_1 = \{\text{TVA}_1^1, \text{TVA}_1^2, \text{TVA}_1^3, \ldots, \text{TVA}_1^k\}$, $\text{TVA}_2 = \{\text{TVA}_2^1, \text{TVA}_2^2, \text{TVA}_2^3, \ldots, \text{TVA}_2^l\}$, …, and $\text{TVA}_{|V|} = \{\text{TVA}_{|V|}^1, \text{TVA}_{|V|}^2, \text{TVA}_{|V|}^3, \ldots, \text{TVA}_{|V|}^m\}$. Thus, we obtain at maximum $z = k \times l \times \ldots \times m$ irredundant and minimal combinations represented by C_Q, where $1 \leq Q \leq z$. A combination is considered as *irredundant* if it is unique and *minimal* if there exists no superset of this combination. For instance, $\text{C}_1 = \{\text{TVA}_1^1 \cup \text{TVA}_2^1 \cup \ldots \cup \text{TVA}_{|V|}^1\}$, $\text{C}_2 = \{\text{TVA}_1^1 \cup \text{TVA}_2^1 \cup \ldots \cup \text{TVA}_{|V|}^2\}$ and so on. If node v_1 and v_2 are the two root nodes in the TAG of Figure 2.1(ii), then $K = \{v_1, v_2\}$. Note from Table 4.1 that $|\text{TVA}_1| = 2$ and $|\text{TVA}_2| = 4$, so, we can obtain at maximum $2 \times 4 = 8$ irredundant and minimal combinations. Further, observe from Table 4.1 that each tv-arborescence has three TSEs, so each of the eight combinations can have a maximum length of six TSEs; if each of the TSE is unique. However, only four of the eight combinations are irredundant and minimal, *i.e.*, $\{e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4 . e_{2,1}^1\}$, $\{e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4 . e_{2,1}^3\}$, $\{e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4 . e_{2,1}^3\}$ and $\{e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4 . e_{2,1}^1\}$, each of which has length four. The remaining four combinations *viz.*, $\{e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4 . e_{2,1}^1 . e_{2,3}^2\}$, $\{e_{1,2}^1 . e_{2,3}^1 . e_{3,4}^4 . e_{2,1}^3 . e_{2,3}^2\}$, $\{e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4 . e_{2,1}^3 . e_{2,3}^1\}$ and $\{e_{1,2}^1 . e_{2,3}^2 . e_{3,4}^4 . e_{2,1}^1 . e_{2,3}^1\}$ are non-minimal as they are superset of others. Step 2 generates the desired reliability expression and/or value from the obtained combinations using any SDP approach (Rai et al. 1995; Chaturvedi 2016). For example, the $R_1(K)$ expression, for $K = \{v_1, v_2\}$, obtained by disjointing the four combinations via SDP-MVI is shown in Equation (5.5). If each $p_{e_{a,b}^t} = 0.90$, then Equation (5.5) results in $R_1(K)$ as 0.793881.

$$\begin{aligned} R_1(K) = {} & p_{e_{1,2}^1} p_{e_{2,3}^1} p_{e_{3,4}^4} p_{e_{2,1}^1} + (1 - p_{e_{2,1}^1}) p_{e_{1,2}^1} p_{e_{2,3}^1} p_{e_{3,4}^4} p_{e_{2,1}^3} + \\ & (1 - p_{e_{2,3}^1}) p_{e_{1,2}^1} p_{e_{2,3}^2} p_{e_{3,4}^4} p_{e_{2,1}^3} + (1 - p_{e_{2,3}^1})(1 - p_{e_{2,1}^3}) p_{e_{1,2}^1} p_{e_{2,3}^2} p_{e_{3,4}^4} p_{e_{2,1}^1} \end{aligned} \tag{5.5}$$

Note that generating the irredundant and minimal combinations in Step 1 in general is impractical as it becomes computationally intractable for TVCNs having large number of tv-arborescences from each node $v_j \in V$. It is worthy to note here that we also tried to invert the irredundant and minimal combinations of tv-arborescences to obtain their dual and use them to compute the well-known Esary and Proschan's (Y. -C. Hsieh 2003) lower bound of reliability. However, we found the inversion task also to be intractable. Further, as the combinations may have high degree of dependence among each other, due to the presence of multiple common TSEs, the Bonferroni inequalities (Schwager 1984) also cannot be utilized to find bounds, as they are expected to result in loose inequalities.

5.3.2.2 Computing $R_2(K)$

To compute metric $R_2(K)$, we propose using any SDP method (Rai et al. 1995; Chaturvedi 2016) on all tv-arborescences generated from each node v_j in K. Recall that $R_2(K)$ will give us the probability that the data broadcasted by *any* node in set K will be received by all other nodes present in the network. For example, consider $K = \{v_1, v_2\}$ and refer Table 4.1. We calculate $R_2(K)$ by disjointing, via SDP-MVI, the six tv-arborescences associated with nodes v_1 and v_2 to obtain the reliability expression given in Equation (5.6).

$$\begin{aligned} R_2(K) = {} & p_{e_{1,2}^1} p_{e_{2,3}^1} p_{e_{3,4}^4} + (1 - p_{e_{2,3}^1}) p_{e_{1,2}^1} p_{e_{2,3}^2} p_{e_{3,4}^4} + (1 - p_{e_{1,2}^1}) p_{e_{2,1}^1} p_{e_{2,3}^1} p_{e_{3,4}^4} + \\ & (1 - p_{e_{2,3}^1})(1 - p_{e_{1,2}^1}) p_{e_{2,1}^1} p_{e_{2,3}^2} p_{e_{3,4}^4} + (1 - p_{e_{2,1}^1})(1 - p_{e_{1,2}^1}) p_{e_{2,1}^3} p_{e_{2,3}^1} p_{e_{3,4}^4} + \\ & (1 - p_{e_{2,3}^1})(1 - p_{e_{2,1}^1})(1 - p_{e_{1,2}^1}) p_{e_{2,1}^3} p_{e_{2,3}^2} p_{e_{3,4}^4} \end{aligned} \tag{5.6}$$

If each $p_{e_{a,b}^t} = 0.90$, the value of $R_2(K)$ from Equation (5.6) is 0.890109, which is larger than the $R_1(K)$ obtained from Equation (5.5), *i.e.*, 0.793881. As the evaluation criteria is more stringent for assessment of $R_1(K)$ compared to $R_2(K)$, we conjecture that $R_2(K) \geq R_1(K)$, $\forall\, K \in [1,2, \ldots, |V|]$. Thus, one can use the reliability $R_2(K)$ as the upper bound for $R_1(K)$. Besides, note that Proposition 5.1 presented before also applies to the reliability metric $R_1(K)$ and $R_2(K)$.

5.4 Simulation Results

We present the simulation results pertaining to the unicast reliability metrics in Section 5.4.1. In contrast, Section 5.4.2 presents the simulation results obtained from the broadcast reliability metrics.

5.4.1 Simulation Results – Unicast Reliability

Recall from Chapter 3 that the number of TS-MPS in a TVCN depends on the number of time slots. Therefore, the 2-terminal reliability, which depends on the enumerated TS-MPS, may be affected by variation in time slots. Besides, the 2-terminal reliability of a TVCN also varies with the edge communication probability; see Section 3.2.1.2. To assess the impact of these parameters on the 2-terminal reliability of a TVCN, we randomly generated TAGs of five and six nodes, respectively. The edge creation in these arbitrarily generated TAGs was dictated by the probability of edge formation. We repeated the simulations from 2000 to 14000 iterations, to obtain all relevant plots. The upcoming paragraph shows all obtained plots.

Figure 5.1 describes the relationship between average 2-terminal reliability of the network and edge formation probability with Monte Carlo Simulation iterations set to 10000, keeping time slots of four units and edge communication probability equal to 0.7.

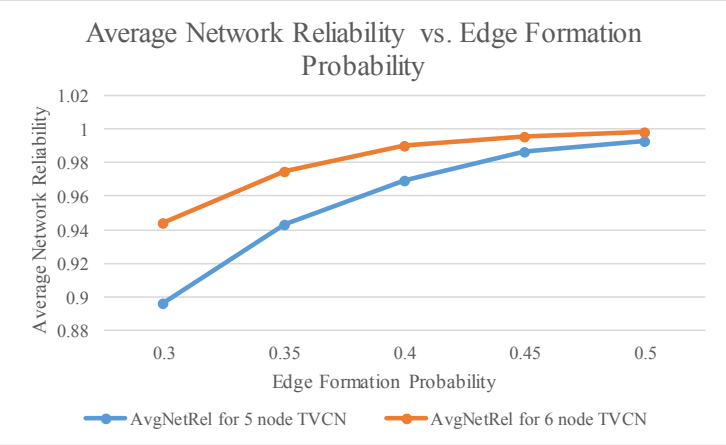


Figure 5.1. Average network reliability vs. edge formation probability at 10000 iterations, 4 time slots and edge communication probability of 0.7.

It is evident from Figure 5.1 that with an increase in the edge formation probability, the overall transmission opportunities increase. In other words, the environment becomes less disruptive. In such a case, the network reliability shows an increasing trend.

Figure 5.2 describes the relationship between average 2-terminal reliability of the network and edge communication probability when the number of iterations is fixed to 10000, time slots to four units and edge formation probability is fixed as 0.3. As the edge communication probability increases, intuitively the network reliability should increase. The observations from simulation validate the intuition. It also shows that though the edge formation probability is very meagre (*i.e.*, 0.3) and edge communication probability is just 0.7, but, the overall network reliability is very high, even in such conditions.

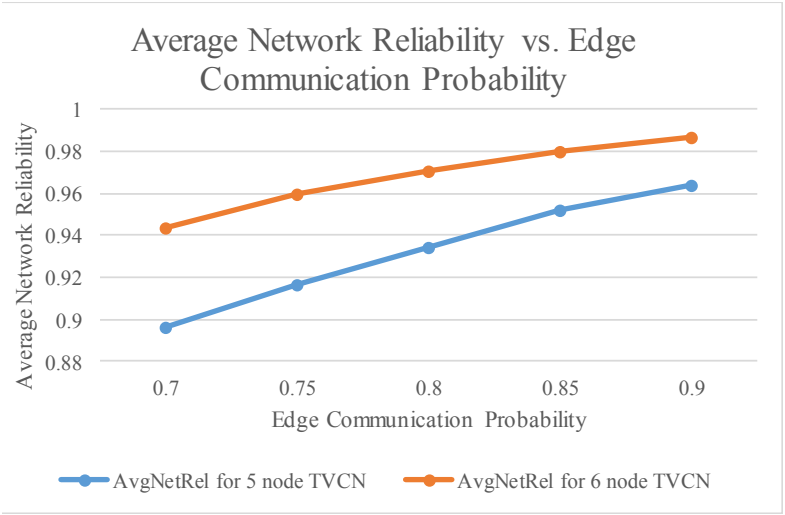


Figure 5.2. Average network reliability vs. edge communication probability at 10000 iterations, 4 time slots and edge formation probability of 0.3.

Figure 5.3 describes the relationship between average 2-terminal reliability of the network and time slots, when the number of iterations is fixed to 10000, edge formation probability is fixed as 0.3 and edge communication probability is fixed as 0.7. It can be observed from Figure 5.3 that as the period of the network increases, the overall network reliability for packet transmission increases. This occurs due to formation of more TS-MPS with increasing time.

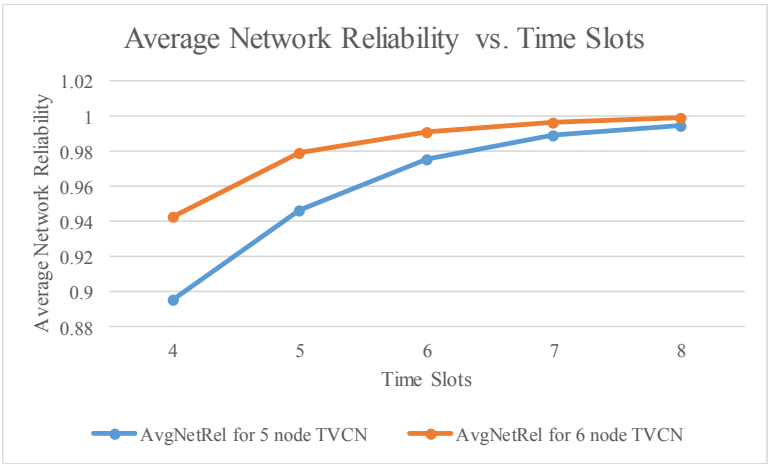


Figure 5.3. Average network reliability vs. time slots at 10000 iterations, edge formation probability of 0.3 and edge communication probability of 0.7.

Figure 5.4 describes the relationship between average 2-terminal reliability of the network and the number of iterations, when time slots are fixed to 4, edge formation probability is fixed as 0.5 and edge communication probability is fixed as 0.7. The plot suggests that network reliability almost stabilizes after 4000 iterations. Therefore, for the selected parameters, large number of Monte Carlo Simulation iterations do not have any remarkable impact on the simulation performance. Hence, here 4000-5000 iterations are enough to produce good results.

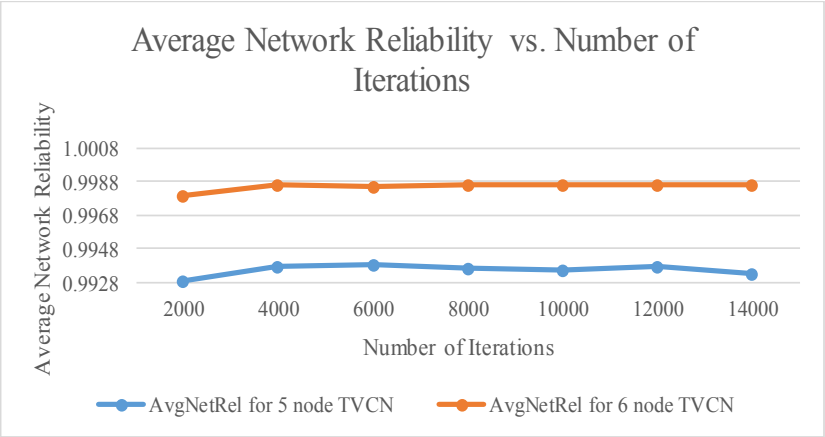


Figure 5.4. Average network reliability vs. number of iterations at 4 time slots, link formation probability of 0.5 and link communication probability of 0.7.

It is important to recall here that 2-terminal unreliability value is complementary to 2-terminal reliability. Hence, similar results of 2-terminal reliability can be obtained using all enumerated TS-MCS between (s, d) node pair. Further, for each TAG of Table 3.6, Table 5.2 analyzes the obtained reliability metric $R(s, d)$ between specified (s, d) pair of nodes. The obtained results verify the fact that it is not necessary that |TS-MPS$_{s,d}$| is equal to |TS-MPS$_{d,s}$|. Note that even if |TS-MPS$_{s,d}$| is equal to |TS-MPS$_{d,s}$|, it is not necessary that $R(s, d) = R(d, s)$ because there may be more multiple common or altogether different TSEs in TS-MPS between (s, d) as compared to between (d, s). For example, in TAG #6, |TS-MPS$_{1,6}$| = |TS-MPS$_{6,1}$| = 9, but $R(6, 1) > R(1, 6)$. In addition, for each TAG, Table 5.2 presents the values of EHC and ESC between the specified pair of nodes. On the basis of their values it is easy to assess the average number of hops and latency required to transmit data between the specified node pairs.

Table 5.2. TS-MPS and 2-terminal reliability results from different TAGs of Table 3.6.

TAG #	(s, d)	|TS-MPS$_{s,d}$|	EHC	ESC	$R(s, d)$
1	(1, 5)	20	2.1130	3.0162	0.9962
	(5, 1)	13	3.0114	2.2869	0.9779
2	(1, 3)	4	2.0149	2.0976	0.9784
	(3, 1)	6	2.0007	3.0177	0.9969
3	(1, 7)	8	3.2203	2.9380	0.9528
	(7, 1)	4	3.1406	3.2968	0.9331
4	(1, 5)	12	1.0100	1.0314	0.9999
	(5, 1)	12	1.0100	1.0315	0.9999
5	(1, 6)	29	3.1079	2.0620	0.9950
	(6, 1)	2	4.0825	2.0825	0.7151
6	(1, 6)	9	1.1975	3.1969	0.9986
	(6, 1)	9	1.1985	2.1391	0.9991
7	(1, 7)	6	3.0946	2.1458	0.9735
	(7, 1)	6	3.2379	2.4218	0.9566
8	(1, 5)	18	2.0019	1.0412	0.9999
	(5, 1)	27	2.0004	1.0428	0.9999
9	(1, 7)	14	2.1934	1.0312	0.9979
	(7, 1)	32	2.0157	1.2261	0.9999
10	(1, 4)	8	1.0998	2.0223	0.9997
	(4, 1)	5	1.0982	2.2054	0.9980

5.4.2 Simulation Results – Broadcast Reliability

Table 5.3 collates the value of $R(v_j)\ \forall\ v_j \in V$ as well as $R_2(K)$ for $|K| = |V|$, for each of the ten TAG examples considered in Table 3.6. The rest of the data in Table 5.3 is mere replica of Table 4.4 and has been included for the readers' ease of understanding of various correlations. The results for $R(v_j)$ and $R_2(K)$ are produced using the approach discussed in Section 5.3.1 and Section 5.3.2, respectively, and considering $p_{e_{a,b}^t} = 0.90$ for each TSE $e_{a,b}^t$. Note that $R_1(K)$ results have not been computed due to the intractability of calculation. Recall from proposition 5.1 that a higher $|\text{TVA}_j|$ for a root node v_i as compared to v_j does not guarantee $R(v_i) > R(v_j)$. For example, in TAG #1 $|\text{TVA}_1| > |\text{TVA}_3|$, but $R(v_1) < R(v_3)$. Notice that here the difference in reliability appears at the fourth decimal place, and might be insignificant for the given sets of values and considered network parameters. However, this result is consistent with Proposition 5.1. Apart from reliability, larger number of tv-arborescences would in general also increase the broadcast throughput.

Table 5.3. Results of reliability metrics obtained from ten arbitrary TAGs of Table 3.6.

Metric	**Root node**						
	$\boldsymbol{v_1}$	$\boldsymbol{v_2}$	$\boldsymbol{v_3}$	$\boldsymbol{v_4}$	$\boldsymbol{v_5}$	$\boldsymbol{v_6}$	$\boldsymbol{v_7}$
TAG #1: $\|A_j\| = 21$, $\|TA_j\| = 364$, $\|E''\| = 30$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	107	186	85	118	31	--	--
$\boldsymbol{R(v_j)}$	0.9962	0.9988	0.9967	0.9985	0.9779	--	--
TAG #2: $\|A_j\| = 8$, $\|TA_j\| = 44$, $\|E''\| = 18$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	11	29	16	21	--	--	--
$\boldsymbol{R(v_j)}$	0.9776	0.9970	0.9951	0.9974	--	--	--
TAG #3: $\|A_j\| = 63$, $\|TA_j\| = 675$, $\|E''\| = 32$, $\boldsymbol{R_2(K)} = 0.9940$							
$\|TVA_j\|$	21	64	86	49	102	70	4
$\boldsymbol{R(v_j)}$	0.8998	0.9223	0.9430	0.8573	0.9508	0.9300	0.6430
TAG #4: $\|A_j\| = 40$, $\|TA_j\| = 262$, $\|E''\| = 26$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	71	95	119	87	85	--	--
$\boldsymbol{R(v_j)}$	0.9965	0.9960	0.9977	0.9888	0.9974	--	--
TAG #5: $\|A_j\| = 55$, $\|TA_j\| = 539$, $\|E''\| = 30$, $\boldsymbol{R_2(K)} = 0.9996$							
$\|TVA_j\|$	67	124	221	76	149	4	--
$\boldsymbol{R(v_j)}$	0.9817	0.9920	0.9948	0.9827	0.9758	0.7144	--
TAG #6: $\|A_j\| = 81$, $\|TA_j\| = 1230$, $\|E''\| = 32$, $\boldsymbol{R_2(K)} = 0.9999$							

$\|TVA_j\|$	70	319	217	191	354	95	--
$\boldsymbol{R(v_j)}$	0.9760	0.9990	0.9955	0.9726	0.9995	0.9919	--
TAG #7: $\|A_j\| = 41$, $\|TA_j\| = 648$, $\|E''\| = 30$, $\boldsymbol{R_2(K)} = 0.9964$							
$\|TVA_j\|$	60	94	21	54	56	57	17
$\boldsymbol{R(v_j)}$	0.9413	0.9580	0.7850	0.8828	0.8834	0.8717	0.8460
TAG #8: $\|A_j\| = 45$, $\|TA_j\| = 661$, $\|E''\| = 32$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	184	175	285	134	223	--	--
$\boldsymbol{R(v_j)}$	0.9997	0.9998	0.9999	0.9987	0.9999	--	--
TAG #9: $\|A_j\| = 368$, $\|TA_j\| = 12750$, $\|E''\| = 44$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	593	365	1944	1273	822	2589	1113
$\boldsymbol{R(v_j)}$	0.9973	0.9854	0.9994	0.9983	0.9988	0.9998	0.9991
TAG #10: $\|A_j\| = 16$, $\|TA_j\| = 51$, $\|E''\| = 18$, $\boldsymbol{R_2(K)} = 0.9999$							
$\|TVA_j\|$	33	30	18	16	--	--	--
$\boldsymbol{R(v_j)}$	0.9994	0.9993	0.9959	0.9879	--	--	--

-- Not Applicable

5.5 Summary

This chapter first introduced and evaluated, via SDP-MVI technique, network unicast related reliability metrics. More specifically, the chapter defined a technique to compute the 2-terminal reliability of device-to-device connected TVCNs, *i.e.,* $R(s,d)$, using all enumerated TS-MPS. Further, the chapter demonstrated that from the generated TS-MPS one can also segregate other possibilities of interest such as all TS-MPS having *earliest arrival date*, the *minimum number of hops*, and the *minimum delay* (time span) to compute the 2-terminal reliability associated with them. The chapter also provided the plots pertaining to the variation of 2-terminal reliability with change in the edge formation probability, edge communication probability, number of time slots and number of iterations. Additionally, the chapter introduced and computed a new network performance metric, *i.e.,* ESC (to find the expected delay in a TVCN) and evaluated an already existing and well-known metric, *viz.*, EHC.

The chapter also introduced and evaluated, via SDP-MVI technique, network broadcast related three reliability metrics, *viz.*, $R(v_j)$, $R_1(K)$ and $R_2(K)$, for TVCNs. The three aforementioned broadcast reliability metrics were computed using the generated tv-

arborescences. It was shown that computing $R_1(K)$ in general is intractable and $R_2(K)$ can be considered as the upper bound of $R_1(K)$. Ten different TAGs already presented in Chapter 3 were used for obtaining the values of all proposed metrics; the results have been collated in the form of tables.

Chapter 6 Management, Design and Upgrade of TVCNs

6.1. Introduction

Topology control, design and upgrade of communication networks by constructing a sparsely connected network of minimum cost has been a problem of great interest to the researchers. Authors in (Paik and Sahni 1995) studied the upgrading problem for several performance measures including delay on edge and the diameter of the resulting network. Usually, in conventional networks, upgrading a node (edge) corresponds to replacing the previous node (edge) with a superior quality node (edge). Such an upgrade reduces the communication delay along each edge emanating from the node. In general, there is a cost for improving each node (or edge) in the existing network by a unit amount (Krumke et al. 1998). Further, note that, in general, network upgrade for conventional networks is an NP-hard problem (Lim et al. 2005). Due to intermittent connectivity via contacts, topology control, design and upgrade in TVCNs is equally challenging.

This chapter considers only broadcast communication model and presents a simple technique for constructing sparse TVCN topology and designing a better contact plan. It also makes first attempt towards TVCN upgrade. More specifically, this chapter extends the findings in Chapter 4 and uses the notations and results provided there.

The chapter is organized as follows: In Section 6.1.1, we first propose four new metrics that can be used by TVCN designer/manager for topology control. Then we discuss some possible uses of the metrics. Section 6.2 and Section 6.3 describe a technique for topology control and contact plan design, respectively, in TVCNs. Section 6.4 first introduces the problem of network upgrade in TVCNs and then presents a greedy algorithm as a solution for network upgrade. Section 6.5 presents simulation results to show the efficacy of the proposals. Finally, Section 6.6 presents the summary of the chapter.

6.1.1 Proposed Metrics

The metrics depicting utilization of TSEs and t-arborescences, respectively, are discussed below in the ensuing paragraphs.

***1) Utilization of TSEs*:** Each root node $v_j \in V$ in TVCN only requires those TSEs which are present in at least one tv-arborescence in TVA_j to broadcast data from v_j. We denote by RQ_j the set of all *required* TSEs with respect to node v_j. Thus, except all TSEs in RQ_j, necessary for establishing connectivity from root node to all other nodes, other TSEs are not *required* and can be deliberately turned off. The switching off of the *non-required* TSEs generates a sparse network topology. This sparse topology will still be connected in time for data broadcast from root node but shall have less cost. Let $U_j(\text{TSE})$ be the percentage ratio between the total number of required TSEs with respect to root node $v_j \in V$ and all TSEs in E''. More specifically, we have

$$U_j(\text{TSE}) = \frac{|\text{RQ}_j|}{|E''|} \times 100\% \tag{6.1}$$

For example, consider the TAG shown in Figure 2.1(ii) having $|E''| = 10$. For root node v_1, we have $|\text{RQ}_1| = 4$, *viz.*, $e_{1,2}^1$, $e_{2,3}^1$, $e_{3,4}^4$ and $e_{2,3}^2$. The remaining six TSEs, *i.e.*, $e_{1,2}^3$, $e_{2,1}^1$, $e_{2,1}^3$, $e_{3,2}^1$, $e_{3,2}^2$ and $e_{4,3}^4$ are *non-required*. Thus, using Equation (6.1), we have $U_1(\text{TSE}) = 4/10 \times 100\% = 40\%$. For root node v_2, we have $\text{RQ}_2 = \{e_{2,1}^1, e_{2,3}^1, e_{3,4}^4, e_{2,3}^2, e_{2,1}^3\}$, and thus $U_2(\text{TSE}) = 50\%$. Similarly, we have $U_3(\text{TSE}) = 50\%$ because for root node v_3, 5 of 10 TSEs are required. However, we have $U_4(\text{TSE}) = 0\%$ because v_4 leads to no tv-arborescence. Therefore, v_4 cannot be chosen as root node. Note that $U_j(\text{TSE})$ with least percentage greater than zero value among all others obtained from different $v_j \in V$ is desirable, as it represents minimum cost of establishing connectivity from the specified root to all other nodes. It is important to note here that two TSEs, *viz.*, $e_{1,2}^3$ and $e_{4,3}^4$ have not been utilized by any root node among v_1, v_2 and v_3. Thus, regardless of root, these two can be switched off. We define $U_{All}(\text{TSE})$ as the percentage ratio between all required TSEs and all available TSEs in the network, given by

$$U_{All}(\text{TSE}) = \frac{|\cup_{j=1}^{|V|} \text{RQ}_j|}{|E''|} \times 100\% \quad (6.2)$$

Thus, for the example, we have $U_{All}(\text{TSE}) = 8/10 \times 100\% = 80\%$. Note that, $U_{All}(\text{TSE}) < 100\%$ denotes non utilization of some TSEs, thereby wastage of resources.

***2) Utilization of Spanning Arborescences*:** Recall that many t-arborescences of a TVCN can be invalid due to presence of non time-ordered edges. To assess how many of the enumerated arborescences are valid, we propose two new metrics. Let $U_j(\text{SA})$ defines the percentage ratio between the total number of tv-arborescences in TVA_j and the total number of t-arborescences in TA_j for each $v_j \in V$. Formally,

$$U_j(\text{SA}) = \frac{|\text{TVA}_j|}{|\text{TA}_j|} \times 100\% \quad (6.3)$$

For example, the TAG of Figure 2.1(ii) results in the $U_1(\text{SA})$ for node v_1 as $2/4 \times 100 = 50\%$, while the percentage for node v_2, v_3, and v_4 are 100%, 75%, and 0%, respectively. A large $U_j(\text{SA})$ value for a specific root node indicates the presence of more number of tv-arborescences from the specified root. It also reflects the better connectivity of root node with all other nodes. Further, the ratio can be utilized for an assessment of the effectiveness of contact allocations in a TVCN. In addition, we define $U_{All}(\text{SA})$ as the percentage ratio between the total number of tv-arborescences in TVA_{All} and the total number of t-arborescences in TA_{All}. Formally,

$$U_{All}(\text{SA}) = \frac{|\text{TVA}_{All}|}{|\text{TA}_{All}|} \times 100\% \quad (6.4)$$

The $U_{All}(\text{SA})$ for the TAG of Figure 2.1 (ii) is $9/16 \times 100\ \% = 56.25\%$. A large value indicates presence of large number of tv-arborescences in the network. It also depicts better utilization of the network resources and possibly a high broadcast reliability value.

6.2. Topology Control by Identification of Required Timestamped Edges

After generating TVA_{All}, a TVCN manager can construct a controlled topology with respect to any given root node $v_j \in V$ such that there exists at least one broadcast route from the specified root to all others. This requires examination of all $\text{TVA}_j^i \in \text{TVA}_j$ to ascertain all *required* TSEs in RQ_j and shutting of all others. Note that each TSE in RQ_j will be present in at least one TVA_j^i. Further, after finding all *required* TSEs in RQ_j a TVCN manager can use Equations (6.1) and (6.2) to calculate the $U_j(\text{TSE})$ for each root $v_j \in V$ and $U_{All}(\text{TSE})$, respectively.

6.3. Rearrangement of Timestamps on Each Edge of a TVCN

A TVCN's performance is dependent on the contact plan under use. An effective contact plan is expected to provide a large $\text{TVA}_j \; \forall \; v_j \in V$. A simple way to modify the contact plan being utilized by a TVCN is to shuffle the current timestamps from one pair of nodes to another without either eliminating or adding any extra timestamp. For instance, consider the TAG of Figure 2.1(ii) that has five potential bidirectional contacts. Now, consider that the timestamps have been shuffled across and the new contact plan after rearrangement of the original bidirectional contacts are $e_{1,2}^{\{1,4\}}$, $e_{2,3}^{\{1,3\}}$ and $e_{3,4}^{\{2\}}$. We need to identify among the two available contact plans, *i.e.*, original and rearranged, which is more effective. Thus, to assess the effectiveness of both contact plans, we calculate the utilization ratio of arborescences. More specifically, we calculate $U_j(\text{SA})$ and $U_{All}(\text{SA})$ for both cases using Equation (6.3) and (6.4), respectively. The $U_j(\text{SA})$ and $U_{All}(\text{SA})$ for the original contact plan have already been computed in Section 6.1.1. The rearranged timestamps result in the $U_j(\text{SA})$ for node v_1, v_2, v_3 and v_4 as 25%, 50%, 75% and 25%, respectively. The $U_{All}(\text{SA})$ for the modified scenario is 43.75%. It is important to note here that the $|\text{TA}_j|$ for each node $v_j \in V$ in the rescheduled case will be same as is obtained with the original network. This happens because we have assumed bidirectional contacts, and reallocated them entirely from one edge to another. Furthermore, it can be observed that after the rearrangement of

the timestamps, the U_j(SA) for nodes v_1 and v_2 decreased, for node v_3 it remained same and for node v_4 it increased. It is also apparent that the U_{All}(SA) figure in the new case has dropped from before. Thus, after carefully examining the values of the two ratios obtained from both contact plans, a designer can infer that TVCN is globally connected with the rearranged contacts, *i.e.*, it has at least one tv-arborescence for each node of the network. It can be observed that the second contact plan is fair to each node, while the first provides higher U_{All}(SA) percentage. Therefore, a decision about *which allocation of contacts is better* needs a trade-off between U_j(SA) and U_{All}(SA) as per requirements. From the discussion above, it is evident that the presented ratios, thereby approach, can assist a network designer to design or choose a better contact plan among alternatives.

6.4. TVCN Upgrade by Addition of Timestamp(s)

A loss of connectivity between any two locations in a communication network represents significant loss of revenue and credibility of the service provider. Therefore, in communication networks, it is often desirable to enhance the network survivability. Authors in (White 2002) addressed this problem by "adding an edge" into the network. However, such an upgrade requires extra cost, and hence upgrading the wrong links or nodes, or inserting new links in the wrong location may waste resources.

In line with the above discussion, a TVCN connectivity and thereby reliability can be enhanced by addition of timestamp(s), *i.e.*, new bidirectional contact(s). The addition of timestamp(s) is equivalent to the addition of data mule(s) to the network at strategic locations within the given period of the network. However, it is imperative to decide where to add the timestamp(s) before making the actual addition. This section considers a *Max-Min* problem, wherein we aim to maximize $|\mathrm{TVA}_j|$ from any root node $v_j \in V$ having the minimum number of tv-arborescences, denoted by $|\mathrm{TVA}_{min}|$. Note that the corresponding root node $v_j \in V$ and set TA_j are denoted by v_{min} and TA_{min}, respectively. More specifically, the problem aims to maximize $|\mathrm{TVA}_{min}|$ by addition of at maximum $\mathbb{N}$ timestamp(s) to the TVCN. Note that additional timestamp(s) can be utilized only if the network is *unsaturated* and the timestamp(s) required to maximize $|\mathrm{TVA}_{min}|$ is no larger

than $\mathbb{N}$. A network is *saturated* if each of its edges contains all possible timestamps within the given period of network. Thus, we can keep upgrading the TVCN to maximize $|\text{TVA}_{min}|$ unless the available number of timestamps, *i.e.*, $\mathbb{N}$ get exhausted or the network itself becomes saturated.

Recall that in general, network upgrade is an NP-hard problem. Thus, we propose a greedy approach for the TVCN upgrade; see Algorithm 6.1. The input to the approach is TA_{All}, TVA_{All}, and the maximum number of timestamps, *i.e.*, $\mathbb{N}$, to be added.

Algorithm 6.1. TVCN_Upgrade

1) Find TVA_{min}.
2) Generate $\text{TIA}_{min} = \text{TA}_{min} - \text{TVA}_{min}$. Thus, TIA_{min} contains all invalid t-arborescences.
3) Greedily find any $\text{TIA}_{min}^{k} \in \text{TIA}_{min}$ that can be made valid by addition of minimum number of timestamp(s) to the network.
4) After addition of the timestamp(s) found in Step 3) to the network, use Method 1 described in Chapter 4 to obtain updated TA_{All} and TVA_{All}.
5) Repeat from Step 1) until all $\mathbb{N}$ timestamps have been added, or no more timestamps can be added to the network because each edge has become saturated.

In essence, the five steps of Algorithm 6.1 iteratively aim to increase the number of tv-arborescences in TVA_j, for any $v_j \in V$, that has the minimum number of tv-arborescences.

6.5. Simulation Results

The greedy algorithm presented in this chapter has been implemented at the facility available at Curtin University by using Java code supported on JDK 8 and above. The tests were run on IntelliJ IDEA Community edition version 2018.3.6 running on PC with the following configurations: (1) Processor: Intel (R) Core (TM) i7-7700 CPU @ 3.60 GHz,

(2) RAM: 16.00 GB, and (3) System Type: 64-bit Operating System, x64-based processor. To analyze the performance of our proposed algorithm, we used the same ten arbitrarily generated TAGs shown earlier in Table 3.6. For each of the ten TAGs, Table 6.1 presents $|\text{TA}_j|\ \forall\ v_j \in V$, as well as the total number of TSEs in each network, *i.e.*, $|E''|$; see column 1. Table 6.1 also collates percentage value of metrics $U_j(\text{SA})$ and $U_j(\text{TSE})$ for each $v_j \in V$. In addition, it presents $U_{All}(\text{SA})$, and $U_{All}(\text{TSE})$ for each TAG.

Table 6.1. Simulation results of different metrics obtained from ten arbitrary TAGs of Table 3.6.

TAG #	Metric	Root node							*All*
		v_1	v_2	v_3	v_4	v_5	v_6	v_7	
1 $\|\text{TA}_j\| = 364$ $\|E''\| = 30$	$U_j(\text{SA})$	29.39	51.09	23.35	32.41	8.51	--	--	28.95
	$U_j(\text{TSE})$	60	63.33	60	66.66	63.33	--	--	100
2 $\|\text{TA}_j\| = 44$ $\|E''\| = 18$	$U_j(\text{SA})$	25	65.90	36.36	47.72	--	--	--	43.75
	$U_j(\text{TSE})$	66.66	72.22	61.11	61.11	--	--	--	100
3 $\|\text{TA}_j\| = 675$ $\|E''\| = 32$	$U_j(\text{SA})$	3.11	9.48	12.74	7.25	15.11	10.37	0.59	8.38
	$U_j(\text{TSE})$	53.12	59.37	53.12	50	59.37	59.37	25	93.75
4 $\|\text{TA}_j\| = 262$ $\|E''\| = 26$	$U_j(\text{SA})$	27.09	36.25	45.41	33.20	32.44	--	--	34.88
	$U_j(\text{TSE})$	76.92	76.92	80.76	88.46	76.92	--	--	100
5 $\|\text{TA}_j\| = 539$ $\|E''\| = 30$	$U_j(\text{SA})$	12.43	23	41	14.10	27.64	0.74	--	19.82
	$U_j(\text{TSE})$	66.66	63.33	66.66	50	63.33	26.66	--	93.33
6 $\|\text{TA}_j\| = 1230$ $\|E''\| = 32$	$U_j(\text{SA})$	5.69	25.93	17.64	15.52	28.78	7.72	--	16.88
	$U_j(\text{TSE})$	43.75	75	71.87	68.75	75	71.87	--	100
7 $\|\text{TA}_j\| = 648$ $\|E''\| = 30$	$U_j(\text{SA})$	9.25	14.50	3.24	8.33	8.64	8.79	2.62	7.91
	$U_j(\text{TSE})$	70	66.66	43.33	53.33	53.33	63.33	50	100
8 $\|\text{TA}_j\| = 661$ $\|E''\| = 32$	$U_j(\text{SA})$	27.83	26.47	43.11	20.27	33.73	--	--	30.28
	$U_j(\text{TSE})$	78.12	75	75	71.87	78.12	--	--	100
9 $\|\text{TA}_j\| = 12750$ $\|E''\| = 44$	$U_j(\text{SA})$	4.65	2.86	15.25	9.984	6.45	20.305	8.73	9.747
	$U_j(\text{TSE})$	72.72	68.18	72.72	72.72	68.18	77.27	75	100
10 $\|\text{TA}_j\| = 51$ $\|E''\| = 18$	$U_j(\text{SA})$	64.70	58.82	35.29	31.37	--	--	--	47.54
	$U_j(\text{TSE})$	72.22	72.22	66.66	50	--	--	--	100

For TAG #1, root node v_2 has U_2(SA) = 51.09, meaning more than 51% of its t-arborescences are tv-arborescences. In contrast, root node v_5 has U_5(SA) = 8.51, which is significantly smaller as compared to that for root v_2. Recall that larger U_j(SA) indicates more effective contacts assignment in the network with respect to node v_j, thereby producing more tv-arborescences from the node. It is worth mentioning that if one can increase U_i(SA) for a given node v_i, *e.g.*, by adding more timestamps, it will always increase broadcast throughput. As discussed in Section 6.4, a network designer can add more timestamps or rearrange timestamps to increase U_j(SA). Note that each TAG of Table 3.6 is globally connected as there is at least one tv-arborescence from each node. This can be verified from non-zero values of U_j(SA) for different nodes in each example. In terms of U_{All}(SA), shown in the last column, TAG #7 has the smallest value, *i.e.*, 7.91%, meaning its current contact plan is less efficient as compared to the other TAGs. More specifically, while TAG #7 contains total 4536 t-arborescences, only 359 of them are valid. Table 6.1 also shows the value of U_j(TSE) for each root node $v_j \in V$ for all TAGs. Assuming unit cost of each TSE, node with least U_j(TSE) will require least cost to maintain connectivity to all other nodes. If the motive is to find the best root location in a topology controlled TVCN, such that it has least cost and is sparsely connected, then node with least U_j(TSE) value, which is greater than zero, will be the best node for root location amongst all other nodes. However, it is important to note that two nodes in a TAG having the same value of U_j(TSE) do not guarantee the same U_j(SA). For example in TAG #10, nodes v_1 and v_2 have U_1(TSE) = U_2(TSE) = 72.22%, but U_1(SA) $\neq$ U_2(SA), because the utilized TSEs contribute in different t-arborescences and tv-arborescences. Further, a node v_i having larger U_i(TSE) as compared to another node v_k does not necessarily mean that U_i(SA) $>$ U_k(SA). For example in TAG #1, U_4(TSE) = 66.66%, while U_2(TSE) = 63.33%, but U_4(SA) $<$ U_2(SA). The reason is the presence of more multiple common TSEs in t-arborescences and tv-arborescences from one node as compared to the other. The value of U_{All}(TSE) depicts the total number of TSEs that will ever be utilized within the network. The results show that one can achieve about 6.5% cost saving for TAG #3 and #5 by shutting down all non-required TSEs in both TAGs. Note that all TSEs in the other eight TAGs are required.

6.5.1. Analysis of Greedy TVCN Upgrade

To assess the performance of our proposed greedy approach presented in Section 6.4, we plot in Figure 6.1 the impact of including $\mathbb{N} = \{1, 2, \dots, 7\}$ additional timestamps on the TAG of Figure 2.1(ii). Note that we cannot have $\mathbb{N} = 8$, as after addition of seven timestamps each edge of the network possesses maximum allowed timestamps, *i.e.*, four, therefore, it becomes saturated.

As shown in the Figure 6.1, also from Table 4.1, initially there is one root node, *i.e.*, v_4, that has zero tv-arborescences, with $|\text{TVA}_{All}| = 9$. This implies that initially $v_{min} = v_4$ and $|\text{TVA}_{min}| = 0$. The addition of one timestamp to the network by using the proposed greedy approach changes $|\text{TVA}_{min}|$ to three and $|\text{TVA}_{All}|$ to 18. Note that new v_{min} will not necessarily be v_4. Similarly, the addition of two, three, four and five timestamps result in $|\text{TVA}_{min}|$ equal to four, seven, nine and ten, and $|\text{TVA}_{All}|$ equal to 28, 44, 55 and 77, respectively. Note that when we increase the number of additional timestamps from five to six, the extra cost does not improve $|\text{TVA}_{min}|$. The reason is because after using the first five timestamps, TVA_{min} requires two new timestamps for its maximization. Thus, increasing the cost to seven timestamps results in $|\text{TVA}_{min}| = 20$ and changes the $|\text{TVA}_{All}|$ to 120.

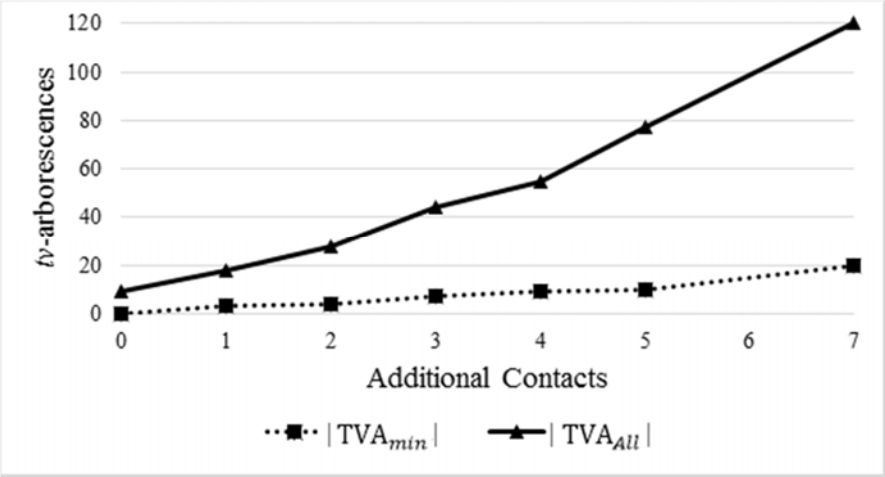


Figure 6.1. Impact of greedy network upgradation on the TAG of Figure 2.1(ii).

In summary, it is important to note here that the addition of a timestamp in the TVCN, for maximizing $|\text{TVA}_{min}|$, also positively impacts other nodes and thereby the broadcast routes originating from them. More specifically, each additional contact not only maximizes $|\text{TVA}_{min}|$, but also increases $|\text{TVA}_j|$ $\forall\, v_j \neq v_{min} \in V$ and $|\text{TVA}_{All}|$. Thus, the value of $|\text{RQ}_j|$, $R(v_j)$ for each $v_j \in V$ and $R_2(K)$ when $|K| = |V|$ always increases after addition of timestamp(s). However, the same cannot be said about U_j(SA) and U_{All}(SA), because the addition of timestamp(s) will not only increase $|\text{TVA}_j|$, $\forall\, v_j \in V$ and $|\text{TVA}_{All}|$, but will also escalate $|\text{TA}_j|$, $\forall\, v_j \in V$ and $|\text{TA}_{All}|$. Thus, if the additional timestamp(s) increases $|\text{TA}_j|$ significantly in comparison to $|\text{TVA}_j|$ $\forall\, v_j \in V$, then the U_j(SA) will worsen as compared to its original state. Similar reasoning also applies to U_{All}(SA).

In order to evaluate the effectiveness of our greedy method, we randomly added five new timestamps into the TAG of Figure 2.1(ii) of Chapter 2 and used Method 1 to find $|\text{TVA}_{min}|$ and $|\text{TVA}_{All}|$. We repeated the simulations 100 times with five new timestamps randomly deployed in each iteration. Among the 100 simulations, we find that the minimum $|\text{TVA}_{min}|$ is four and the maximum is 12, while the average is eight. Similarly, we find the minimum, average and maximum $|\text{TVA}_{All}|$ to be 56, 70 and 81, respectively. Note that, for the TAG, as reported previously, the value of $|\text{TVA}_{min}|$ and $|\text{TVA}_{All}|$ obtained using greedy approach is 10 and 77 respectively, which are higher than the average values of the respective terms obtained using random upgrade, *i.e.*, 8 and 70 respectively. Thus, for the TAG of Figure 2.1(ii), the greedy approach performs better than the random method. Figure 6.2 shows the impact of five extra timestamps added to each of the ten TAGs of Table 3.6 via proposed greedy approach and random upgrade. Figure 6.2 shows that although the greedy method is not optimal, yet, it performs better (*e.g.*, TAG #1 and #2) or comparable (*e.g.*, TAG #4 and #6) to the average case of random deployment of the timestamps. Thus, the proposed greedy technique can assist a TVCN manager or designer in making decisions regarding network upgrade for increasing the connectivity, thereby the reliability, survivability and quality of service, of the TVCN.

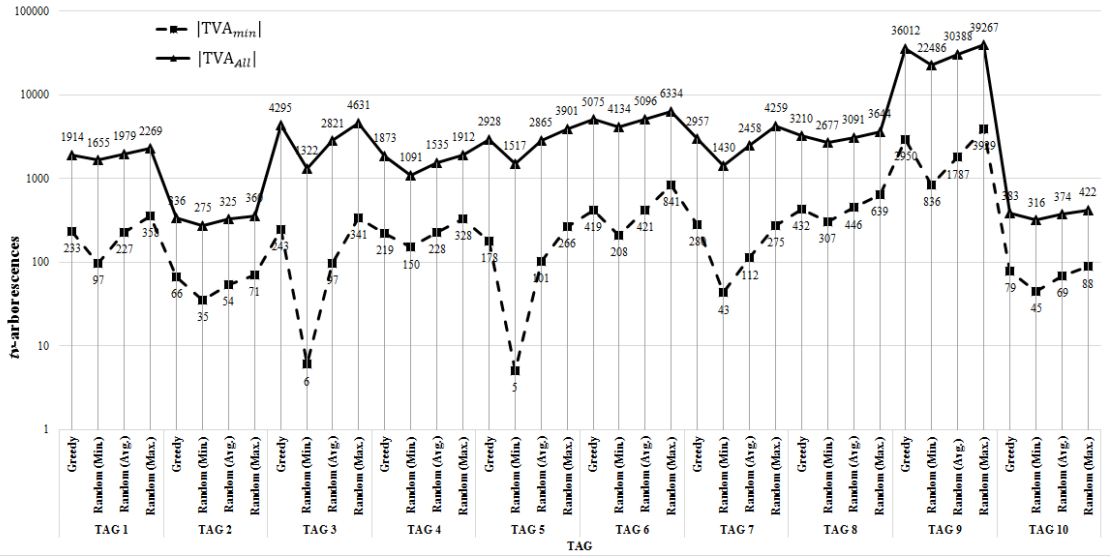


Figure 6.2. Impact of greedy and random network upgradation on the TAGs of Table 3.6.

6.6. Summary

This chapter presented some applications of the findings reported in Chapter 4 earlier. More specifically, using broadcast communication model, three important problems, namely, *topology control*, *contact plan design* and *network upgrade*, of TVCNs have been addressed in this chapter. For the first two problems the chapter has ferreted out four novel ratio metrics, *i.e.*, $U_j(\text{TSE})$, $U_{All}(\text{TSE})$, $U_j(\text{SA})$ and $U_{All}(\text{SA})$, that can serve as an indicator for an effective decision making. On the other hand, for the last problem the chapter has presented a novel greedy algorithm. Simulation results demonstrated that although the proposed greedy algorithm is not optimal, yet, in general it performs better than the average case of random network upgrade for the same problem. The discussions in this chapter seem to be of great utility to a TVCN designer for better utilization of network resources and for looking into feasible and alternative contact plans.

Chapter 7 Reliability Evaluation of End-to-End Connected TVCNs

7.1 Introduction

There are a plethora of algorithms/techniques available in the literature to evaluate the reliability of an infrastructure based network. Most of these algorithms have originated from graph theory and Boolean algebra, *viz.*, factoring theorem, enumeration methods, transformation methods, reduction and decomposition, BDD, direct methods and approximation method etc., (Misra 1993; Chaturvedi 2016). However, the direct applicability of these well-developed methods available in the literature become impractical for end-to-end connected TVCNs, *viz*., MANETs due to their special attributes like unpredictable topology, node mobility, phase transitions, frequent link formation and cessation. Recall that Chapter 5 of this thesis discussed some reliability metrics and their evaluation techniques for device-to-device connected TVCNs. On the contrary this chapter considers reliability evaluation of end-to-end connected TVCNs with node mobility.

To the best of our knowledge, authors in (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013) were the first to consider node mobility and its impact on network configuration(s) of MANETs in tandem with node and network reliability. More specifically, they proposed Monte Carlo Simulation based methods that account for node mobility, node reliability and node performance in the network's connectivity, thereby the resultant reliability of MANETs.

However, the aforementioned approaches overlooked or neglected the state of the network at times between intervals of observation under an assumption that the nodes shall remain connected during an interval $(t,\ t + \Delta t)$ and follow linear motion within this incremental duration (Δt). If this duration is significantly large then this assumption becomes a shortcoming as high node mobility would result in a frequent link formation and/or termination. Hence, a significant amount of information about topology change is

lost within the intermediate evaluation points. This aspect can be effectively dealt by considering Link Expiration Time (LET) (Su et al. 2001) and our proposed concept of Border Time (BT). Note that LET is a simple model which utilizes the mobility information (such as a node's speed, initial coordinates, and direction obtained from a GPS device) to predict the expiry time of links, *i.e.*, the time at which a link ceases to exist due to mobility. On the contrary, BT is the time at which any node reaches at the border of the coverage region. BT is evaluated to deflect the mobile nodes back into the bounded area. These two notions help in maintaining track of times (of all links) at which link breaks, so as to find the instants at which topology changes. The details on how to compute these parameters are discussed later in Section 7.2.

The other motivating factors behind this work (Khanna et al. 2019a) are: i) The Expected Number of Node Failure (ENOF) produced by the existing Monte Carlo Simulation based approaches (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013), turns out to be quite high that also does not conform to the theoretical results, and ii) Today's smart hand-held mobile devices customarily comprise of an integral GPS receiver. In the contemporary scenario, it seems plausible to utilize the GPS coordinates to define a realistic coverage area in which a network is formed and all nodes are bound to move.

Driven by the above factors, firstly, we define a coverage area with the help of latitude and longitude of two real locations (obtained from GPS), as it presents a more realistic and three-dimensional view of the surface of the Earth within which the mobile nodes move. Secondly, we convert these locations from geographic coordinates to their respective Cartesian coordinates and develop a method by considering LET and BT to capture all the topology changes during the complete mission time, T. This process overcomes the limitation of (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013), wherein the network topology was assessed (to find the changes in the network topology) at fixed intervals during a mission duration. Finally, we evaluate the 2-terminal reliability of a MANET, *i.e.*, R_{2TR_m} with the help of Monte Carlo Simulation technique.

The remaining chapter is organized as follows: Section 7.2 to Section 7.5 present the preliminaries required for better understanding of the proposed method. Section 7.6 presents the assumptions made in the proposed method of reliability evaluation. Section 7.7 presents the detailed methodology and a flowchart to clearly explain the steps involved in the whole process of 2-terminal reliability evaluation. Section 7.8 presents the obtained results and associated discussions. Finally, concluding summary along with the future scope of this work is presented in Section 7.9.

7.2 Preliminaries

Before discussing the details of our proposed approach, we provide some necessary preludes for better elucidation of our presented method.

7.2.1 Defining Coverage Area

Consider a section between two geographical coordinates on Earth's surface as shown in Figure 7.1 and Figure 7.2. Two known locations on the Earth's surface would result in either an area (shown in Figure 7.1(a)) or a straight line, involving two locations having either same longitude (φ) or latitude (ψ) (as shown in Figure 7.1(b) and (c), respectively).

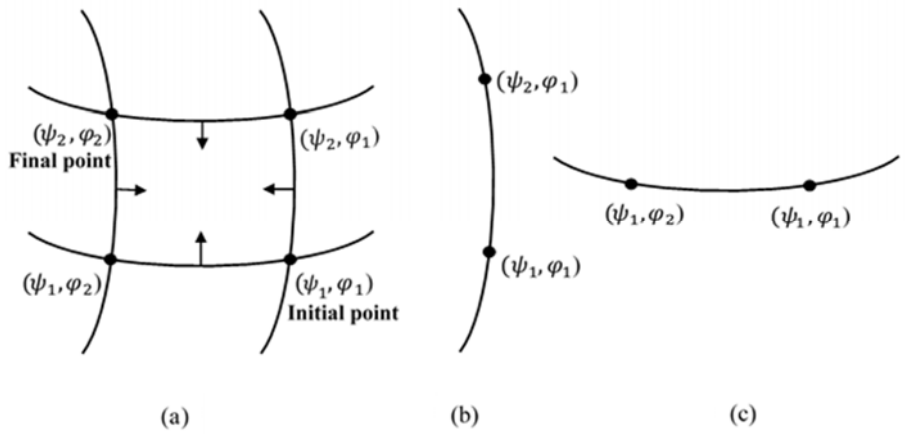


Figure 7.1. (a) A section on the surface of the Earth between two geographical coordinates representing initial point and final point (generating a region for observation), arrows pointing inwards show that all mobile nodes must lie within this inner region; (b) and (c) Another possibility with two locations having either same longitude or latitude, respectively.

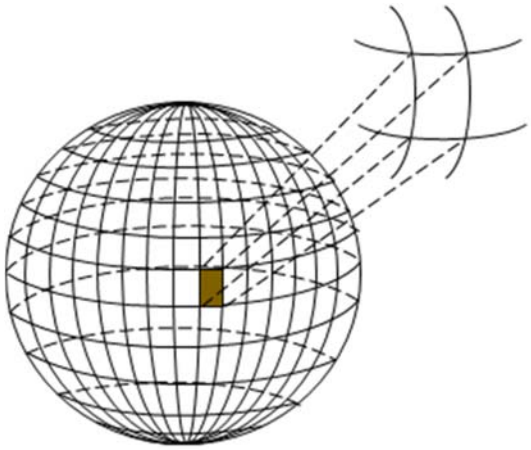


Figure 7.2. A section on the surface of the Earth that has been enlarged to visualize the region of interest in which mobile nodes have been deployed.

The region shown in Figure 7.2 can be considered to be made up of straight lines instead of curved lines because of the infinitesimal area in comparison to the size of the Earth. Besides, as the nodes move on the surface of the Earth within a small region of interest, hence, the altitude is neglected.

The data set or tuple <*latitude, longitude, time*> (collected from GPS devices) of the two points defining the boundaries of the coverage area can be converted from its geographical coordinates (latitude/longitude) into its respective two-dimensional Cartesian coordinates (x, y) as (Zaidi and Mark 2011):

$$x = \mathcal{R}\frac{\pi \cos(lat_0)}{180}(long_0 - longitude) \tag{7.1}$$

$$y = \mathcal{R}\frac{\pi}{180}(latitude - lat_0) \tag{7.2}$$

where, $\mathcal{R}$ is radius of the Earth (in kilometers) and $(lat_0, long_0)$ correspond to the latitude and longitude (in decimal) of the reference point (0, 0) on the local Cartesian coordinate system.

In other words, we can say that from the two known locations obtained from GPS within which we are to deploy an ad hoc network, a coverage area can be defined as shown in Figure 7.1(a), and can be converted to the Cartesian coordinate system as shown in Figure 7.3, by using Equation (7.1) and (7.2) wherein x_{min} and y_{max} are the corner points relative to a reference point (*e.g.*, (0, 0)) in the rectangular and bounded slice. Besides, for any point within the coverage area, we can always use the two equations to convert its GPS location to its corresponding location in xy-coordinates with respect to a reference point.

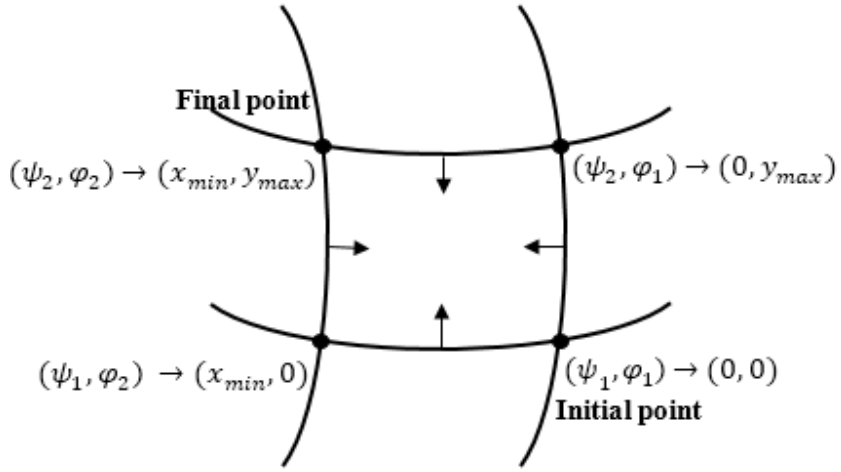


Figure 7.3. Illustration showing the conversion of coverage area coordinates from (latitude, longitude) to (x, y) coordinates.

7.2.2 Network Model

A MANET can be represented as a RGG $G(N, L)$, consisting of N mobile nodes and L number of communicating links. The topology between any time instant t to $t + \Delta t$, within the mission duration T, is fixed and Δt can be obtained on the basis of either LET or BT, explained later in Section 7.3.

It is assumed that nodes and links fail primarily due to factors like mobility, battery drain and/or decrease in signal strength with an increase in the distance from the transmitting node. We also assume that the time-to-failure of nodes follow a known failure distribution (*e.g.,* Weibull) and links form on the basis of the nodes' proximity and transmission range. Further, when the specified or designated nodes, $n \leq N$ are operational,

communication between them can be considered as a random event having probability $R_G(t)$. Therefore, for $n = 2$ nodes or for the communication to occur between a designated node pair (s, d), it is necessary that the pair must be operational; hence, the reliability of the network, *i.e.*, $R_{2TR_m}(t)$ is equal to the product of the reliability of (s, d) node pair and the reliability of the rest of the network with perfect (s, d) pair of nodes. The related mathematical formulae are discussed in the upcoming sections.

7.2.3 Mobility

In real scenario, one can assume the availability of GPS receiver within the mobile nodes that can provide the precise location of mobile nodes inside the coverage area along with their mobility information such as speed and direction of motion. This information provided by the GPS (also known as traces) can be used each time after Δt, to obtain the current location, speed and direction of movement of mobile nodes. This can be helpful in simulating the mobility with real trace. However, for the sake of simulation and incorporating the human mobility, here, we randomly assigned a speed between [V_{min}, V_{max}] to all mobile nodes and direction of node motion, δ, between [0, 2π]. If the present location of a node u_i at time instant, t, is $(x_i(t), y_i(t))$, then its new location after an incremental time, Δt, is determined using formula given in Equation (7.3).

$$\left.\begin{aligned} x_i(t + \Delta t) &= x_i(t) + \Delta t * v_i(t) cos\delta_i(t) \\ y_i(t + \Delta t) &= y_i(t) + \Delta t * v_i(t) sin\delta_i(t) \end{aligned}\right\} \tag{7.3}$$

where, $v_i(t)$ denotes the speed and $\delta_i(t)$ represents the direction of motion, respectively, of the i^{th} node at time instant t.

7.2.4 Node and Link Status

Noting that a mobile node that survives (as no repair is possible for the failed node) an elapsed time $t <$ T would only be capable of carrying on the mission for a next Δt time as decided on the basis of LET and/or BT (refer Figure 7.4).

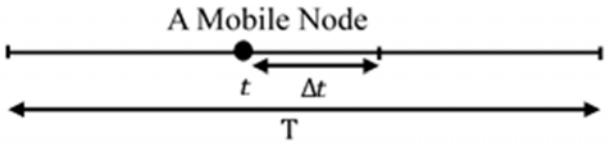


Figure 7.4. Simulating the status of a mobile node.

Hence, the reliability of a mobile node conditioned on the survival up to time t is given by:

$$R(\Delta t|t) = \frac{R(\Delta t + t)}{R(t)} \tag{7.4}$$

The mobile nodes present in the network are assumed to follow Weibull failure distribution, hence, status of node u_i at time $(t + \Delta t)$ can be simulated as:

$$P(u_i(t + \Delta t) = 1) = \begin{cases} 1, & if\ U(0,1) \leq \dfrac{e^{-\left(\frac{t+\Delta t}{\eta}\right)^{\xi}}}{e^{-\left(\frac{t}{\eta}\right)^{\xi}}}, \quad \text{node } u_i \text{ functions} \\ 0, & \text{node } u_i \text{ fails} \end{cases} \tag{7.5}$$

where, η is *scale* and ξ is *shape* parameter of the Weibull distribution. It is worth to note here that for a fixed value of η, different values of parameter ξ affect the slope thereby shape of the Weibull reliability function. More specifically, the value of reliability function decreases sharply and monotonically for $0 < \xi < 1$ and is convex. For $\xi = 1$, reliability decreases monotonically, but less sharply than for $0 < \xi < 1$ and is convex. While, for $\xi > 1$, reliability value decreases as the time increases, but less sharply; however, as wear-out sets in, the curve goes through an inflection point and decreases sharply (Kececioglu 2002). Figure 7.5 shows the Weibull reliability plot with $0 < \xi < 1, \xi = 1$, and $\xi > 1$.

.

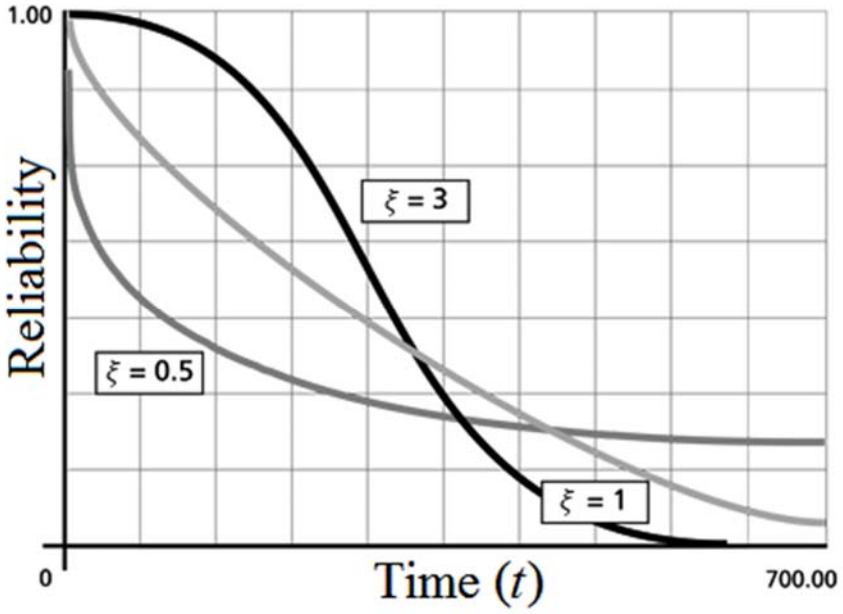


Figure 7.5. Impact of different values of ξ on the Weibull reliability function.

We can determine the links' status by computing the Euclidean distance, d_{ij}, between a pair of nodes (u_i, u_j) at time, t, using Equation (7.6) and compare it with transmission range, r_i (or r_j), of mobile nodes for assessing link status, $L_{ij}(t)$ as in Equation (7.7).

$$d_{ij}(t) = \sqrt{(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2} \tag{7.6}$$

$$L_{ij}(t) = \begin{cases} 1, \text{if } d_{ij}(t) \leq r_i \text{ (or } r_j) \\ 0, \text{otherwise} \end{cases} \tag{7.7}$$

The evaluation of Euclidean distance is valid as per our assumption of a small region of interest spreading approximately, say, across a city; besides, the assumption can be easily relaxed in case of large coverage region by calculating great circle distance between each pair of nodes, computed by using either Haversine formula (Misra and Rajesh 2011) or Vincenty's inverse formula (Vincenty 1975). Hence, at any time, t, the network can be represented by a connection matrix of size $N \times N$, with its elements being $L_{ij}(t)$, *i.e.*, either 0 or 1. The data stored in this connection matrix is used to determine network

connectivity between (s, d) node pair. The network connectivity at any time, t, is '1', if connectivity exists between a specified (s, d) node pair; otherwise it is set to '0'.

7.3 Determining Link Expiration Time, Border Time and Incremental Time

We can view a MANET as a static network up to the time instant from where the topology change can be predicted/assessed by using the concept of LET and BT. We presume that there would be a significant loss of intermediate reliability information if we ignore this phenomenon. Therefore, these concepts are useful to find the precise incremental time at which the network topology changes occur. In the following, we describe the above stated two concepts in detail.

7.3.1 Link Expiration Time

The LET indicates the time for which a link will continue to exist between a pair of nodes. The formal definition of the LET model can be stated as follows.

Definition 7.1: LET (Su et al. 2001) expresses the length of time for which two mobile nodes, u_i and u_j, which are within the transmission range, r, of each other and already have known initial location: (x_i, y_i) and (x_j, y_j), velocity: v_i and v_j and phase angle: δ_i and δ_j $(0 \leq \delta_i, \delta_j \leq 2\pi)$, respectively, will remain connected.

The LET between two mobile users, u_i and u_j, can be predicted as (Su et al. 2001):

$$\text{LET}_{ij} = \frac{-(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}}{(a^2 + c^2)} \tag{7.8}$$

where

$$a = v_i \cos\delta_i - v_j \cos\delta_j$$

$$b = x_i - x_j$$

$$c = v_i \sin\delta_i - v_j \sin\delta_j$$

$$d = y_i - y_j$$

Note that if $v_i = v_j$ and $\delta_i = \delta_j$, then a and c become zero. For this case, $\text{LET}_{ij} = \infty$, *i.e.*, the nodes will remain connected throughout their existence. Although, a node pair may have a large LET value, owing to their mobility any one or both of them may reach the boundary within a time less than LET. In other words, within this large LET duration of link existence, node(s) may cross the boundary of the coverage area. So, while performing simulation we need to find another parameter, *viz.*, BT to confine and simulate the node movement within the coverage area.

7.3.2 Border Time

The MANET topology is dynamically changing with time owing to the mobility of nodes. Note that due to this mobility, some nodes might arrive at the boundary of the coverage region. In order to bring the nodes back inside and within the defined coverage area, a border rule has to be defined to simulate movement of mobile nodes in the bounded region. Here, we define BT, which represents the time at which any node will reach the boundary of the coverage region and is calculated by using the following four steps.

1) Find the direction/phase angle of each node moving inside the defined coverage area.

2) From the direction of motion of each node, (i) identify the quadrant in which a node is moving, and (ii) estimate the probable boundary towards which the node is moving.

   More specifically, for (ii), if the motion is within (0 to $\frac{\pi}{2}$), a node will reach one of the boundary which is common to the point (x_{max}, y_{max}). On the other hand, if the motion is within ($\frac{\pi}{2}$ to π), a node will reach one of the boundary which is common to the point (x_{min}, y_{max}). Similarly, if the motion is within (π to $\frac{3\pi}{2}$), a node will reach one of the boundary which is common to the point (x_{min}, y_{min}) and if a node is moving within ($\frac{3\pi}{2}$ to 2π), then a node will reach one of the boundary which is common to the point (x_{max}, y_{min}); please see Figure 7.6.

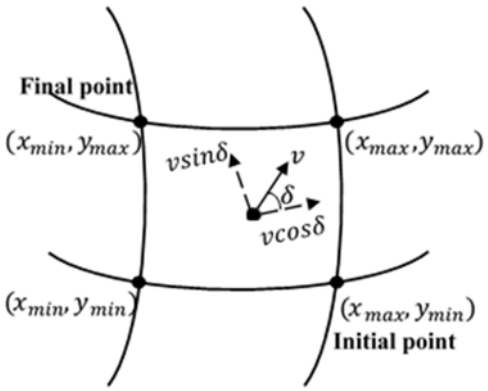


Figure 7.6. Illustration indicating a node moving towards boundary common to point (x_{max}, y_{max}).

3) Calculate the minimum time (amongst $t_{(x)}$ and $t_{(y)}$) at which each node reaches the boundary (either abscissa or ordinate) using Equation (7.9).

$$t_{(x_i)} = \frac{(x_{boundary_i} - x_i)}{v_i cos\delta_i} \text{ and } t_{(y_i)} = \frac{(y_{boundary_i} - y_i)}{v_i sin\delta_i} \qquad (7.9)$$

where, (x_i, y_i) denotes the coordinates of node u_i and $(x_{boundary_i}, y_{boundary_i})$ represents the coordinates of the common point at the intersection of the boundaries towards which node u_i is moving.

4) Compute the BT of node u_i by taking the minimum of $t_{(x_i)}$ and $t_{(y_i)}$ that gives the time in which the node will reach the boundary, *i.e.*, BT = min$(t_{(x_i)}, t_{(y_i)})$.

To bring the mobile nodes back to the simulation zone when they touch the boundary of the coverage region after elapse of BT period, the node(s) reaching the boundary are deflected back into the bounded region along a direction opposite to the direction of original motion with the previously acquired speed. The new direction of motion can be computed by

$$\delta = (\delta + \pi)\, mod(2\pi) \qquad (7.10)$$

7.3.3 Determining Incremental Time

To determine the incremental time, Δt, at which topology change is expected to occur, we use the following steps:

1) Find the minimum among all calculated LETs and BTs, *viz.*, LET_{min} and BT_{min}, respectively.

2) Compute Δt using the following equation

$$\Delta t = \begin{cases} \text{BT}_{\text{min}}, & \text{if all nodes are isolated} \\ \min(\text{LET}_{\text{min}}, \text{BT}_{\text{min}}), & \text{otherwise} \end{cases} \quad (7.11)$$

A node u_i is an *isolated* node if it has *no connection* with all other nodes. For checking whether a node u_i is isolated, we first find its connectivity, by computing Euclidean distance, with all other nodes present in the network. If the distance between a node pair u_i and u_j is less than the transmission range of node u_i, then both u_i and u_j are connected to each other, otherwise they are disconnected. Next, we randomly assign speed between V_{min} and V_{max} (in km per hour) and direction (between 0 to 2π) to an isolated node moving within the defined coverage region.

7.4 Obtaining Same Number of Incremental Time Values in Each Iteration

References (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013) carried out the reliability evaluation of MANETs by utilizing Monte Carlo Simulation and considering a fixed incremental time, Δt, obtained by dividing the entire mission duration, T, into equal slices of Δt hour duration. This process generated a fixed, $\frac{\text{T}}{\Delta t}$, number of slices. To reiterate, this is done to realize the topology, thereby, connectivity between various node pair at a particular interval Δt. However, in our proposed approach, a different and random number of Δt during a mission time T will be generated in each iteration of Monte Carlo Simulation; moreover, any value of Δt in one iteration may not resemble the

corresponding Δt in any other iterations due to its dependency on either LET or BT. Thus, to obtain the same number of incremental time instants in every Monte Carlo Simulation iteration, wherein the value of any Δt in the first iteration is identical to the corresponding value of Δt in all other iterations, we use the following three steps.

1) Merge and sort all unique Δt obtained from different iterations.

2) Search for unique Δt(s) where the connection probability value is unavailable. Fill these void(s) with the connection probability value present at the preceding time. This is done due to the fact that within time t to $t + \Delta t$ the network topology remains same, thereby, leading to same reliability figure for all time instants between t and $t + \Delta t$.

3) Use the connection probability value from every iteration to calculate the mean reliability at each time instant.

7.5 2-Terminal Reliability Evaluation

The 2-terminal reliability, $R_{2TR_m}(t)$ of a MANET can be defined as a function of time, and is evaluated by averaging the connection probability (CP) results of simulation run at each incremental time (Δt), till it is less than or equal to mission time T (Chaturvedi and Padmavathy 2013). More specifically, the reliability is computed as

$$R_{2TR_m}(t) = \frac{R_s(t) \times R_d(t) \sum_{k=1}^{nIterations} \text{netConnectivity}}{nIterations} = \frac{\sum_{k=1}^{nIterations} CP_k(t)}{nIterations} \tag{7.12}$$

where, $R_s(t)$ and $R_d(t)$ are the reliability of source and destination nodes, respectively, at time t and netConnectivity is used to check the (s, d) connectivity status at time t.

7.6 Assumptions

The main assumptions of this approach are as follows:

1) Network is homogeneous and operational at the start of the mission time.

2) Free space propagation model is utilized, where the received signal strength solely depends on its distance to the transmitter; besides, environmental interference and fading are considered to be well within the limits, therefore their impact is not studied explicitly.

3) Nodes transmit data to their neighboring nodes at the same instant when they receive it from some other node without any delay. Besides, route discovery and route maintenance are performed without failure and instantaneously.

4) The creation and destruction of each link depends on the Euclidean distance (d_{ij}) between its end nodes and the transmission range (r) along with LET and BT.

5) Time instants of observations are obtained on the basis of LET_{min} and BT_{min}. Besides, within an incremental change in time (Δt), nodes do not fail and node speed and direction remains constant, however, this assumption can be relaxed if required.

6) All links are bidirectional without any constraint on their load carrying capacity.

7) Failure of nodes are statistically independent and once a node fails, it remains failed for the remaining period of the mission duration.

7.7 The Methodology Proposed

Our proposed methodology follows the following six major steps:

Step 1: Use GPS coordinates (latitude and longitude) to initialize the coverage area.

Step 2: Select a reference location, and assign its xy-coordinates as (0, 0). Convert the coverage area in Cartesian coordinate system from the GPS coordinates of MANET deployment with respect to the reference point.

Step 3: Deploy the mobile nodes in the area in a random fashion. Simulate the initial position, speed and direction of each mobile node.

Step 4: Form an ad hoc network with $n \leq N$ perfect nodes, whose chances of failure during the mission duration are almost nil (*i.e.*, 100% reliable). Next, the converted network is analysed for its (s, d) connectivity by simulating node status, positions of nodes and movement of nodes around the simulation boundary, and determination of links between nodes.

Step 5: Compute LET_{min}, BT_{min}, and incremental time Δt.

Step 6: Repeat Step #4 and #5 at each incremental time, Δt, which is obtained on the basis of LET and BT, till the specified mission time, T, thus becoming one complete iteration (k) out of a total of number of simulation runs (nIterations).

A flowchart is presented in Figure 7.7 to clearly explain the steps involved in the whole process of R_{2TR_m} evaluation (Khanna et al. 2019a).

7.8 Simulation Results

We have implemented our proposed methodology by using MATLAB® version 2014a running on PC with the following configurations: (i) Processor: Intel (R) Core (TM) i5-6500 CPU @ 3.20 GHz, (ii) RAM: 4.00 GB, and (iii) System Type: 64-bit Operating System, x64-based processor. For assessing the performance of our proposed methodology, we define a coverage area between two locations *viz.*, 26.4499° N, 80.3319° E and 26.53° N, 80.42° E, lying approximately 12.5 km apart and forming a coverage area of 78.294 km^2. We assume a deployment of 25 mobile nodes equipped with identical man-portable radios having a transmission range of 3 km, moving within the region with a maximum and minimum velocity of 6 and 3 km per hour, respectively.

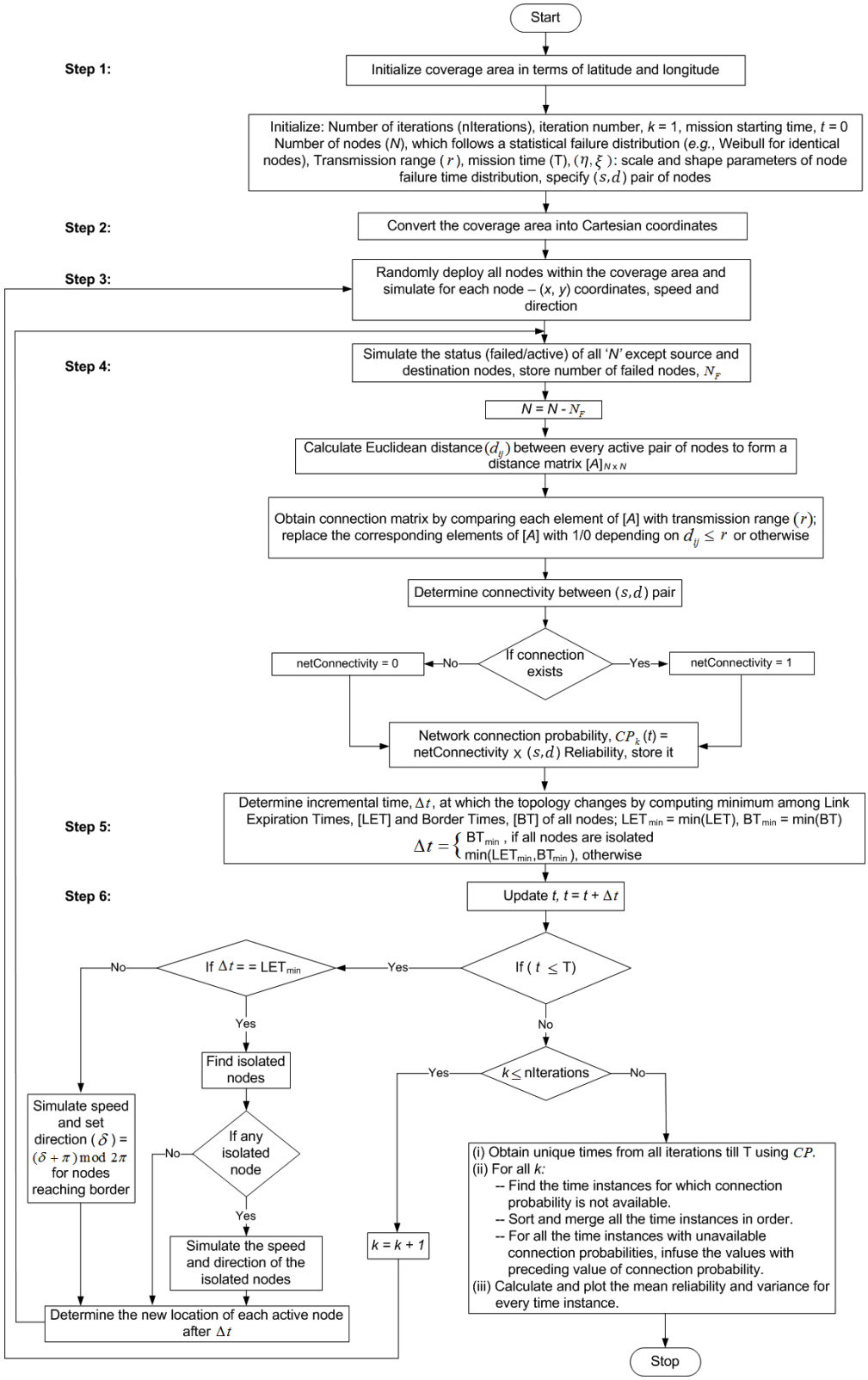


Figure 7.7. Flowchart of the proposed method of MANET reliability evaluation.

Besides, we also assume that the duration of mission is 72 hours and each node's time-to-failure distribution is governed by the Weibull distribution with parameter η = 500 hours and $\xi = 1.2$. We perform our simulation to evaluate the mean reliability of this network by varying number of iterations, coverage area, transmission range and number of nodes as an aid for strategist to select a network with the optimum design parameters during the deployment phase of the network. The obtained results are discussed in the following sub sections.

7.8.1 Effect of Iterations on 2-Terminal Reliability

To study the effect of number of iterations on R_{2TR_m} of the considered ad hoc network, for a mission duration of 72 hours with number of mobile nodes fixed at 25 and other parameters as already defined in Section 7.8, we varied the number of iterations at [2000, 3000, 4000, 5000, 6000]. From the results shown in Figure 7.8, it is observed that there is not much change in the predicted reliability once we change the number of runs from 4000 to 6000 and thus in each simulation we set runs to 5000.

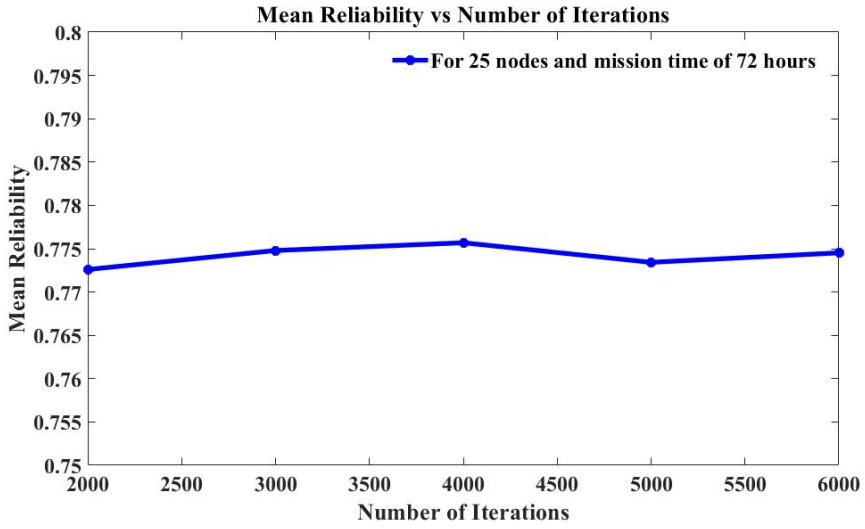


Figure 7.8. Effect of Number of Runs on Expected R_{2TR_m}.

7.8.2 Effect of Coverage Area on 2-Terminal Reliability

Here, we simulated the effect of varying coverage area, between different Geo-coordinates as shown in Table 7.1, on the evaluation of R_{2TR_m}. For this purpose, we keep the value of r fixed at 3 km, number of nodes fixed at 25 and number of runs fixed at 5000.

Table 7.1. Comparison of mean reliability and ENOF.

Initial Geo-Coordinates (A)	Final Geo-Coordinates (B)	Euclidian Distance (AB km)	Area (in sq. km)	Method in (Chaturvedi and Padmavathy 2013)		Proposed method (Khanna et al. 2019a)	
				ENOF	Mean reliability	ENOF	Mean reliability
26.4499° N, 80.3319° E	26.50° N, 80.40° E	8.774	37.7590	22.0862	0.4131	2.1604	0.8229
	26.51° N, 80.43° E	11.83	65.4133	22.0983	0.2615	2.1570	0.7988
	26.53° N, 80.42° E	12.500	78.2940	22.0909	0.2277	2.1398	0.7734
	26.49° N, 80.52° E	19.250	83.6866	22.1108	0.2118	2.1622	0.6023
	26.49° N, 80.58° E	25.090	110.3809	22.1132	0.1707	2.1380	0.4044

The result shown in Figure 7.9 indicates a sharp fall in R_{2TR_m} with an increase in coverage area. Note that with larger coverage area, the 25 nodes are expected to be positioned at longer distances, and hence less connectivity, among themselves. Consequently, there is expected to be less number of potential (s, d) paths, which reduces the network reliability.

The effect of coverage area on R_{2TR_m} for an entire mission duration of 72 hours is shown in Figure 7.10 where a consistent decrease in mean reliability with an increase in coverage area and mission time is very-well captured in our simulation results.

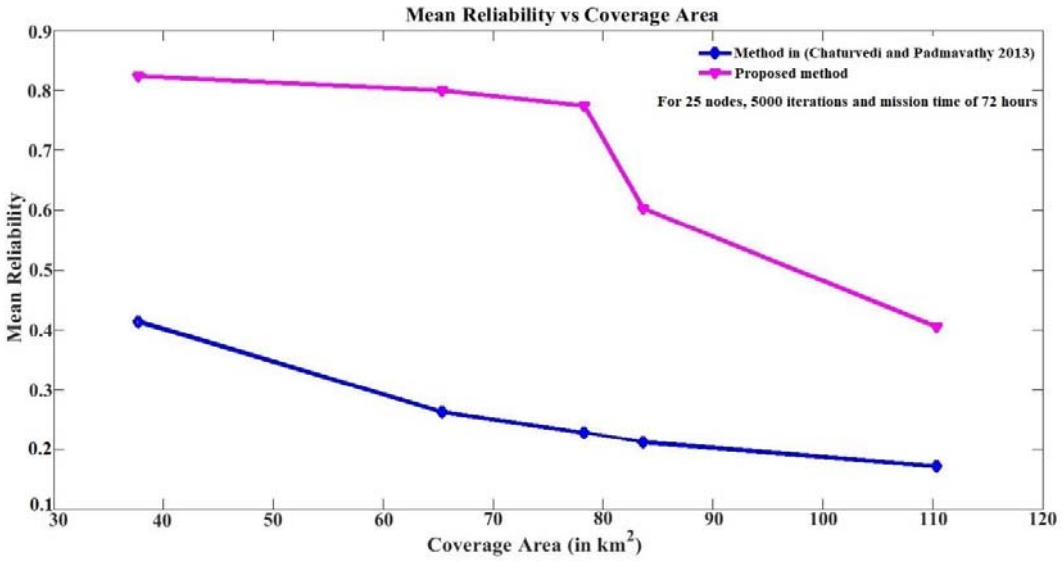


Figure 7.9. Effect of varying Network Coverage area on R_{2TR_m} evaluated using proposed method and method in (Chaturvedi and Padmavathy 2013).

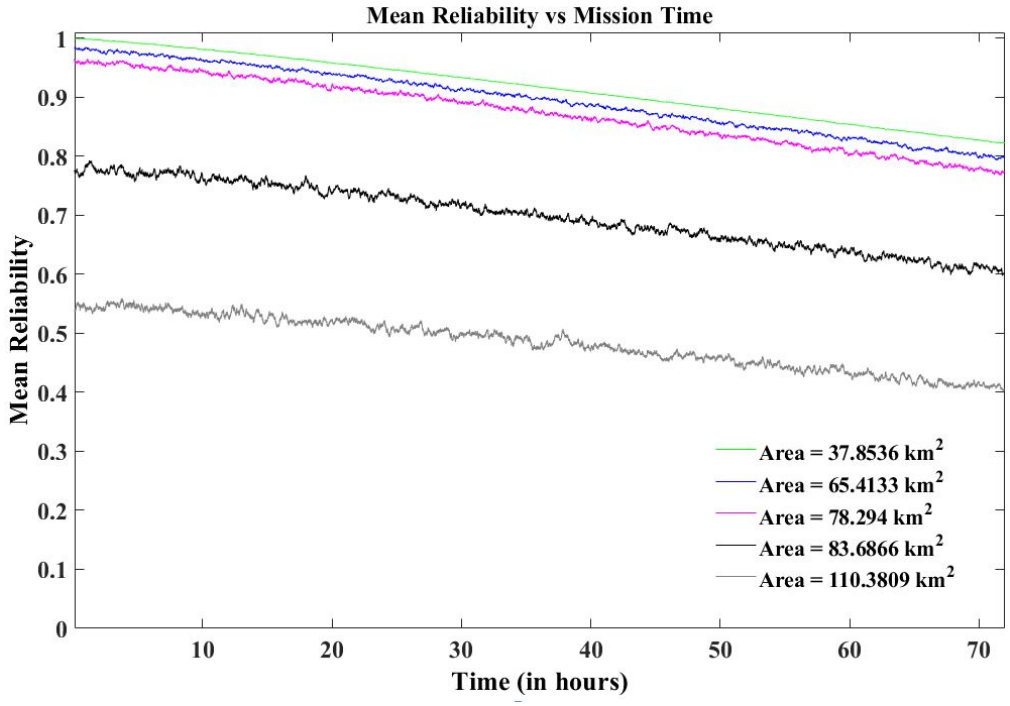


Figure 7.10. Effect of Varying Network Coverage Area on Expected R_{2TR_m} over the entire mission duration.

It can be very well observed from Table 7.1 that the earlier approaches, say (Chaturvedi and Padmavathy 2013), underestimate the network reliability due to large number of node failure (*i.e.*, ENOF) in their simulation, which should have been otherwise close to theoretical result of 2.32 (ENOF $= N(1 - \exp\left(-\left(\frac{t}{\eta}\right)^{\xi}\right) = 25(1 - \exp\left(-\left(\frac{72}{500}\right)^{1.2}\right)$). Further, for a given set of coordinates and range of velocities assumed in this simulation, the shape of the coverage region changes from Square to Rectangle with length >> width; also, the Euclidean distance and the area of the region almost gets tripled. The possible reasons for a sharp decline in reliability could be: (i) now nodes are becoming much distant apart, and (ii) the nodes hitting the boundary of the later region more frequently than in other regions. This may be creating a node becoming isolated making it incommunicable with the rest due to its short transmission range (3 km).

7.8.3 Effect of Varying Transmission Range on 2-Terminal Reliability

The simulation results obtained during the evaluation of mean R_{2TR_m} of a MANET by varying the value of transmission range r from 2 km to 4.5 km, while keeping the other input parameters as defined in Section 7.8, can be seen in Figure 7.11 and Figure 7.12.

Here, it can be clearly seen in Figure 7.11 that with the increase in r the mean reliability increases considerably. One can use the simulation as a guide in selecting the transmission range of the mobile nodes for a mission specific tasks. Note that the result is consistent with that in Section 7.8.2, since with fixed number of nodes, increasing transmission range has the same effect as decreasing coverage area, *i.e.*, both cases grow node connectivity which, in turn, provide larger number of (s, d) paths, and hence improve reliability. Further, for each r, reliability decreases with increasing mission time.

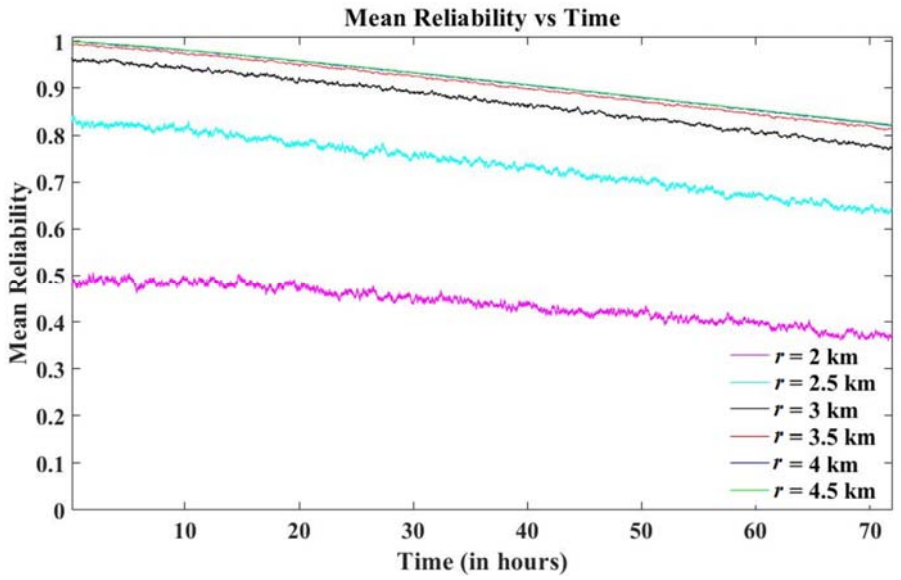


Figure 7.11. Effect of Varying Transmission Range on Expected R_{2TR_m} over entire mission duration.

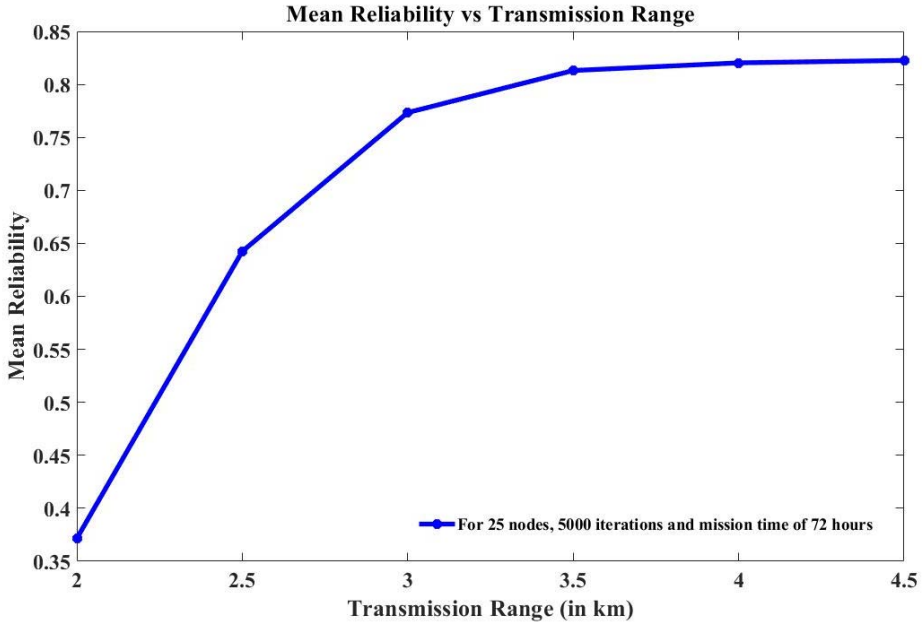


Figure 7.12. Effect of Varying Transmission Range on R_{2TR_m} for a 72-hours of Mission Time.

7.8.4 Effect of Varying Number of Nodes on 2-Terminal Reliability

We studied the impact of varying number of nodes on the network performance by setting the number of nodes at 15, 20, 25, 30, 35 and 40, while keeping the other input parameters as defined in Section 7.8. We observed that an increase in the number of nodes in the network increases the R_{2TR_m} due to the increase in the number of potential paths between (s, d) pair of nodes; see Figure 7.13 and Figure 7.14.

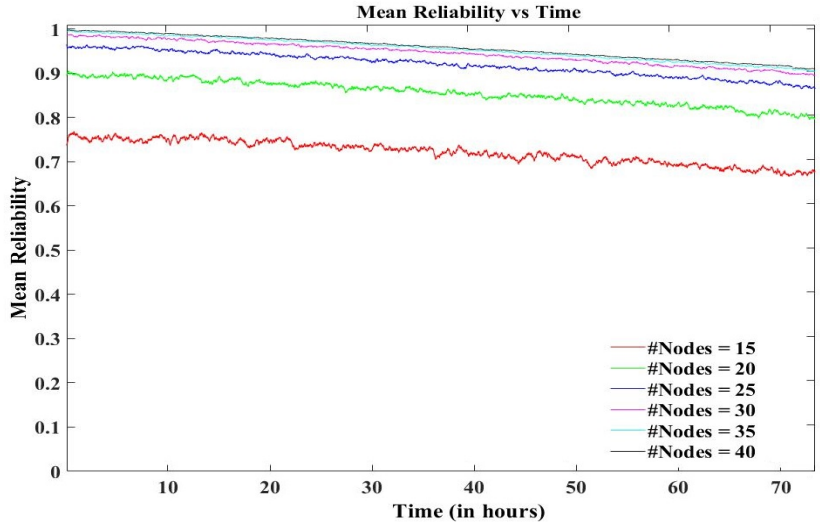


Figure 7.13. Effect of Varying Network Size on Expected R_{2TR_m} over 72-hr mission duration.

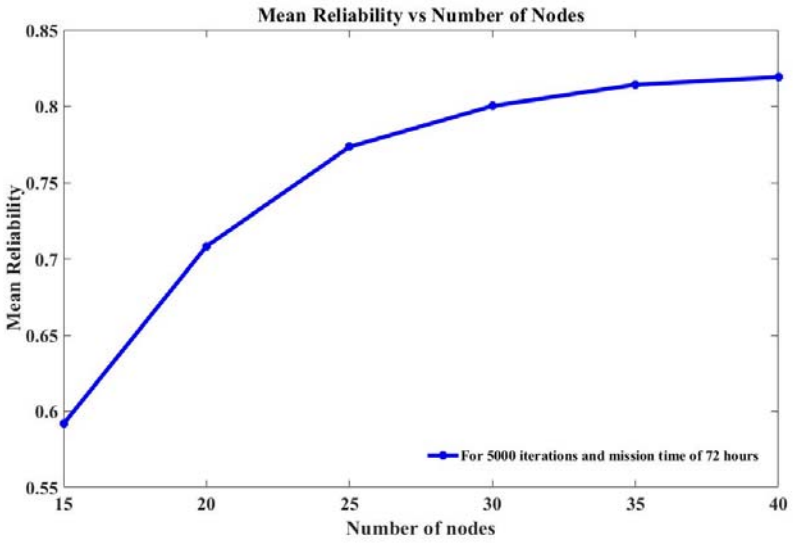


Figure 7.14. Effect of Varying Network Size on Expected R_{2TR_m}.

The result is again consistent with that in Section 7.8.2 with decreasing coverage area, and Section 7.8.3 when transmission range is increased, each of which improves reliability. Further, for each network size, the reliability decreases with longer mission duration, consistent with the results shown in Figure 7.10 and Figure 7.11. For a mission critical application that requires a minimum network reliability, one can use the results in the figures to determine the maximum mission duration for each network setup. For example, for the setup as defined in Section 7.8, *i.e.*, with coverage area 78.2940 km^2, 25 nodes, and r = 3 km, if the objective is to achieve a reliability of at least 0.84, then its mission duration should be no longer than 48 hours.

7.9 Summary

In this chapter, an approach considering LET and BT for link expiry modelling, and Monte Carlo Simulation has been described for the evaluation of reliability of MANETs. We also pointed out that the earlier approaches have overlooked the conditional survival of the mobile nodes executing a mission and hence provided erroneous results and underestimated the value of reliability. Further, we also provided the impact of various scenario metrics *viz.*, transmission range, number of nodes, coverage area and mission time, on MANET reliability, and found that by varying these parameters we can optimally select a MANET design to meet the reliability requirements. From our simulations, we also inferred that utilization of a fixed incremental time (as in (Cook and Ramirez-Marquez 2008; Chaturvedi and Padmavathy 2013)) to capture the topology changes during an interval t to Δt is an inadequate and misleading presumption.

The limitation of this work is that it assumes the presence of the *end-to-end* connectivity, thereby path(s) between (s, d) node pair via single or multiple hops; however, such an assumption does not always apply in general TVCNs. Under such situations, the existing methodologies of reliability evaluation will result in a zero reliability figure. Recall that Chapter 5 presented reliability evaluation technique for TVCNs which seldom have an end-to-end path and in which the transmission of the information among peer nodes takes place in a *store-carry-and-forward* fashion by the virtue of opportunistically created paths

in time. More specifically, Chapter 5 introduced the mechanism to evaluate different reliability measures of device-to-device connected TVCNs using all enumerated TS-MPS and tv-arborescences. Thus, the reliability evaluation approach presented in Chapter 7 is inferior compared to the approach in Chapter 5. However, the work has been included in this thesis for addressing the shortcomings of the existing techniques, and for the sake of completeness of discussions on both device-to-device and end-to-end connected TVCNs.

Chapter 8 Summary and Conclusions

The research presented in this thesis primarily focused on extending the concept of *MPS*, *MCS*, and *arborescences*, commonly used to evaluate the reliability of static networks to the present days' paradigm of TVCNs for their reliability assessment. We have formally defined each problem and proposed novel algorithms to enumerate all *MPS*, *MCS* and *arborescence* equivalent TS-MPS, TS-MCS and tv-arborescences, respectively, for TVCNs. Note that the proposed notions effectively enabled us to model device-to-device communication. Each of the presented or devised algorithms have been thoroughly evaluated for assessing their efficiency and accuracy. More specifically, the proposed solution(s) of each objective set out in this research has been provided with a necessary background, detailed formulation, analysis with detailed supporting discussions, and conclusions. The excerpt of each contribution to the research fraternity has been presented in this chapter.

1) We have presented an in-depth survey of the *state-of-the-art* techniques available for the performance analysis of TVCNs. Through our literature review, we inferred that modelling and evaluation of reliability related measures of TVCNs is critical for their better performance in harsh environmental conditions.
2) At the beginning, we ferreted out the scarcity of much understanding on TVCNs (from reliability perspective) due to the presence of timestamps over edges, which brought new characteristics of the network into picture. To the best of our knowledge, there was no work which enumerated all TS-MPS, TS-MCS and tv-arborescences vital for device-to-device communication modelling. Besides, no work dealt in reliability evaluation of device-to-device communication dependent TVCNs. We addressed these novel aims subsequently during the course of this research. The detailed conclusions are as follows:
    a) We made a maiden attempt to enumerate all TS-MPS of a TVCN by considering store-carry-and-forward mechanism of data transmission for modelling

connectivity between (s, d) node pair in device-to-device connected TVCNs. This problem enables more understanding about the notion of an MPS over time. For example, in contrast to static networks, in TVCNs the number of all TS-MPS between (s, d) node pair are not necessarily same as the number of all TS-MPS between (d, s) node pair. Besides, it is also important to appreciate the fact that a TS-MPS connects two nodes over time even in the case where the nodes are never able to connect themselves in any time slot. In the quest of better TS-MPS enumeration algorithms, we presented four *all* TS-MPS enumeration algorithms. The performance of the four proposed algorithms was evaluated against arbitrarily generated TVCN topologies. Each algorithm produced an exactly same set of TS-MPS, thereby certified their respective veracity. Further, the time complexity of each algorithm was also analyzed. Other useful network metrics, *viz*., all foremost, shortest and fastest TS-MPS were generated by using the enumerated TS-MPS. The network reliability, $R(s, d)$, between a pair of nodes was evaluated, for several simulated networks, by applying SDP-MVI algorithm on the enumerated set of TS-MPS. Thereafter, the 2-terminal reliability associated with all foremost, shortest and fastest TS-MPS was also evaluated. Besides, this work introduced and evaluated a new metric ESC to find the expected delay in the network and calculated a well-known network metric EHC for TVCNs. The aforementioned metrics may serve as one of the objective function in different routing strategies, *e.g.*, in energy-aware routing, shortest TS-MPS might be enumerated and utilized for passing the data from v_s to v_d.

b) We proposed two simple TS-MCS enumeration techniques for generating all TS-MCS between a specified (s, d) node pair of a TVCN. Note that the concept of TS-MCS can help in identification of weak links in TVCN designs. The very first approach which enumerates TS-MCS by inverting TS-MPS is a crude approach, while the second approach which utilizes Line Graph is a better alternative. The time complexity to generate a Line Graph was gauged to be $O(R^2)$, where R denotes the maximum numeric value assigned to the TSEs plus one. It was also shown that the presented algorithm generated each Line Graph that contained 6 to 20 nodes and

6 to 42 edges using the maximum of 0.022 seconds CPU time. However, the limitation of the Line Graph based algorithm is that it requires transformation of the TAG into Line Graph. To this end, we could not succeed in developing a procedure which can enumerate all TS-MCS from the TAG itself, *i.e.*, directly without any transformation.

The results obtained from experiments over ten arbitrary TAGs proved the existence of duality between TS-MPS and TS-MCS, and also evidently displayed that the number of TS-MCS is not necessarily less in comparison to the number of TS-MPS as it occurs most often in conventional static networks. Thus, due to the presence of duality between TS-MCS and TS-MPS, we can utilize either of them to compute different reliability metrics. However, it must be borne in mind that, with the proposed approaches, TS-MCS enumeration is a slow process compared to TS-MPS enumeration.

c) Riding on the concepts used in static networks, we proposed two methods to enumerate *all* tv-arborescences as the conventional idea of arborescence does not directly holds well in the case of TVCNs. The notion of tv-arborescences was effective to find all broadcasting routes in a TVCN. The first approach, *i.e.*, Method 1 initially generated all t-arborescences which were then processed to filter all tv-arborescences. The time complexity of Method 1 was assessed to be $O(|E||\mathrm{TA}_{All}|)$. Nevertheless, the first method was observed to be computationally expensive. Consequently, an efficient approach named as Method 2 was developed. Method 2 enumerated all tv-arborescences using all arborescences. The time complexity of Method 2 was analyzed as $O(|\mathrm{A}_{All}||V||TS|^{|V|})$. In a nutshell, both the methods extended the Tutte's Matrix Tree theorem – well-known in graph theory – to TVCNs. Further, we also showed the properties of both type of arborescences of TVCNs along with their significance. The simulation results corroborated all stated properties. Besides, in order to i) assess the performance of data broadcast in TVCNs operating, in general, in extreme environmental conditions, and ii) develop techniques for the reliability design of TVCNs, three network reliability metrics were introduced and evaluated, *viz*., $R(v_j)$, $R_1(K)$ and $R_2(K)$, by the application of

well-known SDP-MVI technique on the enumerated tv-arborescences. The outcome of this work also conjectured that the reliability metric $R_2(K)$ could be used as the upper bound of $R_1(K)$ and showed that evaluating $R_1(K)$ is often intractable. The performance analysis of the two proposed methods against ten arbitrarily generated TVCN topologies has also been provided to convince the planner that the proposed algorithms are also feasible for application over large TVCNs. However, it is worth to note that the two proposed methods pose a limitation in case the TVCNs form disconnected components which might never be connected during the period of the network. In such cases instead of tv-arborescences disconnected forests would form. Thus, the simulation results in this case are of much practical significance and can serve as an indicator for proper planning and design of TVCNs.

d) This thesis also addressed the topology control problem in TVCNs and considered network upgrade problem, which in general is an NP-hard problem. Both problems are well known for conventional static networks, while they are equally important and perhaps more challenging in case of TVCNs. More specifically, by utilizing the enumerated t- and tv-arborescences, we presented some initial insights into the problem of topology control and contact plan management in TVCNs. This rendered four new ratios which can assist a network designer to take apt decisions while choosing a contact plan among alternatives or for controlling its topology. The work also considered a *max-min* problem for TVCN upgrade and presented a greedy algorithm for upgrading a TVCN. Simulation results from ten arbitrarily generated TAGs revealed that the proposed algorithm is sub-optimal; however, it performs better than the average case of random TVCN upgrade. Hopefully, in future, more efficient procedures will appear in this area.

3) The literature survey indicated that the existing approaches have overlooked the conditional survival of the mobile nodes executing a mission and hence provided erroneous results and underestimated the value of reliability. Therefore, this thesis also evaluated the 2-terminal reliability, *i.e.*, R_{2TR_m}, of MANETs, which utilize end-to-end connections for data exchange, with the help of Monte Carlo Simulation to take up the

oversighted deficiency and utilized the concept of LET and BT to determine more accurately the incremental times at which topology of a MANET changes. We have shown through simulations the effectiveness of the proposed approach compared to the existing approaches. The R_{2TR_m} analysis of MANETs using the proposed approach would also assist the designers/decision makers to design/deploy an application-oriented MANET with an adequate and acceptable reliability.

It is important to note here that this problem has been considered for the sake of completeness, *i.e.*, to also cover the available techniques for end-to-end connected TVCNs and also to improve upon the shortcomings of the existing techniques. However, as this technique does not consider store-carry-and-forward mechanism of data exchange; thus, it poses a limitation for most practical scenarios.

8.1. Future Work

In future, the scholar envisage that the work contained in this thesis will pave way to find *broadcast reliability bounds* for TVCNs, as the evaluation of $R_1(K)$, in general, is intractable. Further, the work may be extended to design efficient, exact and/or greedy, algorithms for TVCN upgrade. Another possible extension of this thesis could be to utilize the proposed approaches of TVCN topology design to consider and/or include other network performance constraints, *e.g.*, delay, cost, flow and throughput. An additional significant problem can be the development of algorithm(s) to find global cuts for TVCNs and/or the evaluation of capacity related reliability of TVCNs.

References

Abolhasan M, Wysocki T, Dutkiewicz E (2004) A review of routing protocols for mobile ad hoc networks. Ad Hoc Netw 2:1–22.

Aggarwal K, Gupta J, Misra K (1975) A simple method for reliability evaluation of a communication system. IEEE Trans Commun 23:563–566.

Ahmad M, Mishra DK (2012) A Reliability Calculations Model for Large-Scale MANETs. Int J Comput Appl 59: 17–21.

Ahmad SH (1988) Simple enumeration of minimal cutsets of acyclic directed graph. IEEE Trans Reliab 37:484–487.

Aigner M (2007) A course in enumeration. Springer Science & Business Media.

Alotaibi E, Mukherjee B (2012) A survey on routing algorithms for wireless Ad-Hoc and mesh networks. Comput Netw 56:940–965.

Amor SB, Bui M, Lavallée I (2010) Optimizing Mobile Networks Connectivity and Routing Using Percolation Theory and Epidemic Algorithms. In: IICS. Citeseer, pp 63–78.

Aschenbruck N, Munjal A, Camp T (2011) Trace-based mobility modeling for multi-hop wireless networks. Comput Commun 34:704–714.

Bai F, Helmy A (2004) A survey of mobility models. Wirel Adhoc Netw Univ South Calif USA 206: p.147.

Batabyal S, Bhaumik P (2015) Mobility Models, Traces and Impact of Mobility on Opportunistic Routing Algorithms: A Survey. IEEE Commun Surv Tutor 17:1679–1707.

Baudic G, Perennou T, Lochin E (2016) Following the right path: Using traces for the study of DTNs. Comput Commun 88:25–33.

Bekmezci İ, Sahingoz OK, Temel Ş (2013) Flying Ad-Hoc Networks (FANETs): A survey. Ad Hoc Netw 11:1254–1270.

Bhadra S, Ferreira A (2003) Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In: Ad-Hoc, Mobile, and Wireless Networks. Springer, pp 259–270.

Bhadra S, Ferreira A (2002) Computing multicast trees in dynamic networks using evolving graphs. [Research Report] RR-4531, INRIA. 2002. <inria-00072057>.

Billinton R, Allan RN (1992) Reliability Evaluation of Engineering Systems Concepts and Techniques. Springer US : Imprint : Springer, Boston, MA.

Caldarelli G (2007) Scale-free networks: complex webs in nature and technology. Oxford University Press, Oxford.

Casteigts A, Flocchini P, Quattrociocchi W, Santoro N (2012) Time-varying graphs and dynamic networks. Int J Parallel Emergent Distrib Syst 27:387–408.

Casteigts A, Flocchini P, Quattrociocchi W, Santoro N (2011) Time-Varying Graphs and Dynamic Networks. In: Frey H, Li X, Ruehrup S (eds) Ad-hoc, Mobile, and Wireless Networks. Springer Berlin Heidelberg, pp 346–359.

Chakraborty M, Chowdhury S, Chakraborty J, et al (2019) Algorithms for generating all possible spanning trees of a simple undirected connected graph: an extensive review. Complex Intell Syst 5: 265-281.

Chaturvedi SK (2016) Network reliability: measures and evaluation. John Wiley & Sons ; Scrivener Publishing, Hoboken, New Jersey : Salem, Massachusetts.

Chaturvedi SK, Khanna G, Soh S (2018) Reliability evaluation of time evolving Delay Tolerant Networks based on Sum-of-Disjoint products. Reliab Eng Syst Saf 171:136–151.

Chaturvedi SK, Mishra R (2009) An Efficient Approach to Enumerate Cutsets Arising in Capacity Related Reliability Evaluation. Qual Technol Quant Manag 6:43–54.

Chaturvedi SK, Misra KB (2002) An Efficient Multi-Variable Inversion Algorithm for Reliability Evaluation of Complex Systems using Path Sets. Int J Reliab Qual Saf Eng 09:237–259.

Chaturvedi SK, Padmavathy N (2013) The Influence of Scenario Metrics on Network Reliability of Mobile Ad Hoc Network. Int J Perform Eng 9:61-74.

Chen H, Shi K (2015) Topology control for predictable delay-tolerant networks based on probability. Ad Hoc Netw 24:147–159.

Chen H, Shi K, Wu C (2016) Spanning tree based topology control for data collecting in predictable delay-tolerant networks. Ad Hoc Netw 46:48–60.

Chen W (2014) Explosive Percolation in Random Networks. Springer Berlin Heidelberg, Berlin, Heidelberg.

Clark J, Holton DA (2005) A first look at graph theory, Reprint. World Scientific, Singapore.

Coll-Perales B, Gozalvez J, Friderikos V (2016) Energy-efficient opportunistic forwarding in multi-hop cellular networks using device-to-device communications: B. Coll-Perales, J. Gozalvez and V. Friderikos. Trans Emerg Telecommun Technol 27:249–265.

Coll-Perales B, Gozalvez J, Lazaro O, Sepulcre M (2015) Opportunistic Multihopping for Energy Efficiency: Opportunistic Multihop Cellular Networking for Energy-Efficient Provision of Mobile Delay-Tolerant Services. IEEE Veh Technol Mag 10:93–101.

Cook JL, Ramirez-Marquez JE (2008) Mobility and reliability modeling for a mobile ad hoc network. IIE Trans 41:23–31.

Cook JL, Ramirez-Marquez JE (2007) Two-terminal reliability analyses for a mobile ad hoc wireless network. Reliab Eng Syst Saf 92:821–829.

Coscia M (2018) Using arborescences to estimate hierarchicalness in directed complex networks. PLOS ONE 13:e0190825.

Dai C-Q, Song Q, Guo L (2018) An Intelligent Computing Method for Contact Plan Design in the Multi-Layer Spatial Node-Based Internet of Things. Sensors 18:2852.

Díaz J, Mitsche D, Santi P (2011) Theoretical aspects of graph models for MANETs. In: Theoretical aspects of distributed computing in sensor networks. Springer, pp 161–190.

Du D, Hu X (2008) Steiner Tree Problems in Computer Communication Networks. World Scientific.

Egeland G, Engelstad P (2009) The availability and reliability of wireless multi-hop networks with stochastic link failures. IEEE J Sel Areas Commun 27:1132–1146.

Eiza MH, Ni Q (2013) An Evolving Graph-Based Reliable Routing Scheme for VANETs. IEEE Trans Veh Technol 62:1493–1504.

Ferreira A (2003) Building a Reference Combinatorial Model for Dynamic Networks: Initial Results in Evolving Graphs. RR-5041, INRIA. 2003. <inria-00071542>.

Ferreira A (2002) On models and algorithms for dynamic communication networks: The case for evolving graphs. In: In Proc. ALGOTEL, sn. p.7.

Fraire J, Finochietto JM (2015a) Routing-aware fair contact plan design for predictable delay tolerant networks. Ad Hoc Netw 25:303–313.

Fraire JA, Finochietto JM (2015b) Design challenges in contact plans for disruption-tolerant satellite networks. IEEE Commun Mag 53:163–169.

Fraire JA, Madoery P, Finochietto JM (2017a) Contact Plan Design for Predictable Disruption-tolerant Space Sensor Networks. In: Rashvand HF, Abedi A (eds) Wireless Sensor Systems for Extreme Environments. John Wiley & Sons, Ltd, Chichester, UK, pp 123–150.

Fraire JA, Madoery PG, Finochietto JM (2014) On the Design and Analysis of Fair Contact Plans in Predictable Delay-Tolerant Networks. IEEE Sens J 14:3874–3882.

Fraire JA, Madoery PG, Finochietto JM, Leguizamón G (2017b) An evolutionary approach towards contact plan design for disruption-tolerant satellite networks. Appl Soft Comput 52:446–456.

Gabow HN, Myers EW (1978) Finding All Spanning Trees of Directed and Undirected Graphs. SIAM J Comput 7:280–287.

George B, Shekhar S (2008) Time-aggregated graphs for modeling spatio-temporal networks. In: Journal on Data Semantics XI. Springer, pp 191–212.

Gunturi V, Shekhar S, Bhattacharya A (2010) Minimum spanning tree on spatio-temporal networks. In: Database and Expert Systems Applications. Springer, pp 149–158.

Hariharan R, Kapoor S, Kumar V (1995) Faster enumeration of all spanning trees of a directed graph. In: Akl SG, Dehne F, Sack J-R, Santoro N (eds) Algorithms and Data Structures. Springer Berlin Heidelberg, pp 428–439.

Hekmat R (2006) Ad-hoc networks: fundamental properties and network topologies. Springer Science & Business Media.

Holme P (2015) Modern temporal network theory: a colloquium. Eur Phys J B 88:1–30.

Huang M, Chen S, Zhu Y, Wang Y (2013) Topology Control for Time-Evolving and Predictable Delay-Tolerant Networks. IEEE Trans Comput 62:2308–2321.

Huang S, Cheng J, Wu H (2014) Temporal graph traversals: Definitions, algorithms, and applications. ArXiv Prepr ArXiv14011919.

Huang S, Fu AW-C, Liu R (2015) Minimum Spanning Trees in Temporal Graphs. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM Press, Melbourne, Victoria, Australia, pp 419–430.

Ivanic N, Rivera B, Adamson B (2009) Mobile Ad Hoc Network emulation environment. In: MILCOM 2009, IEEE, pp 1–6.

Juang P, Oki H, Wang Y, et al (2002) Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. ACM SIGARCH Comput Archit News 30:96.

Kamiyama N, Kawase Y (2015) On packing arborescences in temporal networks. Inf Process Lett 115:321–325.

Kawamoto Y, Nishiyama H, Kato N (2013) Toward terminal-to-terminal communication networks: A hybrid MANET and DTN approach. In: Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on. IEEE, pp 228–232.

Kececioglu D (2002) Reliability and life testing handbook. DEStech Publications, Lancaster, PA.

Khanna G, Chaturvedi SK (2018) A Comprehensive Survey on Multi-hop Wireless Networks: Milestones, Changing Trends and Concomitant Challenges. Wirel Pers Commun 101:677–722.

Khanna G, Chaturvedi SK, Soh S (2020a) Time Varying Communication Networks: Modelling, Reliability Evaluation and Optimization. In: Ram M, Pham H (eds) Advances in Reliability Analysis and its Applications. Springer International Publishing, Cham, pp 1–30.

Khanna G, Chaturvedi SK, Soh S (2020b) Two-Terminal Reliability Analysis for Time-Evolving and Predictable Delay-Tolerant Networks. Recent Adv Electr Electron Eng Former Recent Pat Electr Electron Eng 13:236–250.

Khanna G, Chaturvedi SK, Soh S (2019a) Reliability evaluation of mobile ad hoc networks by considering link expiration time and border time. Int J Syst Assur Eng Manag 10:399–415.

Khanna G, Chaturvedi SK, Soh S (2019b) On computing the reliability of opportunistic multihop networks with Mobile relays. Qual Reliab Eng Int 35:870–888.

Khanna G, Soh S, Chaturvedi SK, Chin K-W (2020c) On Enumeration of Spanning Arborescences and Reliability for Network Broadcast in Fixed-Schedule Dynamic Networks. IEEE Trans Netw Sci Eng 7:2980–2996.

Kharbash S, Wang W (2007) Computing two-terminal reliability in mobile ad hoc networks. In: Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE. IEEE, pp 2831–2836.

Kiess W, Mauve M (2007) A survey on real-world implementations of mobile ad-hoc networks. Ad Hoc Netw 5:324–339.

Kropff M, Krop T, Hollick M, et al (2006) A Survey on RealWorld and Emulation Testbeds for Mobile Ad hoc Networks. IEEE, pp 448–453.

Krumke SO, Marathe MV, Noltemeier H, et al (1998) Approximation Algorithms for Certain Network Improvement Problems. J Comb Optim 2:257–288.

Leiserson CE, Rivest RL, Cormen TH, Stein C (2001) Introduction to algorithms. MIT press Cambridge, MA.

Li D, Zhang Q, Zio E, et al (2015a) Network reliability analysis based on percolation theory. Reliab Eng Syst Saf 142:556–562.

Li F, Chen S, Huang M, et al (2015b) Reliable Topology Design in Time-Evolving Delay-Tolerant Networks with Unreliable Links. IEEE Trans Mob Comput 14:1301–1314.

Liang Q, Modiano E (2017) Survivability in Time-Varying Networks. IEEE Trans Mob Comput 16:2668–2681.

Lim KY, Soh S, Rai S (2005) Computer communication network upgrade for optimal capacity related reliability. In: 2005 Asia-Pacific Conference on Communications. IEEE, pp 1102–1106.

Lucet C, Manouvrier J-F (1999) Exact methods to compute network reliability. In: Statistical and Probabilistic Models in Reliability. Springer, pp 279–294.

Maini AK, Agrawal V (2014) Satellite technology: principles and applications, Third edition. Wiley, Chichester, West Sussex.

Masbaum G, Vaintrob A (2002) A new matrix-tree theorem. Int Math Res Not 2002:1397–1426.

Matis M, Doboš L, Papaj J (2016) An Enhanced Hybrid Social Based Routing Algorithm for MANET-DTN. Mob Inf Syst 2016:1–12.

Meena KS, Vasanthi T (2016a) Reliability Design for a MANET with Cluster-Head Gateway Routing Protocol. Commun Stat - Theory Methods 45:3904–3918.

Meena KS, Vasanthi T (2016b) Reliability Analysis of Mobile Ad Hoc Networks Using Universal Generating Function: Reliability Analysis of MANET using UGF. Qual Reliab Eng Int 32:111–122.

Mertzios GB, Michail O, Spirakis PG (2019) Temporal Network Optimization Subject to Connectivity Constraints. Algorithmica 81:1416–1449.

Migov DA, Shakhov V (2014) Reliability of Ad Hoc Networks with Imperfect Nodes. Springer International Publishing, pp 49–58.

Mihalák M, Uznański P, Yordanov P (2016) Prime factorization of the Kirchhoff polynomial: Compact enumeration of arborescences. In: 2016 Proceedings of the Thirteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO). SIAM, pp 93–105.

Miklós I (2018) Computational complexity of counting and sampling. CRC Press/Taylor & Francis Group, Boca Raton.

Mishra R, Chaturvedi SK (2009) A Cutsets-Based Unified Framework to Evaluate Network Reliability Measures. IEEE Trans Reliab 58:658–666.

Misra KB (1992) Reliability analysis and prediction: a methodology oriented treatment. Elsevier, Amsterdam ; New York.

Misra KB (1993) New Trends in System Reliability Evaluation. Elsevier Science Ltd.

Misra S, Rajesh G (2011) Bird Flight-Inspired Routing Protocol for Mobile Ad Hoc Networks. ACM Trans Auton Adapt Syst 6:1–37.

Munjal A, Camp T, Aschenbruck N (2012) Changing Trends in Modeling Mobility. J Electr Comput Eng 2012:1–16.

Newman M, Barabasi A-L, Watts DJ (2006) The structure and dynamics of networks. Princeton University Press.

Padmavathy N, Chaturvedi SK (2013) Evaluation of mobile ad hoc network reliability using propagation-based link reliability model. Reliab Eng Syst Saf 115:1–9.

Padmavathy N, Chaturvedi SK (2015) Reliability evaluation of capacitated mobile ad hoc network using log-normal shadowing propagation model. Int J Reliab Saf 9:70–89.

Paik D, Sahni S (1995) Network upgrading problems. Networks 26:45–58.

Panda DK, Dash RK (2017) Reliability Evaluation and Analysis of Mobile Ad Hoc Networks. Int J Electr Comput Eng IJECE 7:479-485.

Patel KN, Jhaveri R h. (2015) A Survey on Emulation Testbeds for Mobile Ad-hoc Networks. Procedia Comput Sci 45:581–591.

Peiravi A, Kheibari HT (2008) Fast estimation of network reliability using modified Manhattan distance in mobile wireless networks. J Appl Sci 8:4303–4311.

Pentland A, Fletcher R, Hasson A (2004) DakNet: rethinking connectivity in developing nations. Computer 37:78–83.

Raffelsberger C, Hellwagner H (2014) Combined Mobile Ad-Hoc and Delay/Disruption-Tolerant Routing. In: Guo S, Lloret J, Manzoni P, Ruehrup S (eds) Ad-hoc, Mobile, and Wireless Networks. Springer International Publishing, Cham, pp 1–14.

Rai S, Kumar A, Prasad EV (1986) Computing terminal reliability of computer network. Reliab Eng 16:109–119.

Rai S, Veeraraghavan M, Trivedi KS (1995) A survey of efficient reliability computation using disjoint products approach. Networks 25:147–163.

Rebaiaia M-L, Ait-Kadi D (2015) Reliability Evaluation of Imperfect K-Terminal Stochastic Networks using Polygon-to Chain and Series-parallel Reductions. In: Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks. ACM, pp 115–122.

Roy RR (2011) Handbook of Mobile Ad Hoc Networks for Mobility Models. Springer US, Boston, MA.

Santoro N (2016) Time to Change: On Distributed Computing in Dynamic Networks (Keynote). In: Anceaume E, Cachin C, Potop-Butucaru M (eds) 19th International Conference on Principles of Distributed Systems. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp 1–14.

Schwager SJ (1984) Bonferroni Sometimes Loses. Am Stat 38:192–197.

Shen C-C, Huang Z, Jaikaeo C (2006) Directional broadcast for mobile ad hoc networks with percolation theory. Mob Comput IEEE Trans On 5:317–332.

Silva AP, Hilario MR, Hirata CM, Obraczka K (2015) A Percolation-Based Approach to Model DTN Congestion Control. IEEE, pp 100–108.

Singh MM, Baruah M, Mandal JK (2014) Reliability computation of mobile Ad-Hoc network using logistic regression. In: 2014 Eleventh International Conference on Wireless and Optical Communications Networks (WOCN). IEEE, pp 1–5.

Soh S, Lau W, Rai S, Brooks RR (2007) On computing reliability and expected hop count of wireless communication networks. Int J Perform Eng 3:267–279.

Soh S, Rai S (2005) An Efficient Cutset Approach for Evaluating Communication-Network Reliability With Heterogeneous Link-Capacities. IEEE Trans Reliab 54:133–144.

Soh S, Rai S (1993) Experimental results on preprocessing of path/cut terms in sum of disjoint products technique. IEEE Trans Reliab 42:24–33.

Soh S, Rai S, Brooks RR (2008) Performability Issues in Wireless Communication Networks. In: Handbook of Performability Engineering. Springer, pp 1047–1067.

Su W, Lee S-J, Gerla M (2001) Mobility prediction and routing in ad hoc wireless networks. Int J Netw Manag 11:3–30.

Vincenty T (1975) Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Surv Rev 23:88–93.

Wang Y, Sheng M, Li J, et al (2016) Dynamic Contact Plan Design in Broadband Satellite Networks With Varying Contact Capacity. IEEE Commun Lett 20:2410–2413.

Waxman BM (1988) Routing of multipoint connections. IEEE J Sel Areas Commun 6:1617–1622.

White T (2002) Evolving Optimally Reliable Networks by Adding an Edge. In: PDPTA. pp 1712–1717.

Xing L, Amari SV (2015) Binary decision diagrams and extensions for system reliability analysis. Scrivener Publishing/Wiley, Hoboken, New Jersey.

Xuan BB, Ferreira A, Jarry A (2003a) Evolving graphs and least cost journeys in dynamic networks. In: WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks. Sophia Antipolis, France, p 10.

Xuan BB, Ferreira A, Jarry A (2003b) Computing shortest, fastest, and foremost journeys in dynamic networks. Int J Found Comput Sci 14:267–285.

Y. -C. Hsieh (2003) New Reliability Bounds for Coherent Systems. J Oper Res Soc 54:995–1001.

Zaidi ZR, Mark BL (2011) Mobility Tracking Based on Autoregressive Models. IEEE Trans Mob Comput 10:32–43.

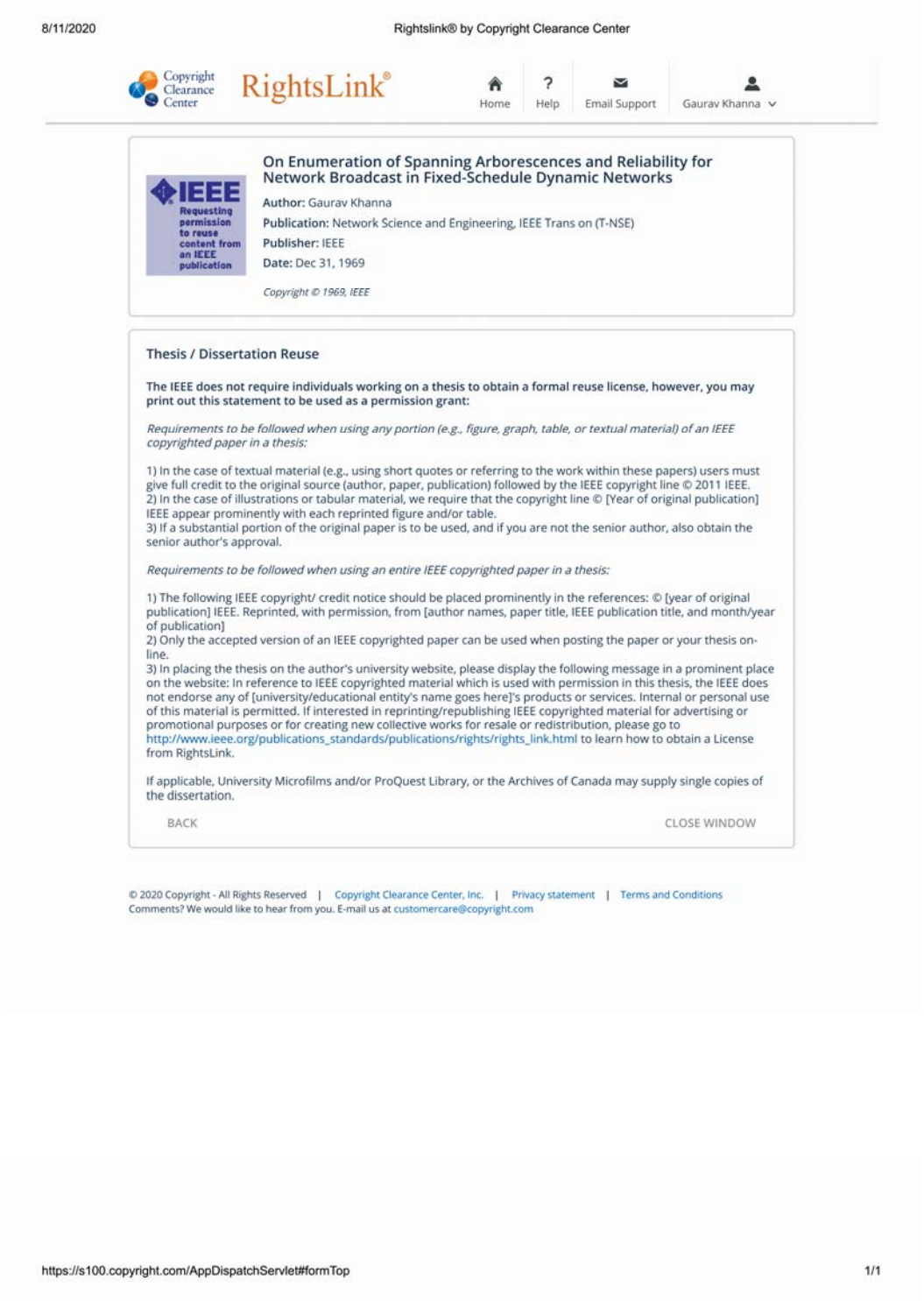
Zang X, Sun H, Trivedi KS (2000) A BDD-based algorithm for reliability graph analysis. Department of Electrical Engineering, Duke University, Tech Rep.

Zhang W, Ma H, Wu T, et al (2019) Efficient topology control for time-varying spacecraft networks with unreliable links. Int J Distrib Sens Netw 15:155014771987937.

Zhang Z (2006) Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. IEEE Commun Surv Tutor 8:24–37.

Zhou D, Sheng M, Wang X, et al (2017) Mission Aware Contact Plan Design in Resource-Limited Small Satellite Networks. IEEE Trans Commun 65:2451–2466.

Rights and Permissions



8/11/2020 Rightslink® by Copyright Clearance Center

Copyright Clearance Center RightsLink® Home Help Email Support Gaurav Khanna

IEEE Requesting permission to reuse content from an IEEE publication

On Enumeration of Spanning Arborescences and Reliability for Network Broadcast in Fixed-Schedule Dynamic Networks

Author: Gaurav Khanna
Publication: Network Science and Engineering, IEEE Trans on (T-NSE)
Publisher: IEEE
Date: Dec 31, 1969

Copyright © 1969, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

https://s100.copyright.com/AppDispatchServlet#formTop 1/1

Gmail gaurav khanna <gauravkhanna88@gmail.com>

Fwd: Rights and Permissions: Reference No: RAEENG-19-27422

Ambreen Irshad <ambreenirshad@benthamscience.net> Sun, Mar 15, 2020 at 8:04 PM
To: gauravkhanna88@gmail.com

Grant of Permission

Dear Dr. Khanna :

Thank you for your interest in our copyrighted material, and for requesting permission for its use.

Permission is granted for the following subject to the conditions outlined below:

"Two-Terminal Reliability Analysis for Time-Evolving and Predictable Delay-Tolerant Networks" by ***Gaurav Khanna***, *S. K. Chaturvedi and Sieteng Soh* in the journal **Recent Advances in Electrical & Electronic Engineering.**

To be used in the following manner:

1. Bentham Science Publishers grants you the right to reproduce the material indicated above on a one-time, non-exclusive basis, solely for the purpose described. Permission must be requested separately for any future or additional use.

2. For an article, the copyright notice must be printed on the first page of article or book chapter. For figures, photographs, covers, or tables, the notice may appear with the material, in a footnote, or in the reference list.

Thank you for your patience while your request was being processed. If you wish to contact us further, please use the address below.

Sincerely,

AMBREEN IRSHAD
***Permissions & Rights* Manager**
Bentham Science Publishers
Email: ambreenirshad@benthamscience.net

On Fri, Mar 13, 2020 at 10:41 AM gaurav khanna <gauravkhanna88@gmail.com> wrote:

Dear Ms. Humaira,

I seek permission to reuse below given article in my Ph. D. Thesis. Note that I am one of the author of the concerned paper.

"Two-Terminal Reliability Analysis for Time-Evolving and Predictable Delay-Tolerant Networks" by ***Gaurav Khanna***, *S. K. Chaturvedi and Sieteng Soh* in the journal **Recent Advances in Electrical & Electronic Engineering.**

I tried to use **Rights and Permissions** option on journal portal (Link: http://www.eurekaselect.com/node/170026/article/two-terminal-reliability-analysis-for-time-evolving-and-predictable-delay-tolerant-networks) but could not find our paper! In this regard, I request you to do the needful at the earliest. Further, for your information, the expected month of my thesis submission is October 2020.

Thank you in advance.

With Warm Regards,
Gaurav khanna
Research Scholar
SCSQ&R, IIT Kharagpur and SEEC&MS, Curtin University.
Mob: +91-8909441497, +91-9910278985

On computing the reliability of opportunistic multihop networks with Mobile relays

Author: Sieteng Soh, S.K. Chaturvedi, Gaurav Khanna
Publication: Quality & Reliability Engineering International
Publisher: John Wiley and Sons
Date: Jan 8, 2019

Order Completed

Thank you for your order.

This Agreement between Gaurav Khanna ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

Printable Details

License Number	4726231258777
License date	Dec 11, 2019

Licensed Content

Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	Quality & Reliability Engineering International
Licensed Content Title	On computing the reliability of opportunistic multihop networks with Mobile relays
Licensed Content Author	Sieteng Soh, S.K. Chaturvedi, Gaurav Khanna
Licensed Content Date	Jan 8, 2019
Licensed Content Volume	35
Licensed Content Issue	4
Licensed Content Pages	19

Order Details

Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley article
Format	Print and electronic
Portion	Full article
Will you be translating?	No

About Your Work

Title of your thesis / dissertation	Modeling and Reliability Evaluation of Time Varying Communication Networks
Expected completion date	Jan 2021
Expected size (number of pages)	1

Additional Data

Order reference number	15RE91R04

Requestor Location

Requestor Location	Gaurav Khanna SCSQR IIT Kharagpur West Mednipur Kharagpur, Kharagpur 721302 India Attn:

Tax Details

Publisher Tax ID	EU826007151

Price

Total	0.00 USD

Would you like to purchase the full text of this article? If so, please continue on to the content ordering system located here: Purchase PDF
If you click on the buttons below or close this window, you will not be able to return to the content ordering system.

Total: 0.00 USD

CLOSE WINDOW ORDER MORE

Home | Help | Live Chat | Gaurav Khanna

Reliability evaluation of mobile ad hoc networks by considering link expiration time and border time

Author: Gaurav Khanna, S. K. Chaturvedi, Sieteng Soh

Publication: International Journal of Systems Assurance Engineering and Management

Publisher: Springer Nature

Date: Jan 1, 2019

Order Completed

Thank you for your order.

This Agreement between Gaurav Khanna ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

License Number	4726230964899
License date	Dec 11, 2019

Printable Details

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	International Journal of Systems Assurance Engineering and Management
Licensed Content Title	Reliability evaluation of mobile ad hoc networks by considering link expiration time and border time
Licensed Content Author	Gaurav Khanna, S. K. Chaturvedi, Sieteng Soh
Licensed Content Date	Jan 1, 2019
Licensed Content Volume	10
Licensed Content Issue	3

Order Details

Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	30 - 99
Author of this Springer Nature content	yes

About Your Work

Title	Modeling and Reliability Evaluation of Time Varying Communication Networks
Institution name	Indian Institute of Technology Kharagpur, India and Curtin University, Australia
Expected presentation date	Jan 2021

Additional Data

Order reference number	15RE91R04

Requestor Location

Requestor Location	Gaurav Khanna SCSQR IIT Kharagpur West Mednipur Kharagpur, Kharagpur 721302 India Attn:

Tax Details

$ Price

Total	0.00 USD

Total: 0.00 USD

CLOSE WINDOW ORDER MORE

Copyright Clearance Center RightsLink®

Home | Help | Live Chat | Gaurav Khanna

SPRINGER NATURE

A Comprehensive Survey on Multi-hop Wireless Networks: Milestones, Changing Trends and Concomitant Challenges

Author: Gaurav Khanna, S. K. Chaturvedi
Publication: Wireless Personal Communications
Publisher: Springer Nature
Date: Jan 1, 2018

Order Completed

Thank you for your order.

This Agreement between Gaurav Khanna ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

License Number	4726230461302
License date	Dec 11, 2019

Printable Details

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Wireless Personal Communications
Licensed Content Title	A Comprehensive Survey on Multi-hop Wireless Networks: Milestones, Changing Trends and Concomitant Challenges
Licensed Content Author	Gaurav Khanna, S. K. Chaturvedi
Licensed Content Date	Jan 1, 2018
Licensed Content Volume	101
Licensed Content Issue	2

Order Details

Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	30 - 99
Author of this Springer Nature content	yes

About Your Work

Title	Modeling and Reliability Evaluation of Time Varying Communication Networks
Institution name	Indian Institute of Technology Kharagpur, India and Curtin University, Australia
Expected presentation date	Jan 2021

Additional Data

Order reference number	15RE91R04

Requestor Location

Requestor Location	Gaurav Khanna SCSQR IIT Kharagpur West Mednipur Kharagpur, Kharagpur 721302 India Attn:

Tax Details

$ Price

Total	0.00 USD

Total: 0.00 USD

CLOSE WINDOW ORDER MORE

Copyright Clearance Center RightsLink®

Home | Help | Email Support | Gaurav Khanna

SPRINGER NATURE

Time Varying Communication Networks: Modelling, Reliability Evaluation and Optimization

Author: Gaurav Khanna, S. K. Chaturvedi, Sieteng Soh
Publication: Springer eBook
Publisher: Springer Nature
Date: Jan 1, 2020

Order Completed

Thank you for your order.

This Agreement between Gaurav Khanna ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

License Number	4786870179393
License date	Mar 13, 2020

Printable Details

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Time Varying Communication Networks: Modelling, Reliability Evaluation and Optimization
Licensed Content Author	Gaurav Khanna, S. K. Chaturvedi, Sieteng Soh
Licensed Content Date	Jan 1, 2020

Order Details

Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	30 - 99
Author of this Springer Nature content	yes

About Your Work

Title	Modeling and Reliability Evaluation of Time Varying Communication Networks
Institution name	Indian Institute of Technology Kharagpur, India and Curtin University, Australia
Expected presentation date	Jan 2021

Additional Data

Order reference number	15RE91R04

Requestor Location

Requestor Location	Gaurav Khanna SCSQR IIT Kharagpur West Mednipur Kharagpur, Kharagpur 721302 India Attn:

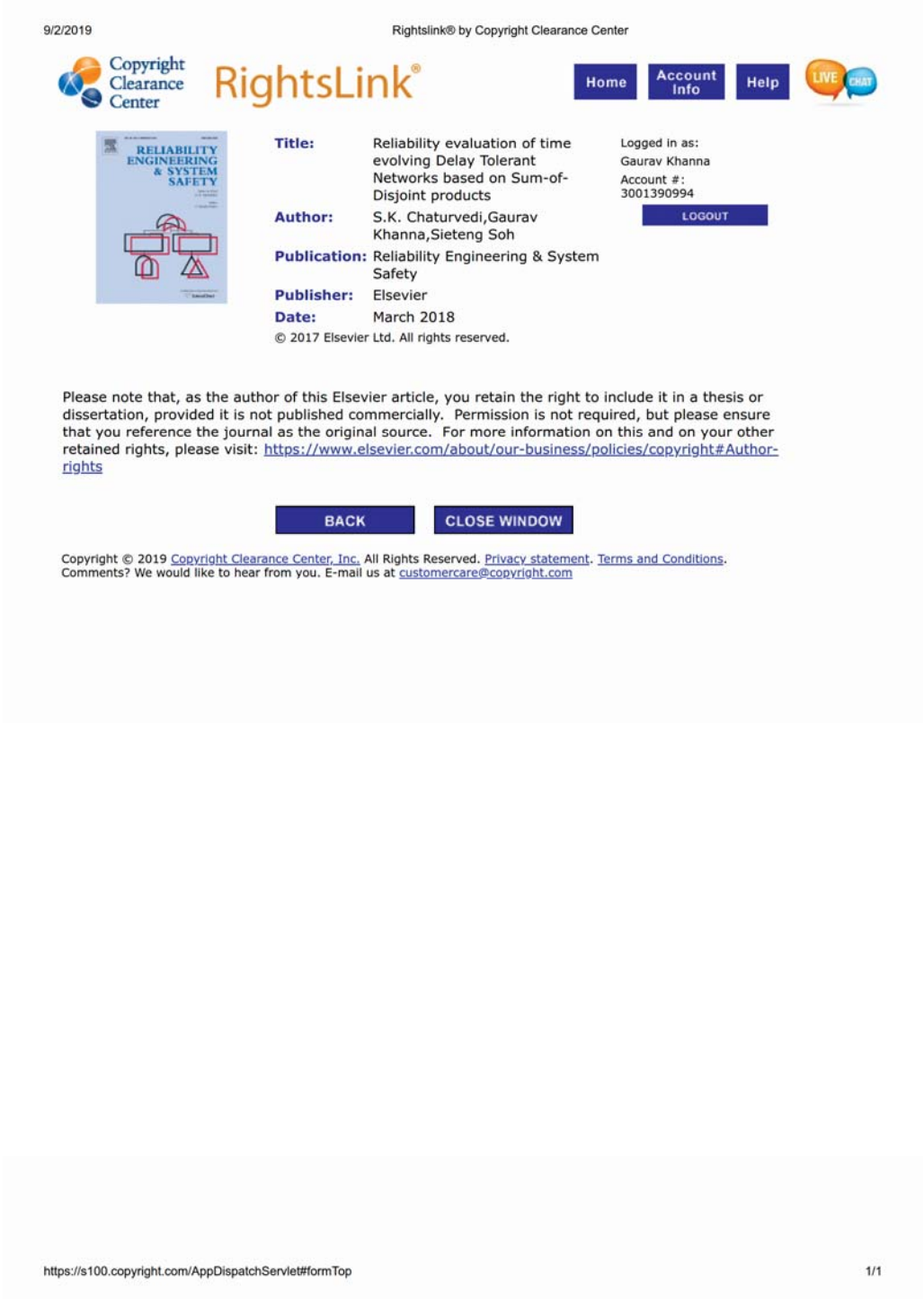
Tax Details

$ Price

Total	0.00 USD

Total: 0.00 USD

CLOSE WINDOW ORDER MORE



Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.

Curriculum Vitae

Gaurav Khanna was born in Lucknow – capital city of the Indian state of Uttar Pradesh. He completed his B. Tech. Degree in Electronics and Communication Engineering from Gautam Buddh Technical University, Lucknow, India, in the year 2011. Thereafter, he obtained his M. Tech. (Information and Communication Technology) with specialization in Wireless Communication and Networks from the School of Information and Communication Technology, Gautam Buddha University, Gautam Budh Nagar, India in the year 2015. Soon after his Masters, he joined Subir Chowdhury School of Quality and Reliability (erstwhile Reliability Engineering Center), Indian Institute of Technology Kharagpur as a Doctoral Scholar. Later, in 2018, he concurrently enrolled with the School of Electrical Engineering, Computing and Mathematical Sciences at Curtin University, Australia, under a Dual Doctoral Degree programme between Indian Institute of Technology Kharagpur and Curtin University, Australia. His current research interests include Mobile Ad Hoc Networks, Delay Tolerant Networks, Multistage Interconnection Networks and Reliability Engineering. During the course of his doctoral studies at Indian Institute of Technology Kharagpur and Curtin University, Australia, he has published several research papers, in refereed international journals, and book chapters.