Influence of Path Correlation on the Complexity of QoS Routing

Almerima Jamaković
Preface

Telecommunication is an integral part of modern society: mobile telephony, Internet, data and computer communication, e-mail … Within a reasonably short period of time a wide range of new facilities become available: it is easy to exchange information, communication is not limited to a fixed location and time anymore and distribution of services has become self-evident. On the other hand this also means that network traffic is highly diverse and each traffic type has unique requirements in terms of bandwidth, delay, loss, and availability. With the explosive growth of Internet, most network traffic today is based on the Internet Protocol (IP); IP packets do not take a specific path as they pass through the network. Having a single IP-based end-to-end transport protocol is beneficial because networking equipment becomes less complex to maintain, resulting in lower operational costs. This benefit, however, is countered by the fact that IP is a connectionless protocol. A consequence of this property is an unpredictable Quality of Service in a best-effort network.

This thesis, which leads towards the Master of Science (M.Sc.) degree, has been conducted at the Network Architectures and Services Group of the Telecommunications Department at the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology. The central theme of the M.Sc. program in Telecommunication offered at the Delft University of Technology is collection, extraction, transport and distribution of information. Basically, the program covers all principles of telecommunication: information transport and observation technology, system design as well as service-oriented applications.

This M.Sc. thesis targets the problems surrounding Quality of Service (QoS) routing with constraints on multiple QoS routing metrics, also referred to as the Multi-Constrained Path (MCP) problem. It mainly focuses on the correlation between the links in a network or briefly the path correlation. Moreover, the influence that specific path correlation structures have on the complexity of QoS routing will be in detail examined.

Acknowledgement

I would like to acknowledge the help of two individuals in preparation and completion of the thesis. Primarily, I would like to thank Prof. dr. ir. Piet Van Mieghem. He has done the pioneering work in this new research area and therefore provided the necessary basis for the thesis. Especially, I am grateful to ir. F. Kuipers for his instructions, comments and suggestions during this work.
Abstract

One of the basic problems in Quality of Service (QoS) routing is how to find a path in a network with constraints on multiple QoS metrics or in particular, how to solve the Multi-Constrained Path (MCP) problem. For multiple additive QoS metrics the path finding problem has been proven to be NP-complete. This discovery has dramatically influenced the research community, resulting in the common belief that exact QoS routing in practice is intractable. In fact, the MCP problem is not strong NP-complete, meaning that in practice an exact QoS algorithm may work in polynomial-time, making guaranteed Multi-Constrained QoS routing possible.

The goal of this M.Sc. thesis is to analyse and evaluate the influence of path correlation on the complexity of QoS routing or in detail, to analyse and evaluate NP-complete behaviour of the MCP problem. The proof of NP-completeness of the MCP problem indicates that specific correlation structures, between QoS metrics, have influence on the worst-case complexity of the MCP problem. Therefore, the complexity of QoS routing is analysed by systematically studying specific path correlation structures or inter-link correlation structures. If required, link correlation or intra-link correlation is added as well to terminate in the search for NP-complete behaviour. Evaluation of the complexity of Multi-Constrained QoS routing is done through simulations via an exact QoS routing algorithm, referred to as SAMCRA.

Keywords:
- Quality of Service (QoS) routing
- Multi-Constrained Path (MCP) problem
- NP-completeness
- Link Correlation or Intra-Link Correlation. (Correlation between Link Metrics)
- Path Correlation or Inter-Link Correlation. (Correlation between Path Metrics)
- SAMCRA, Self Adapting Multiple Constraints Routing Algorithm
# Table of Contents

Preface ............................................................................................................................ I
Acknowledgement ............................................................................................................. I
Abstract ............................................................................................................................. III
Table of Contents ............................................................................................................... V
List of Figures ..................................................................................................................... IX

1. Introduction: Motivation, Goal and Thesis Layout ....................................................... 1

2. Multi-Constrained QoS Routing .................................................................................. 3
   2.1. Introduction to Multi-Constrained QoS Routing .................................................... 3
   2.2. Introduction to Complexity Theory [4] .................................................................. 3
   2.3. NP-complete Problems [4] .................................................................................... 4
   2.4. Multi-Constrained QoS Metrics and Constraints [20,9] ....................................... 4
   2.5. Multi-Constrained Path Problem [20,9] .............................................................. 5
   2.6. Multi-Constrained Path Problem is NP-complete [9,21] ...................................... 6
   2.8. SAMCRA Algorithm [20] .................................................................................... 9
      2.8.1. Four Concepts of SAMCRA .......................................................................... 9
      2.8.2. Meta-Code of SAMCRA .............................................................................. 10
      2.8.3. Complexity of SAMCRA ............................................................................ 11
   2.9. Correlation Techniques [14] ............................................................................... 12
   2.10. Network Topology ............................................................................................. 13

3. Analysing Complexity of Multi-Constrained QoS Routing ......................................... 15
   3.3. Non-Dominance Property in Two-Dimensional Lattice Network Topology ........ 16

4. Analysing Covariance and Correlation Matrices ......................................................... 23
# Table of Contents

4.2. Theorems Involving Covariance and Correlation Matrices [12,13,14,15] ........... 25  
4.3. Positive Definiteness Limits Covariance and Correlation Matrices ................. 28  
4.5. Solving Positive Definiteness with Eigenvalues and Eigenvectors .................. 31  
5. Generating Correlated Vector Random Variables .............................................. 33  
  5.1. Jointly Gaussian Random Variables [13].................................................. 33  
6. Configuring Network Topology for Simulations .............................................. 37  
  6.1. Proposed Path Correlation Classes ......................................................... 37  
  6.1.1 Generating Path Correlation Classes via Correlation Matrix ...................... 37  
  6.1.2 Generating Path Correlation Classes via Underlying Chain Topologies ...... 41  
  6.2. Simulation Parameters .............................................................................. 42  
  6.3. Configuring Two-Dimensional Lattice Network Topology ............................. 43  
  6.3.1. Configuring Link Weights Consecutively ............................................ 43  
  6.3.2. Configuring Link Weights Randomly ................................................... 44  
7. Simulation Results on Complexity of Multi-Constrained QoS Routing ............... 47  
  7.1. Simulation Results for the Consecutive-Configuration of Link Weights ........ 47  
  7.2. Simulation Results for Random-Configuration of the Link Weights ............. 54  
8. Conclusions and Recommendations ............................................................... 79  
Appendix A ............................................................................................................ 81  
  Orthogonality [6] .............................................................................................. 81  
Appendix B ............................................................................................................ 83  
  Positive Definite Matrices [3] ............................................................................ 83  
Appendix C ............................................................................................................ 85  
  The Cholesky Decomposition [3] ....................................................................... 85  
Appendix D ............................................................................................................ 87  
  C-code for Cholesky Decomposition Algorithm [23] ....................................... 87  

- VI -
# Table of Contents

Appendix E ............................................................................................................................ 89

MATLAB M-code for Nearest Correlation Matrix Problem [7,16] ........................................... 89

Appendix F ............................................................................................................................. 91

C-code for Nearest Correlation Matrix Problem .................................................................... 91

Appendix G ............................................................................................................................. 95

Part of Correlation.C Program where Path Correlation Classes are generated ................. 95

Path Correlation of the Class 1 ................................................................................................ 95

Path Correlation of the Class 2 .............................................................................................. 96

Path Correlation of the Class 3 .............................................................................................. 97

Path Correlation of the Class 4 .............................................................................................. 98

Path Correlation of the Class 5 .............................................................................................. 99

Path Correlation of the Class 6 ............................................................................................ 100

Path Correlation of the Class 7 ............................................................................................ 104

Appendix H ............................................................................................................................. 107

Correlation.C Program ......................................................................................................... 107

References ............................................................................................................................. 111
List of Figures

Figure 2.6-1 The assignment of link weights to the links in the chain topology between nodes $i$ and $i+1$. ................................................................. 7

Figure 2.8.2-1 Meta-code of SAMCRA. ............................................................... 11

Figure 2.10-1 Example of a two-dimensional lattice network topology with 16 nodes and 24 links. .......................................................... 14

Figure 2.10-2 Example of a two-dimensional lattice network topology with two link weights on each link creating path correlation vector $Z[1, E]$ and link correlation vector $X[1, E]$. ................................................................. 14

Figure 3.2-1 The assignment of link weights to the links in the chain topology between nodes $i$ and $i+1$. ................................................................. 16

Figure 3.3-1 A two-dimensional lattice network topology with the main underlying chain topology, illustrated in bold lines. ........................................ 17

Figure 3.3-2 A two-dimensional lattice network topology with other underlying chain topology, illustrated in bold lines. ........................................ 17

Figure 3.3-3 A two-dimensional lattice network topology with links, illustrated in bold lines, which have to satisfy the non-dominance property if for the main underlying chain topology the Property 3.2-1 is obeyed. ................................. 17

Figure 3.3-4 Example of a two-dimensional lattice network topology with two link weights on each link creating vectors $Z[1, E]$ and $X[1, E]$. This example has four different chain topologies. Here, only the main chain topology is illustrated in bold lines. ................................................................. 18

Figure 3.3-5 Example of a two-dimensional lattice network topology with two link weights on each link creating vectors $Z[1, E]$ and $X[1, E]$. The links, which have to satisfy the non-dominance property if the Property 3.2-1 for the main underlying chain topology is obeyed, are illustrated in bold lines. ............... 21

Figure 4.5-1 Meta-code of Alternative Projection Programming Algorithm. ......................... 30

Figure 6.1.2-1 Example of path correlation of the class 7 in a two-dimensional lattice network topology, where the paths in underlying main chain topology are non-dominated. In the same way, in other underlying chain topologies, the paths are non-dominated, creating correlated links. ........................................ 41

Figure 6.3.1-1 Consecutive configuration of link and path vectors. .................................. 43

Figure 6.3.2-1 Random configuration of link and path vectors. ....................................... 45

Figure 7.1-1 Expected $k_{\min}$ queue-size for the Class 1 with consecutively configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 47

Figure 7.1-2 Maximum in $k_{\min}$ queue-size for the Class 1 with consecutively configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 48

Figure 7.1-3 Variance in $k_{\min}$ queue-size for the Class 1 with consecutively configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 49
List of Figures

Figure 7.1-4 Expected kmin queue-size for the Class 1 with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{link}}$ do not have influence on the simulation results. ................................................................. 51

Figure 7.1-5 Maximum in kmin queue-size for the Class 1 with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{link}}$ do not have influence on the simulation results. ................................................................. 52

Figure 7.1-6 Variance in kmin queue-size for the Class 1 with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{link}}$ do not have influence on the simulation results. ................................................................. 53

Figure 7.2-1 Expected kmin queue-size for the Class 1 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 54

Figure 7.2-2 Maximum in kmin queue-size for the Class 1 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 55

Figure 7.2-3 Variance in kmin queue-size for the Class 1 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 56

Figure 7.2-4 Expected kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant. ................................................................. 57

Figure 7.2-5 Maximum in kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant. ................................................................. 58

Figure 7.2-6 Variance in kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant. ................................................................. 59

Figure 7.2-7 Expected kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 60

Figure 7.2-8 Maximum in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 61

Figure 7.2-9 Variance in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 62

Figure 7.2-10 Expected kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant. ................................................................. 63

Figure 7.2-11 Maximum in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant. ................................................................. 64
List of Figures

Figure 7.2-12 Variance in $k_{\text{min}}$ queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 1.0$ is constant. ................................................................. 65

Figure 7.2-13 Expected $k_{\text{min}}$ queue-size for the Class 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 67

Figure 7.2-14 Maximum in $k_{\text{min}}$ queue-size for the Class 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 68

Figure 7.2-15 Variance in $k_{\text{min}}$ queue-size for the Class 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 69

Figure 7.2-16 Expected $k_{\text{min}}$ queue-size for the Class 4 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 70

Figure 7.2-17 Maximum in $k_{\text{min}}$ queue-size for the Class 4 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 71

Figure 7.2-18 Variance in $k_{\text{min}}$ queue-size for the Class 4 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 72

Figure 7.2-19 Expected $k_{\text{min}}$ queue-size for the Class 5 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 73

Figure 7.2-20 Maximum in $k_{\text{min}}$ queue-size for the Class 5 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 74

Figure 7.2-21 Variance in $k_{\text{min}}$ queue-size for the Class 5 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 75

Figure 7.2-22 Expected $k_{\text{min}}$ queue-size for the Class 6 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 76

Figure 7.2-23 Maximum in $k_{\text{min}}$ queue-size for the Class 6 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 77

Figure 7.2-24 Variance in $k_{\text{min}}$ queue-size for the Class 6 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant. ................................................................. 78
Chapter 1 Introduction

1. Introduction: Motivation, Goal and Thesis Layout

Due to the introduction of real-time applications the need for guaranteed QoS has increased over the past few years. In order to provide guaranteed QoS an exact QoS routing algorithm is needed. The task of an exact QoS algorithm is to find a path through a network that will satisfy constraints on multiple QoS metrics. This means that an exact QoS routing algorithm should solve the MCP problem. For some time already, it is known that the MCP problem is an NP-complete problem. On the contrary, NP complete behaviour seems only to occur in a network with a specific correlation structure between the link weights. Hence, the goal of this thesis is to investigate the influence of specific path correlation structures on the complexity of QoS routing. To achieve the goal different path correlation structures will be scrutinized. In detail, the path correlation structure is the correlation between the metrics of all links that constitute a network topology. This means that by systematically studying the path correlation structures, the worst-case complexity of the MCP problem will be analysed and evaluated. SAMCRA, which is proved to be an exact QoS routing algorithm, is used as the path computation algorithm.

In the course of the thesis the following analysis is made. The MCP problem is proven to be NP-complete. The proof of NP-completeness of the MCP problem indicates that specific correlation structures, between link metrics, have influence on its NP-complete behaviour. This implies that NP-completeness of the MCP problem depends very much on the size of the link metrics, level of the correlation between those link metrics and the class of the underlying network topology. Consequently, specific path correlation structures will be generated in order to investigate its influence on the complexity of QoS routing. If necessary, link correlation will also be added to terminate in the search for NP-complete behaviour. In detail, link correlation in a network is the correlation between multiple link metrics. The simulation results, based on the specific path correlation structures, will show the QoS complexity behaviour, suggesting that in practice an exact QoS algorithm may or may not work in polynomial-time. The simulations are performed in a two-dimensional lattice network topology.

The thesis is organized as follows. In Chapter 2, a detailed model of Multi-Constrained QoS routing and its relation to the goal of the thesis is presented. Furthermore, the complexity theory will be described here, where NP-completeness of the MCP problem will be thoroughly discussed. The analysis of the complexity of Multi-Constrained QoS routing in a two-dimensional lattice network topology is presented in Chapter 3. In detail, the properties involving the complexity for the given network topology will be analysed here. In Chapter 4, the covariance and the correlation matrices and their theoretic background are explained. In Chapter 5, details on the generation of correlated vector random variables are given. With the aim of simulating NP-complete behaviour, the configuration of a two-dimensional lattice network topology is explained in Chapter 6. This means that Chapter 6 includes also the proposed models of path correlation classes. In Chapter 7, the simulation results on the complexity of Multi-Constrained QoS routing are provided. Finally, in Chapter 8 the conclusions and some recommendation will be presented and discussed.
2. Multi-Constrained QoS Routing

This chapter reviews a detailed model of QoS routing with constraints on multiple QoS metrics, which is also often denoted as the Multi-Constrained Path (MCP) problem. Its primary purpose is to serve as a glossary of concepts and notations of the MCP problem.

Besides the MCP problem, this chapter includes also a detailed description of the issues that will appear subsequently in the thesis: SAMCRA, an exact QoS routing algorithm, the class of the network topology and the correlation techniques applied on the pairs of variables.

2.1. Introduction to Multi-Constrained QoS Routing

In traditional data networks, like Internet, routing is primarily concerned with connectivity and has no guarantees. Traditional networks are usually characterized by routing protocols, which have single metrics such as delay or hopcount. They use the shortest path algorithms for path computation. The only service provided by traditional networks is the service stated as best-effort service, which is largely controlled by the end-system. This kind of service was adequate when communication was not real-time. In recent years though, several new classes of networked applications, such as real-time multimedia applications over the Internet, have been developed. It is therefore becoming increasingly clear that traditional networks, which often have stringent QoS requirements, are inadequate for these applications. For a network to support and guarantee diverse QoS requirements, it is necessary for routing to have more complex network models. Thus, the task of QoS routing is to find a path in a network that satisfies constraints on multiple metrics, such as bandwidth, delay, hopcount and cost. This type of selection problem is referred to as Multi-Constrained Path (MCP) problem. For a definition of the MCP problem, Section 2.5 can be consulted. The MCP problem has been proven to be NP-complete, indicating that guaranteed QoS routing is impossible. However, the MCP problem is not strong NP-complete. This property implies that in practice an exact QoS algorithm may work in polynomial-time, which in turn suggests that guaranteed QoS routing maybe possible. For a definition of NP-complete and the strong NP-complete problem, Section 2.2 and Section 2.3 can be consulted.

2.2. Introduction to Complexity Theory [4]

The theory of classifying problems based on how difficult they are to solve is called the complexity theory. A problem is assigned to the P-problem (polynomial-time) class if the number of steps needed to solve it, is bounded by some power of the problems size. A problem is assigned to the NP-problem (non-deterministic polynomial-time) class if it permits a non-deterministic solution and the number of steps to verify the solution is bounded by some power of the problems size. The class of P-problems is a subset of the class of NP-problems.
If a problem is known to be NP and a solution to the problem is somehow known, then demonstrating the correctness of the solution can always be reduced to a single P verification. If P and NP are not equivalent, then the solution of NP-problems requires (in the worst-case) an exhaustive search. A problem is said to be NP-hard if an algorithm for solving it can be translated into one for solving any other NP-problem. It is much easier to show that a problem is NP than to show that it is NP-hard. A problem which is both NP and NP-hard is called an NP-complete problem.

### 2.3. NP-complete Problems [4]

The class of NP-complete problems has been the focus of considerable theoretical and practical interest in recent years. This class contains many well-known and much studied problems, e.g. the travelling salesman problem, the knapsack problem, the graph colouring problem, and is characterized by two important properties:

- If we had a polynomial-time algorithm for one of the NP-complete problems, we could obtain polynomial-time algorithms for all the NP-complete problems. A polynomial-time algorithm is defined as one whose time complexity function is \( O(p(n)) \) for some polynomial \( p(n) \), where \( n \) is used to denote the input length.
- Whether all NP-complete problems are intractable has not been proved, but it is commonly acknowledged that no NP-complete problem is known to be solvable by a polynomial-time algorithm.

In the light of these two properties it is widely conjectured that no polynomial-time algorithm can solve the NP-complete problem. For this reason the NP-complete problems are frequently considered to be computationally intractable [4 et al.]. Proving a problem to be NP-complete is usually done by reducing it to a known NP-complete problem. All other NP-complete problems can therefore be reduced to this (or in theory to any other) NP-complete problem. Therefore if one NP-complete problem can be proved to be intractable or solvable in polynomial-time, this would also hold for the entire class of NP-complete problems. In [4] six other basic NP-complete problems are listed and proved. Of these six problems, the Partition problem is best suited for proving that the MCP-problem is NP-complete. The Partition problem is particularly useful for proving that NP-completeness gives best results for problems involving numerical parameters such as lengths, weights, costs, capacities, etc.

### 2.4. Multi-Constrained QoS Metrics and Constraints [20,9]

Let \( G(N, E) \) denote a network topology, where \( N \) is the set of nodes and \( E \) is the set of links. With a slightly abuse of notation, the \( N \) is also used to denote the number of nodes and the \( E \) for the number of links. The number of QoS metrics is denoted by \( m \). Each link is characterized by a \( m \)-dimensional link weight vector, consisting of \( m \) nonnegative QoS weights, \( \omega_i(u, v), i = 1, \ldots, m, (u, v) \in E \), as components, each reflecting a QoS measure such as delay, jitter, loss, cost, administrative weight, etc. The QoS constraints of an application are expressed in the \( m \)-dimensional vector \( L \). QoS routing algorithm, which will be discussed in the Section 2.8, computes the path \( P \) that obeys multiple constraints, \( \omega_i(P) \leq L_i \) for all \( 1 \leq i \leq m \).
Chapter 2 Multi-Constrained QoS Routing

The possible QoS metrics of a path can belong to two different classes: additive and min-max QoS metrics. For min-max QoS metrics, the path weight of the QoS metric is the minimum (or maximum) of the QoS weights of the links that constitute that path. Examples of min-max metrics are the minimum needed bandwidth and (policy related) transit flags. Routing with min (max) QoS metrics can easily be treated by omitting all links from the topology that do not satisfy one of the requested min (max) QoS constraints. In contrast, constraints on additive QoS metrics cause more difficulties. In the case of the additive QoS metrics, the path weight of that metric equals the sum of the QoS weights of the links defining the path. Examples of additive QoS metrics are the delay, the hopcount and the costs. QoS routing with two or more additive QoS metrics is proved to be NP-complete by Wang and Crowcroft [21]. For multiplicative metrics, the value of the QoS metric along a path is the product of the QoS values of the constituent edges of the path. Multiplicative metrics can be transformed into additive weights by using the logarithm. An example of the multiplicative metric is the packet loss.

In the sequel, additive QoS metrics are considered, for which the weight of a path $P = n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_k$ consisting of $k-1$ hops (links) equals the vector-sum of the weights of its constituent links:

$$\omega(P) = \sum_{j=1}^{k-1} \omega(n_j \rightarrow n_{j+1}).$$

A Multi-Constrained QoS path $P_{A \rightarrow B}$ is a path between node $A$ and node $B$ that satisfies $\omega_i(P_{A \rightarrow B}) \leq L_i$ for all $1 \leq i \leq m$. The following section proceeds with more formal definitions.

2.5. Multi-Constrained Path Problem [20,9]

The Multi-Constrained Path (MCP) Problem and the Multi-Constrained Optimal Path (MCOP) problem are the instances of QoS routing.

**Definition 2.5-1. Multi-Constrained Path (MCP) problem**

Consider a network $G(N, E)$. Each link in a network, $(u, v) \in E$, is specified by a link weight vector with as components $m$ additive QoS weights $\omega_i(u, v) \geq 0$, $i = 1, \ldots, m$. Given $m$ constraints $L_i, i = 1, \ldots, m$ the problem is to find a path $P$ from a source node $A$ to a destination node $B$ such that

$$\omega(P) \overset{\text{def}}{=} \sum_{(u, v) \in P} \omega_i(u, v) \leq L_i \text{ for } i = 1, \ldots, m.$$ 

There may be multiple different paths in a network $G(N, E)$ that satisfy the constraints. Such paths are said to be feasible. According to Definition 2.5-1 any of these paths is a solution to the MCP problem. However, it might be desirable to retrieve the path with smallest length $l(P)$ from the set of feasible paths. The precise definition of length $l(.)$ will be discussed in Section 2.8. This more difficult problem that additionally optimises some length function $l(.)$ is called the Multi-Constrained Optimal Path (MCOP) Problem.
**Definition 2.5-2. Multi-Constrained Optimal Path (MCOP) problem**

Consider a network \( G(N, E) \). Each link, \((u, v) \in E\), is specified by a link weight vector with as components \( m \) additive QoS weights \( \omega_i(u, v) \geq 0, i = 1, \ldots, m \). Given \( m \) constraints \( L_i, i = 1, \ldots, m \) the problem is to find a path \( P \) from a source node \( A \) to a destination node \( B \) satisfying Definition 2.5-1 and, in addition, minimizing some length criterion such that \( l(P) \leq l(P') \), for all paths \( P' \) and \( P \) between \( A \) and \( B \) that satisfy the constraints.

**2.6. Multi-Constrained Path Problem is NP-complete [9,21]**

The proof, which first was given by [4] and later on was provided by [21], shows that the MCP problem for \( m \geq 2 \) is NP-complete. Basically, they have reduced the MCP problem for \( m = 2 \) to an instance of the partition problem and have showed that partition is \( \propto 2\text{CP} \) problem. The partition problem is a well-known NP-complete problem. It gives best results for problems involving numerical parameters such as lengths, weights, costs, capacities, etc.

In the proof the following approach is used to show that that 2CP problem is NP-complete: Since partition is a well-known NP-complete problem, the proof shows that partition \( \propto 2\text{CP} \) problem to prove its NP-completeness.

**Definition 2.6-1. Partition Problem**

Given a finite set \( A \) and a size \( s(a) \in \mathbb{Z}^+ \) for each \( a \in A \). Is there a subset \( A' \subseteq A \) such that

\[
\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a).
\]

**Theorem 2.6-1. The MCP problem is NP-complete**

Given a network \( G(N, E) \) with for each link \((u, v) \in E\), \( m \) additive weights \( \omega_i(u, v), i = 1, \ldots, m \), the source node \( s \) and the destination node \( d \) and \( m \) constraints \( L_i, i = 1, \ldots, m \), where \( m \geq 2, \omega_i(u, v) \geq 0, L_i \geq 0 \). The problem of deciding if there is a simple path \( P \) from source \( s \) to destination \( d \) that satisfies the following constraints \( \omega_i(u, v) \leq L_i, i = 1, \ldots, m \), implicating to the MCP-problem, is NP-complete.

**Proof of Theorem 2.6-1.**

Given a chain topology with \( i + 1 \) nodes and \( 2i \) links, each with a two-component weight vector \( \bar{\omega} \) as depicted in Figure 2.6-1 and a set of numbers \( a_i \in A, 0 \leq a_i \leq S \), for \( i = 1, \ldots, n \), where \( S = \sum_{i=1}^{n} a_i \). The constraints are chosen as follows: \( L_1 = iS - S/2 \), and \( L_2 = S/2 \). To solve the MCP problem, a path from node 1 to node \( i + 1 \), that obeys the constraints, has to be found. Since, for all link weight vectors, the sum of the components equals \( S \), the following is valid: \( \omega_1(P) + \omega_2(P) = iS \). Accordingly, a solution satisfying the constraints is only found if \( \omega_1(P) = iS - S/2 \) and \( \omega_2(P) = S/2 \). The problem has now become an instance of the above described NP-complete partition problem and can only be solved by finding the set \( A' \subseteq A \), for which \( \sum_{a_i \in A'} a_i = S/2 \). A feasible path exists if the set \( A' \) exists, in which case, the set is retrieved by choosing the lower link if \( a_i \in A' \) and the upper link if \( a_i \not\in A' \).
Chapter 2 Multi-Constrained QoS Routing

![Figure 2.6-1](image)

The assignment of link weights to the links in the chain topology between nodes $i$ and $i+1$.

Furthermore, the proof also shows that the MCP problem with $m = n$ is the MCP-problem with $m = n + 1$. Herein, the constraint $L_{n+1}$ is aimed to be a large number, say

$$L_{n+1} = \sum_{(u,v) \in E} \omega_{n+1}(u,v),$$

therefore $\omega_{n+1} \leq L_{n+1}$ holds for any path $P$, meaning that $\omega_i(P) \leq L_i$, $i = 1, \ldots, n$ if and only if $\omega_i(P) \leq L_i$, $i = 1, \ldots, n+1$ holds. This completes the proof. Details on the Proof of Theorem 2.6-1 can be found in [21].

2.7. Non NP-complete Multi-Constrained Path Problems [9]

**Property 2.7-1.**
If all the components of a link weight vector are the same, the MCP problem is solvable in polynomial-time with a standard one-dimensional shortest path algorithm. This property can be seen as an extreme case of positive link correlation: the $m$ link weights on a link are correlated with coefficient $\rho = 1$.

**Property 2.7-2.**
If $\omega_i(u,v) = \omega_i'(u',v')$ for $i = 1, \ldots, m$, for all $(u,v),(u',v') \in E$, then the MCP problem is solvable in polynomial-time. This property can be seen as an extreme case of positive path correlation: the $i$-th weights of all links are correlated to each other with coefficient $\rho = 1$.

**Property 2.7-3.**
If the weight vector of a link,

$$\omega = \begin{bmatrix} \varphi_1(x_i) \\ \vdots \\ \varphi_m(x_i) \end{bmatrix},$$

with $\varphi_j(x_i)$ concave functions, is a function of a single parameter $x_i$ and if $P$ is the shortest path from $s$ to $d$ in $G_x$ (the graph where the links have only weight $x_i$) with length $X = \sum_{j=1}^i x_j$ and hopcount $h$, then $P$ in $G_\omega$ (the graph where each link $i$ has a weight vector $\omega$) satisfies the constraint vector $L$ if

$$X \leq h\varphi_j\left(\frac{L_j}{h}\right), \quad 1 \leq j \leq m.$$

The proof of the Property 2.7-3 can be found in [9].
Chapter 2 Multi-Constrained QoS Routing

Property 2.7-4.
If the weight vector of a link,

\[ \omega = \begin{bmatrix} f_1(x_i) \\ \vdots \\ f_m(x_i) \end{bmatrix} \]

with \( f_j(x_i) \) convex functions, is a function of a single parameter \( x_i \) and if \( P \) is the shortest path from \( s \) to \( d \) in \( G_s \) (the graph where the links have only weight \( x_i \)) with length \( X = \sum_{i=1}^{h} x_i \) and hopcount \( h \), then \( P \) violates the constraints in \( G_w \) (the graph where each link \( i \) has a weight vector

\[ \omega = \begin{bmatrix} f_1(x_i) \\ \vdots \\ f_m(x_i) \end{bmatrix} \]

where \( x_i \) is the common link parameter) if \( X > hf_j^{-1}\left(\frac{L_j}{h}\right) \) for at least one \( j \). The proof of the Property 2.7-4 can be found in [9].

Property 2.7-5.
If the weight vector of a link,

\[ \omega = \begin{bmatrix} g_1(x_i) \\ \vdots \\ g_m(x_i) \end{bmatrix} \]

with \( g_j(x_i) \) monotone increasing and \( P \) is the shortest minimum hop path from \( s \) to \( d \) in \( G_s \) (the graph where the links have only weight \( x_i \)) and \( x_i \leq x'_i \), where \( x'_i \) is the \( i \)-th ordered common link weight of an other path \( P' \) from \( s \) to \( d \) in \( G_s \), then \( P \) is also the shortest path in \( G_w \) (the graph where each link \( i \) has a weight vector

\[ \omega = \begin{bmatrix} f_1(x_i) \\ \vdots \\ f_m(x_i) \end{bmatrix} \]

where \( x_i \) is the common link parameter). The proof of the Property 2.7-5 can be found in [9].
2.8. SAMCRA Algorithm [20]

2.8.1. Four Concepts of SAMCRA

Network routing essentially consists of two identities, the routing protocol and the routing algorithm. The routing protocol supplies each node in a network with a consistent view of that topology and, in some cases, of its resources at some moment in time. The routing protocol deals with the complex dynamic processes such as topology updates, determination of significant changes and the flooding of topology information to each node in a network. The routing algorithm provides the intelligence to compute a path from source to destination possibly subject to constraints and mostly optimising some criterions. The routing algorithm, SAMCRA, which is proven to be an exact QoS algorithm, is the main focus of this section.

SAMCRA (Self Adapting Multiple Constraints Routing Algorithm) is the successor of TAMCRA (Tuneable Accuracy Multiple Constraints Algorithm). As opposed to TAMCRA, SAMCRA guarantees to find a path within the constraints, provided such path exists. Furthermore, SAMCRA only allocates queue space when truly needed, whereas in TAMCRA the allocated queue space is predefined. The four fundamental concepts of SAMCRA are:

- **The non-linear path length.**
  The length of a path $P$ is defined as follows
  \[
  l(P) = \max_{1 \leq i \leq m} \left( \frac{\omega_i(P)}{L_i} \right) \text{ where } \omega_i(P) = \sum_{(u,v)\in P} \omega_i(u,v).
  \]
  A definition of the path length has to be non-linear in order to guarantee that a retrieved path lies within the constraints, i.e. $l(P) \leq 1$. A solution to the MCP problem is a path whose link weights are all within the constraints, $\omega_i(P) \leq L_i$, for all $i = 1, \ldots, m$. By using the above length function, all QoS measures are considered as equally important. An important corollary of the non-linear path length is that the subsections of the shortest path in multiple dimensions are not necessary shortest paths. As a consequence, possibly more than one path in each node needs to be considered, leading to the $k$-shortest path approach.

- **The $k$-shortest paths approach.**
  The $k$-shortest path algorithm is essentially Dijkstra’s algorithm that does not stop when the destination is reached, but continues until the destination has been reached $k$ times. If $k$ is not restricted, the $k$-shortest path algorithm returns all possible paths between source and destination. The $k$-shortest path approach is applied to intermediate nodes on the path from the source node to the destination node, where the track of multiple subpaths from the source node to those intermediate nodes is kept. Not all paths are stored, but an efficient distinction based on non-dominance is made.

- **The principle of non-dominated paths.**
  A (sub)path $Q$ is dominated by a (sub)path $P$ if $\omega_i(P) \leq \omega_i(Q)$ for $i = 1, \ldots, m$ with an inequality for at least one link weight component $i$. SAMCRA only considers non-dominated (sub)paths. The principle of the non-dominance enables an efficient reduction in the search space (all paths between source and destination) without compromising the solution.
• The Look-Ahead concept.
  Besides path dominance, the look-ahead concept can be viewed as an additional
  mechanism to reduce the search space of possible paths. The idea is to further limit the set
  of possible paths by using information of the remaining (sub)paths towards the
  destination. Actually, the look-ahead concept proposes to compute the shortest path tree
  rooted at the destination to each node \( n \) in the graph for each of the \( m \) link weights
  separately. Hence, for each link weight component, \( 1 \leq i \leq m \), the lowest value from the
  destination to a node \( n \) is stored in the queue of that node \( n \). In total, Dijkstra’s shortest
  path algorithm is executed \( m \) times resulting in \( N-1 \) vectors with shortest values for
  each link weight component from a node \( n \) to the destination. The basic importance of
  look-ahead is to provide each node \( n \) with an exact, attainable low
  bound of \( \omega_i(P_{A \rightarrow \delta}) \)
  for each individual link weight component \( i \).

2.8.2. Meta-Code of SAMCRA

The main algorithm (see Figure 2.8.2-1) starts with the execution of the subroutine
\textit{INITIALIZE} in line 1. The subroutine \textit{INITIALIZE} initialises the necessary parameters for
the main algorithm and computes the look-ahead information. Provided the queue \( Q \) is not
empty (otherwise no feasible path is present), the \textit{EXTRACT_MIN} function in line 3 selects
the minimum path length in the queue \( Q \) and returns \( u[i] \), the \( i \)-th path \( P_{A \rightarrow B} \) stored in the
queue at node \( u \). With these numbers and the predecessor list \( \pi \), the entire path can be
reconstructed via back tracing. The extracted path is marked \textit{GREY} in line 4. If the node \( u \),
corresponding to the extracted path \( u[i] \), equals the destination \( B \), the shortest path satisfying
the constraints is returned. If \( u \neq B \), the scanning procedure is initiated in line 8. Line 8
describes how the \( i \)-th path up to node \( u \) is extended towards its neighbouring node \( \nu \),
except for the previous node where it came from. The previous node on the path \( u[i] \) is stored
in the predecessor list \( \pi \). Returning to this previous node induces a loop, which must be
avoided. Since the link weights are non-negative, paths without loops always dominate paths
that have a loop. This property relieves us from the time-consuming task of storing/back
tracing the entire path \( u[i] \) to avoid loops.

Line 9 invokes the \textit{FEASIBILITY} subroutine to check whether all stored paths at node \( \nu \) are
non-dominated and obey \textit{maxlength}. \textit{FEASIBILITY} also checks whether the new extended
path is not dominated by previously stored paths at node \( \nu \). In line 10 the length of the
predicted end-to-end path weight vector (composed of the real subpath weight vector from \( A \) to \( \nu \) plus the lower bound vector from \( \nu \) to \( d \) ) is calculated. Line 11 tests if the new
extended path is non-dominated and has a \textit{predicted_length} \( \leq \textit{maxlength} \). If this is the case
it can be stored and the queue must be updated (line 12). Removing paths for which
\textit{predicted_length} \( > \textit{maxlength} \) is the search space reduction of the look-ahead concept.
Updating is performed by the subroutine \textit{UPDATEQUEUE}, which has the task of updating
the queue \( Q \) with a new path, namely the extended path from \( u[i] \) to node \( \nu \). Finally,
\textit{maxlength} can be updated in lines 13-14.

Meta codes of the subroutines: \textit{INITIALIZE}, \textit{FEASIBILITY} and \textit{UPDATEQUEUE} can be
found in [20].
Chapter 2 Multi-Constrained QoS Routing

\[ \text{SAMCRA}(G, m, A, B, L) \]

1. \text{INITIALIZE}(G, m, A, B) \rightarrow \vec{b}
2. \text{while} (Q \neq \emptyset)
3. \text{EXTRACT } _\text{MIN}(Q) \rightarrow u[i]
4. \text{if } (u = B)
5. \text{STOP} \rightarrow \text{return } \text{path}
6. \text{else}
7. \text{for each } v \in \text{Adj}[u] \setminus \{\pi[u[i]], A\}
8. \text{FEASIBILITY}(G, u, i, v, counter, d, \omega, \text{maxlength}) \rightarrow \text{dominated}
9. \text{predicted } _\text{length} \leftarrow l(\vec{d}[u[i]] + \vec{\omega}(u \rightarrow v) + \vec{b}[v])
10. \text{if } (\text{predicted } _\text{length} < \text{maxlength AND domiated } \neq 1)
11. \text{UPDATEQUEUE}(Q, u, i, v, j, d, \omega, \pi, \text{counter}[v], \text{predicted } _\text{length})
12. \text{if } (v = B \text{ AND predicted } _\text{length} < \text{maxlength})
13. \text{maxlength} \leftarrow \text{predicted } _\text{length}

**Figure 2.8.2-1** Meta-code of SAMCRA.

### 2.8.3. Complexity of SAMCRA

The worst-case complexity of SAMCRA as presented in the meta-code in the previous section will be explained. First, the worst-case complexity of the subroutines is determined, after which the total worst-case complexity of SAMCRA will be computed.

The total worst-case complexity of SAMCRA is constructed as follows. The \text{INITIALIZE} function adds \( O(mN \log N + mE + m^2N) \). The queue \( Q \) can never contain more than \( k_{\text{max}}N \) path lengths. \( k_{\text{max}} \) is the worst-case number of parameters and is

\[
k_{\text{max}} = \min \left[ \prod_{i=1}^{m} L_i \right]
\]

where the second argument of the min operator denotes the maximum number of paths that exists between two nodes in any graph. The second bound applies in the case the granularity is infinitely small or, equivalently, for real values of \( \omega_j \).

When using a Fibonacci or Relaxed heap to structure the queue, selecting the minimum path length among \( k_{\text{max}}N \) different path lengths takes at most a calculation time of the order of \( O(\log(k_{\text{max}}N)) \). As each node can be selected at most \( k_{\text{max}} \) times from the queue, the \text{EXTRACT } _\text{MIN} function in line 3 takes \( O(k_{\text{max}}N \log(k_{\text{max}}N)) \) at most. Returning a path in line 6 takes at most \( O(N) \). The for-loop starting on line 8 is invoked at most \( k_{\text{max}} \) times from each side of each link in the graph, leading to \( O(k_{\text{max}}E) \). \text{FEASIBILITY} takes \( O(k_{\text{max}}m) \). Calculating the length in line 10 takes \( O(m) \) and updating the queue takes \( O(k_{\text{max}}m) \). Combining all those contributions yields a total worst-case complexity of SAMCRA of
Chapter 2 Multi-Constrained QoS Routing

\[ O(mN \log N + mE + m^2 N + N + k_{max}^N \log(k_{max} N) + k_{max}^2 mE) \] or

\[ C_{SAMCRA} = O(k_{max}^N \log(kN) + k_{max}^2 mE) \]

where \( m \) is fixed. It should be noticed that for a single constraint, meaning \( m = 1 \), and \( k_{max} = 1 \), \( C_{SAMCRA} \) reduces to the complexity of the Dijkstra algorithm

\[ C_{Dijkstra} = O(N \log N + E). \]

When the link weights are real numbers, the granularity is infinitely small implying that \( k_{max} = O(N!) = O(\exp(N \ln N)) \). In this case, the QoS routing problem is NP-complete. But, in practice these measures will have finite granularity such that link weights \( \omega_i \) are integers. Hence, \( k_{max} \) is limited by the first, finite argument in \( k_{max} \), which does not depend on the size of the topology. This means, for a fixed number of constraints \( m \) and the finite granularity in the constraints, SAMCRA has pseudo-polynomial-time complexity.

2.9. Correlation Techniques [14]

Correlation is a statistical technique, which can be used to show whether and how strongly pairs of variables are related. Like all statistical techniques, correlation is only appropriate for certain kinds of data. Correlation works for data in which numbers are meaningful, usually quantities of some sort. There are several different correlation techniques. In the thesis the most common types are used: the Pearson correlation and the intra-class correlation coefficient (ICC). The ICC is used as a measure of the reliability of a variable, whereas the Pearson is used for the validity of the variable. The values of the Pearson and intra-class correlation coefficients are usually similar for the same set of data.

Pearson’s correlation is the usual measure of correlation, sometimes called product-moment correlation or the interclass correlation. Pearson’s correlation is a measure of association which varies from \(-1\) to \( +1 \) with \( 0 \) indicating no relationship and \( +1 \) indicating perfect relationship taking the form, the more the \( x \), the more the \( y \), and vice versa. A value of \(-1\) is a perfect negative relationship taking the form, the more the \( x \), the less the \( y \), and vice versa.

The Pearson’s correlation coefficient is defined by

\[ \rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_i^2 \sigma_j^2}} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \]

where \( \sigma_{ij} \) is the covariance between the two variables \( i \) and \( j \), \( \sigma_i^2 (\sigma_j^2) \) denotes the variance of the variable \( i(\ j) \) and \( \sigma_i (\sigma_j) \) denotes the standard deviation\(^1\) of the variable \( i(\ j) \).

---

\(^1\) Standard deviation (\( \sigma \)) is the square root of the variance (\( \sigma^2 \)).
Chapter 2 Multi-Constrained QoS Routing

The *coefficient of determination* is the square of the *Pearson’s correlation coefficient*. It represents the percent of the variance in the dependent variable explained by the independent variable. Of course, since correlation is bi-directional, the square of the Pearson correlation is also the percent of the variance in the independent variable explained by the dependent variable.

The *coefficient of intraclass correlation (ICC)* is a type of correlation that measures the relative homogeneity within groups in ratio to the total variation. In general ICC is a coefficient, which approaches +1.0 as the between groups effect (the row effect) is very large relative to the within groups effect (the column effect) whatever the rows and columns represent. In this way ICC is a measure of homogeneity: it approaches +1.0 when any given row tends to have the same values for all columns and ICC is 0 when within groups variance equals between groups variance, indicating the grouping variable having no effect. A negative ICC occurs when between group effect is less than within group effect, indicating some third variable has introduced non-random effects on the different groups.

ICC has the maximum value of +1, but its maximum negative value is $-1/(n-1)$, meaning that ICC ranges $-1/(n-1) \leq ICC \leq +1$, where $n$ is the number of columns in a matrix.

2.10. Network Topology

Routing is an operation that acts upon a network topology (graph). A network topology is denoted by $G(N, E)$, where $N$ is a set of nodes (vertices) connected by a set of links (edges) $E$. The network topology used in the thesis is a *two-dimensional lattice (grid) network topology (graph)*. In Figure 2.10-1 an example of a two-dimensional lattice network topology is given. Presented network topology consists of 16 nodes connected by 24 links. The source node is chosen in the upper left corner and the destination node in the lower right corner. This configuration of the network topology is chosen for the reason that it contains the chain structure, which is illustrated in bold lines. The chain structure yields the longest minimum hopcount, causing NP-complete behaviour to emerge. Therefore, this location of the source and destination nodes leads to the worst-case behaviour of Multi-Constrained QoS routing.

Figure 2.10-1 gives an example of a two-dimensional lattice network topology consisting of $N=16$ nodes and $E=24$ links. Number of links in a network topology is $E = 2 \times \sqrt{N} \times (\sqrt{N} - 1)$. In the example, every internal node has degree 4 while nodes on the boundaries have degree 3. The four corner nodes have degree 2. Figure 2.10-1 demonstrates also the assignment of link numbers, completed by the simulation program.
Chapter 2 Multi-Constrained QoS Routing

Figure 2.10-1  Example of a two-dimensional lattice network topology with 16 nodes and 24 links.

A $m$-dimensional link weight vector characterizes each link with $m = 2$ QoS weights as components. This means that a network topology consists of $E$ link weight vectors. The first $E$ components of the link weight vector form the path correlation vector $\mathbf{Z}[1, \ldots, E]$, which represents the correlation between $E$ links in a network or the path correlation. The second $E$ components of the link weight vector form the link correlation vector $\mathbf{X}[1, \ldots, E]$, which represents the correlation between the link weights on each link or the link correlation.

The path and link correlation vectors are generated following two different methods. In the first method the path correlation vector $\mathbf{Z}[1, \ldots, E]$ with zero-mean, unit-variance and correlated Gaussian random variables is generated from a vector consisting of zero-mean, unit-variance and uncorrelated Gaussian random variables. Subsequently, the link correlation vector is correlated in the same way to the path correlation vector. For a detailed description on the generation of correlated vector random variables, Chapter 5 can be consulted. In the second method the components of path and link correlation vectors are generated via the non-dominance property, creating uniformly distributed correlated random link weights. For a detailed description on the generation of correlated link weights via the non-dominance property, Chapter 3 can be consulted.
3. **Analysing Complexity of Multi-Constrained QoS Routing**

In this chapter the worst-case complexity of Multi-Constrained QoS routing will be analysed. This includes the properties, which may lead to NP-complete behaviour and the properties, which may reduce the complexity of Multi-Constrained QoS routing. Analysed complexity is first presented for a chain topology. Subsequently, the chain topology is used as the framework for analysing two-dimensional lattices.

### 3.1. Properties Involving Complexity of Multi-Constrained QoS Routing [9]

In general, networks have the potential to induce NP-complete behaviour if the following two properties are obeyed:
- The class of a network topology, which contains a chain like structure.
- Specific structures of a two-dimensional link weight vectors, where the correlation between the weights is negative.

In general, there are two important properties that can reduce the search space when solving the MCP problem without losing exactness:
- Non-dominance property. If for two paths $P_1, P_2$ from source $s$ to destination $i$ holds that $P_1 \leq P_2$, then all weights of $P_1$ are smaller (or equal) than those of $P_2$ and hence $P_2$ can be omitted from the search space. Search is continued with $P_1$.
- The values of the constraints. If a sub-path $P$ from source $s$ to node $i$ exceeds one or more constraints, it can never become a feasible path, because the path weight vector from $i$ to destination node $d$ consists of non-negative weights. Similar, if for two paths $P_1, P_2$ from $s$ to $i$ holds that $P_1 \leq P_2$, then all weights of $P_1$ are smaller (or equal) than those of $P_2$ and hence $P_2$ can be omitted from the search space. Search is continued with $P_1$.

This chapter proceeds with the characterization of the two-dimensional link weight vector structure in the chain topology, for which NP-complete behaviour emerge.

### 3.2. Non-dominance Property in Chain Network Topology [9]

**Property 3.2-1.**
The link weights are chosen such that $a_i > c_i$ and $b_i < d_i$, for $i = 1,...,N$ ( $c_i > a_i$ and $d_i < b_i$ would also have been possible). If, in a chain topology, there holds that

$$a_i - c_i > \sum_{j=0}^{i-1} (a_j - c_j)$$

$$b_i - d_i > \sum_{j=0}^{i-1} (b_j - d_j)$$

for $i = 1,...,N-1$, where $a_0 = b_0 = c_0 = d_0 = 0$, then all $2^{N-1}$ paths from node 1 to node $N$ are non-dominated. It can be verified that if $a_i \geq c_i$ and $b_i \geq d_i$ or $c_i \geq a_i$ and $d_i \geq b_i$ were allowed, this would lead to dominance. For the proof of the Property 3.2-1, [9] can be consulted.
Corollary 3.2-1.
Property 3.2-1 is a sufficient but also necessary condition for all paths in the chain topology to be non-dominated. For the proof of the Corollary 3.2-1, [9] can be consulted.

Corollary 3.2-2.
If there are more than two links (all with two weights) between two nodes in the chain topology, Property 3.2-1 should hold for all possible pairs of links, in order for all paths from node 1 to node N to be non-dominated.

Property 3.2-2.
For the chain topology, besides the Property 3.2-1, the constraints must lie in the range:
\[
\sum_{j=0}^{N-1} c_j \leq L_1 \leq \sum_{j=0}^{N-1} a_j
\]
\[
\sum_{j=0}^{N-1} d_j \leq L_2 \leq \sum_{j=0}^{N-1} b_j
\]
for NP-complete behaviour to occur. Since \( c_i < a_i \), the shortest path for measure 1 from node 1 to node N equals \( \sum_{j=0}^{N-1} c_j \). If \( L_1 < \sum_{j=0}^{N-1} c_j \), then no feasible path exists. If \( L_1 > \sum_{j=0}^{N-1} a_j \), then all possible (loop-free) paths can obey this constraint. The same reasoning applies to \( L_2 \).

Corollary 3.2-3.
If there are more that two links (all with the two links) between two nodes in the chain topology, property 3.2-1 should hold for all possible pair of links, in order for all paths from node 1 to node N to be non-dominated.

Corollary suggest that the links in the chain topology may be seen as two different subpaths from node i to node i+1, what indeed is the case by two-dimensional lattices. This chapter proceeds with the characterization of the two-dimensional link weight vector structure in a two-dimensional lattice network topology, for which NP-complete behaviour emerge.

3.3. Non-Dominance Property in Two-Dimensional Lattice Network Topology

The chain topology as depicted in Figure 3.2-1 will be used to aid in the search for specific path correlation structures in a two-dimensional lattice network topology, for which all paths from source s to destination d are non-dominated. The non-dominated paths are created, besides the main reason that already is mentioned in Section 3.1, for the reason that the non-dominance property induces the correlation between the path weights.
Property 3.3-1.
If the link weights are chosen such that there holds (see Figure 3.3-1 and Figure 3.3-2)
\[
Z_{\text{lower, path}} - Z_{\text{upper, path}} > \sum_{j=0}^{i-1} (Z_{j, \text{lower, path}} - Z_{j, \text{upper, path}}) \\
X_{\text{lower, path}} - X_{\text{upper, path}} < \sum_{j=0}^{i-1} (X_{j, \text{lower, path}} - X_{j, \text{upper, path}})
\]
for all \(i\)

for all underlying chain topologies in a two-dimensional lattice network topology, where
\[
Z_{0, \text{lower, path}} = Z_{0, \text{upper, path}} = X_{0, \text{lower, path}} = X_{0, \text{upper, path}} = 0
\]
and the Property 3.2-1 is obeyed for the main chain topology, then for other existing chain topologies there holds that (see Figure 3.3-3)
\[
a_{i, \text{lower, path}} > a_{i, \text{upper, path}} \\
b_{j, \text{lower, path}} < b_{j, \text{upper, path}}
\]
for all paths in those chain topologies from node 1 to node \(N\) to be non-dominated.

---

Figure 3.3-1  Two-dimensional lattice network topology with the main underlying chain topology, illustrated in bold lines.

Figure 3.3-2  Two-dimensional lattice network topology with other underlying chain topology, illustrated in bold lines.

Figure 3.3-3  Two-dimensional lattice network topology with links, illustrated in bold lines, which have to satisfy the non-dominance property if for the main underlying chain topology the Property 3.2-1 is obeyed.
Example 3.3-1.

![Diagram](image)

Figure 3.3-4 Example of a two-dimensional lattice network topology with two link weights on each link creating vectors $Z[1,\ldots,E]$ and $X[1,\ldots,E]$. This example has four different chain topologies. Here, only the main chain topology is illustrated in bold lines.

If in all underlying chain topologies the link weights are chosen such that the concept of non-dominance is satisfied, then the following equations for the first link weights could be created:

**1st and main chain topology**

$$z_2 + z_6 > z_1 + z_4$$
$$z_{11} + z_{17} > z_{10} + z_{13} > z_2 + z_6 > z_1 + z_4$$
$$z_{20} + z_{24} > z_{19} + z_{21} > z_{11} + z_{17} > z_{10} + z_{13} > z_2 + z_6 > z_1 + z_4$$

**2nd chain topology**

$$z_2 + z_9 + z_{15} + z_{17} > z_1 + z_3 + z_6 + z_{13}$$
$$z_{20} + z_{24} > z_{19} + z_{21} > z_2 + z_9 + z_{15} + z_{17} > z_1 + z_3 + z_6 + z_{13}$$

**3rd chain topology**

$$z_2 + z_6 > z_1 + z_4$$
$$z_{11} + z_{18} + z_{23} + z_{24} > z_{10} + z_{12} + z_{14} + z_{21} > z_2 + z_6 > z_1 + z_4$$

**4rd chain topology**

$$z_2 + z_9 + z_{16} + z_{22} + z_{23} + z_{24} > z_1 + z_3 + z_5 + z_7 + z_{14} + z_{21}$$
Chapter 3 Analysing Complexity of Multi-Constrained QoS Routing

1st and main chain topology
\[(z_2 + z_8) > (z_1 + z_4)\]
\[(z_{11} + z_{17}) > (z_{10} + z_{13})\]
\[(z_{20} + z_{24}) > (z_{19} + z_{21})\]

2nd chain topology
\[(z_2 + z_9 + z_{15} + z_{17}) > (z_1 + z_3 + z_6 + z_{13})\]
\[(z_{20} + z_{24}) > (z_{19} + z_{21})\]

3th chain topology
\[(z_2 + z_8) > (z_1 + z_4)\]
\[(z_{11} + z_{18} + z_{23} + z_{24}) > (z_{10} + z_{12} + z_{14} + z_{21})\]

4th main chain topology
\[(z_2 + z_9 + z_{16} + z_{22} + z_{23} + z_{24}) > (z_1 + z_3 + z_5 + z_7 + z_{14} + z_{21})\]

If the Property 3.2-1 for the main chain topology is obeyed, then the following equations hold

\[ (z_2 > z_1) \text{ and } (z_8 > z_4) \]
\[ (z_{11} > z_{10}) \text{ and } (z_{17} > z_{13}) \]
\[ (z_{20} > z_{19}) \text{ and } (z_{24} > z_{21}) \]
Chapter 3 Analysing Complexity of Multi-Constrained QoS Routing

Subsequently, as a consequence of the property

\[ \text{if } a < b \text{ and } c < d \text{ then } a + c < b + d \]

for other underlying chain topologies, the following equations could be created

1st and main chain topology

\[
\begin{align*}
(z_2 > z_1) & \quad \text{and} \quad (z_8 > z_4) \\
(z_{11} > z_{10}) & \quad \text{and} \quad (z_{17} > z_{13}) \\
(z_{20} > z_{19}) & \quad \text{and} \quad (z_{24} > z_{21})
\end{align*}
\]

2nd chain topology

\[
(z_2 + z_9 + z_{15} + z_{17}) > (z_3 + z_6 + z_{13})
\]

3rd chain topology

\[
(z_{11} + z_{16} + z_{23} + z_{24}) > (z_{12} + z_{14} + z_{21})
\]

4th chain topology

\[
(z_2 + z_9 + z_{16} + z_{22} + z_{23} + z_{24}) > (z_3 + z_5 + z_7 + z_{14} + z_{21})
\]

Finally, for all paths in a two-dimensional lattice network topology to be non-dominated, the following should hold for the components of the correlation vectors \( Z[1...E] \) and \( X[1...E] \) (see also Figure 3.3-5)

1st and main chain topology

\[
\begin{align*}
(z_2 > z_1) & \quad \text{and} \quad (z_8 > z_4) \\
(z_{11} > z_{10}) & \quad \text{and} \quad (z_{17} > z_{13}) \\
(z_{20} > z_{19}) & \quad \text{and} \quad (z_{24} > z_{21})
\end{align*}
\]

2nd chain topology

\[
(z_9 + z_{15}) > (z_3 + z_6)
\]

3rd chain topology

\[
(z_{18} + z_{23}) > (z_{12} + z_{14})
\]

4th chain topology

\[
(z_9 + z_{16} + z_{22} + z_{23}) > (z_3 + z_5 + z_7 + z_{14})
\]
1st and main topology
\[(x_2 < x_1) \text{ and } (x_8 < x_4)\]
\[(x_{11} < x_{10}) \text{ and } (x_{17} < x_{13})\]
\[(x_{20} < x_{19}) \text{ and } (x_{24} < x_{21})\]

2nd chain topology
\[(x_9 + x_{15}) < (x_3 + x_6)\]

3rd chain topology
\[(x_{18} + x_{23}) < (x_{12} + x_{14})\]

4th chain topology
\[(x_9 + x_{16} + x_{22} + x_{23}) < (x_3 + x_5 + x_7 + x_{14})\]

Figure 3.3-5 Example of a two-dimensional lattice network topology with two link weights on each link creating vectors \(Z[1,\ldots, E]\) and \(X[1,\ldots, E]\). The links, which have to satisfy the non-dominance property if the Property 3.2-I for the main underlying chain topology is obeyed, are illustrated in bold lines.
4. Analysing Covariance and Correlation Matrices

In this chapter the covariance and correlation matrices will be analysed. The covariance and correlation matrices are used to generate vectors of jointly Gaussian correlated random variables. The method for vector generation of jointly random variables is presented in the following chapter. What is done in this chapter, includes:

- From the analysis of covariance and correlation matrices, theorems involving covariance and correlation matrix will be presented.
- The most important property of the correlation matrix, the positive definiteness, brings the limits on the given correlation matrix. Therefore, the nearest correlation matrix has to be found.
- Where the given correlation matrix fails to satisfy the valid correlation matrix properties, the nearest correlation matrix will be used to generate vectors of jointly Gaussian random variables.
- This chapter includes also a different method of generating the valid correlation matrix. Here, the valid correlation matrix is generated from the random eigenvalues and its random eigenvectors.

4.1. Analysis of Covariance and Correlation Matrices [3]

Let \( \mathbf{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \) denote a random vector of dimension \( p \). Then

\[
E[\mathbf{X}] = \begin{pmatrix} E[X_1] \\ \vdots \\ E[X_p] \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_p \end{pmatrix} = \mathbf{\mu}
\]

is the mean vector of \( \mathbf{X} \).

The covariance matrix of a \( p \)-variate, random vector \( \mathbf{X} \) is the \( p \times p \) matrix

\[
\text{Cov}[\mathbf{X}] = E[(\mathbf{X} - \mathbf{\mu})(\mathbf{X} - \mathbf{\mu})'] = \begin{pmatrix}
\text{var}[X_1] & \text{cov}[X_1, X_2] & \cdots & \text{cov}[X_1, X_p] \\
\text{cov}[X_2, X_1] & \text{var}[X_2] & \cdots & \text{cov}[X_2, X_p] \\
\vdots & \vdots & \ddots & \vdots \\
\text{cov}[X_p, X_1] & \text{cov}[X_p, X_2] & \cdots & \text{var}[X_p]
\end{pmatrix}
\]

\[
\text{Cov}[\mathbf{X}] = E[(\mathbf{X} - \mathbf{\mu})(\mathbf{X} - \mathbf{\mu})'] = E[\mathbf{XX}' - \mathbf{\mu X}' - \mathbf{X'\mu} + \mathbf{\mu'\mu}'] = E[\mathbf{XX}'] - \mathbf{\mu \mu}'
\]

The covariance matrix of a random vector contains the variances as diagonal elements and the covariance’s as off diagonal elements. The covariance matrix is often written as

\[
\text{Cov}[\mathbf{X}] = \mathbf{C} = \begin{pmatrix}
\sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\
\sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{p1} & \sigma_{p2} & \cdots & \sigma_p^2
\end{pmatrix}
\]
Chapter 4 Analysing Covariance and Correlation Matrices

Covariance matrices are symmetric because $C_{ov}[X,Y] = C_{ov}[Y,X]$ for any two random variables $X$ and $Y$. Furthermore, if $X = Y$ then $C_{ov}[X,Y] = C_{ov}[X,X] = Var[X]$. Also, when $E[X]$ or $E[Y]$ is zero, the covariance equals $E[XY]$, which has a name on its own, the correlation. Random variables $X$ and $Y$ are uncorrelated if $C_{ov}[X,Y] = 0$.

If $C$ is the covariance matrix of a $p$-variate random vector $X$, then it contains $p(p-1)/2$ distinct off diagonal elements, or covariance’s $\sigma_{ij}$. Each of them can be transformed into correlation coefficients $\rho_{ij}$ by the transformation

$$\rho_{ij} = corr[X_i, X_j] = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} = \frac{\sigma_{ij}}{\sigma_{i}\sigma_{j}}.$$ 

The $p \times p$ matrix with $1$’s on the diagonal and correlation coefficients $\rho_{ij}$ off diagonal is called the correlation matrix of $X$

$$R = \begin{pmatrix}
1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1p} \\
\rho_{21} & 1 & \rho_{23} & \cdots & \rho_{2p} \\
\rho_{31} & \rho_{32} & 1 & \cdots & \rho_{3p} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho_{p1} & \rho_{p2} & \rho_{p3} & \cdots & 1
\end{pmatrix}.$$ 

There are some straightforward matrix manipulations relating the covariance matrix $C$ and the correlation matrix $R$, based on multiplication of matrix by a diagonal matrix. Let $diag(C)$ denote a diagonal matrix with elements $\sigma_1, \ldots, \sigma_p$ on the diagonal, i.e.,

$$diag(C) = diag(\sigma_1, \ldots, \sigma_p) = \begin{pmatrix}
\sigma_1 & 0 & \cdots & 0 \\
0 & \sigma_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma_p
\end{pmatrix}.$$ 

Then the square root of $diag(C)$ can be defined by taking the square root of all diagonal elements

$$\left[diag(C)\right]^{1/2} = diag(\sigma_1, \ldots, \sigma_p) = \begin{pmatrix}
\sqrt{\sigma_1} & 0 & \cdots & 0 \\
0 & \sqrt{\sigma_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sqrt{\sigma_p}
\end{pmatrix}.$$ 

The inverse of $\left[diag(C)\right]^{1/2}$ is

$$\left[diag(C)\right]^{-1/2} = \left[diag(C)\right]^{1/2}^{-1} = \begin{pmatrix}
1/\sqrt{\sigma_1} & 0 & \cdots & 0 \\
0 & 1/\sqrt{\sigma_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1/\sqrt{\sigma_p}
\end{pmatrix}.$$ 

With this notation, it can be verified that

$$R = \left[diag(C)\right]^{-1/2}C\left[diag(C)\right]^{1/2}.$$ 

Conversely, the covariance matrix $C$ can be written in terms of the standard deviation and correlation as

$$C = \left[diag(C)\right]^{1/2}R\left[diag(C)\right]^{1/2}.$$
4.2. Theorems Involving Covariance and Correlation Matrices [12,13,14,15]

**Theorem 4.2-1.**
Let $X$ be $p \times 1$ with mean vector $\mu$ and the covariance matrix $\text{Cov}[X] = C$, and let

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$$

be a vector of constraints. Then, if the following is allowed

$$Y = \mathbf{a}'X = \sum_{i=1}^{p} a_i X_i$$

then $\mu_Y = E[Y] = E[\sum a_i X_i] = \sum a_i \mu_i = \mathbf{a}'\mu$

so

$$\text{Var}[Y] = E[(Y - \mu_Y)^2] = E[(\mathbf{a}'X - \mathbf{a}'\mu)^2]$$

$$= E[(\mathbf{a}'(X - \mu))(\mathbf{a}'(X - \mu))']$$

$$= E[\mathbf{a}'(X - \mu)(X - \mu)']\mathbf{a} = \mathbf{a}'C\mathbf{a}$$

Because $\text{Var}[Y] \geq 0$ also $\mathbf{a}'C\mathbf{a} \geq 0$ for any vector $\mathbf{a}$. This says that any covariance matrix $C$ must be a *positive definite* matrix. If the variances are chosen to be of unit value, then the covariance matrix becomes the correlation matrix with 1’s on the diagonal and the correlation coefficient off diagonal. Therefore, this property holds also for the correlation matrix. In the sequel of this thesis, only correlation matrices are considered.

The linear relation between two random variables is measured by their correlation coefficient. It takes values between $[-1, 1]$, where $-1$ indicates perfect negative correlation and $1$ perfect positive correlation. The $(i, j)$ element of a correlation matrix is the correlation coefficient of the $i$-th and $j$-th random variable. Obviously, $(i, i) = 1$, since a variable is clearly perfectly correlated with itself. For all random variables $i$ and $j$, $(i, j) = (j, i)$ and are less or equal to $1$. Therefore, the valid correlation matrix can be defined as a symmetric matrix with unit diagonal. Furthermore, the valid correlation matrix is also a *positive definite* matrix. It is well known that for a *positive definite* matrix $A \in \mathbb{R}^{p \times p}$

$$|a_{ij}| \leq (a_{ii} + a_{jj})/2 \quad i \neq j$$

and

$$|a_{ij}| \leq \sqrt{a_{ii}a_{jj}}.$$  

Thus for a correlation matrices the following is valid

$$|a_{ij}| \leq 1$$

and the inequality is strict if variables $i$ and $j$ are not perfectly correlated.

---

2 The matrix $A$ is symmetric if $A' = A$.

3 *Appendix B on Positive Definite Matrices* can be consulted on this subject.
Chapter 4 Analysing Covariance and Correlation Matrices

Theorem 4.2-2.
Let $\mathbf{M}$ be a real symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_p$. Then $\mathbf{M}$ is similar to the diagonal matrix $\mathbf{\Lambda}$ given by

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_p \end{bmatrix}$$

under the transformation

$$\mathbf{U}^{-1}\mathbf{M}\mathbf{U} = \mathbf{\Lambda}$$

where $\mathbf{U}$ is a matrix whose columns are the ordered mutually orthogonal unit eigenvectors $\phi_i$, $i = 1, \ldots, p$ of $\mathbf{M}$. Thus

$$\mathbf{U} = (\phi_1, \ldots, \phi_p).$$

Moreover, it can be shown that $\mathbf{U}'\mathbf{U} = \mathbf{I}$ (and that $\mathbf{U}' = \mathbf{U}^{-1}$) so that the following equations can be written

$$\mathbf{U}'\mathbf{M}\mathbf{U} = \mathbf{\Lambda} \quad \text{and} \quad \mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$$

where $\mathbf{\Lambda}$ is a diagonal matrix that consists of the eigenvalues of $\mathbf{M}$ and $\mathbf{U}$ is a matrix whose columns consist of an orthonormal set of eigenvectors of $\mathbf{M}$.

If $\mathbf{\Lambda}^{1/2}$ is defined to be a diagonal matrix whose elements are the square root of the elements of $\mathbf{\Lambda}$ and if the matrix

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}^{1/2}$$

then

$$\mathbf{A}\mathbf{A}' = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}\mathbf{U}' = \mathbf{U}\mathbf{\Lambda}\mathbf{U}' = \mathbf{M}.$$

Theorem 4.2-3.
The correlation matrix is a valid correlation matrix if it is a symmetric positive definite, that is, a real-valued symmetric matrix with nonnegative eigenvalues.

First let $\lambda_i > 0$, $i = 1, \ldots, p$. Then with the linear transformation $\mathbf{x} = \mathbf{U}\mathbf{y}$, for any vector $\mathbf{x}$ it can be written

$$\mathbf{x}'\mathbf{M}\mathbf{x} = (\mathbf{U}\mathbf{y})'\mathbf{M}(\mathbf{U}\mathbf{y})$$

$$= \mathbf{y}'\mathbf{U}'\mathbf{M}\mathbf{U}\mathbf{y}$$

$$= \mathbf{y}'\mathbf{A}\mathbf{y}$$

$$= \sum_{i=1}^{n} \lambda_i y_i^2 > 0$$

unless $\mathbf{y} = \mathbf{0}$. But if $\mathbf{y} = \mathbf{0}$ then from $\mathbf{x} = \mathbf{U}\mathbf{y}$, $\mathbf{x} = \mathbf{0}$ as well. Hence, this shows that $\mathbf{M}$ is positive definite if $\lambda_i > 0$ for all $i$.

---

4 That is, $\phi_i$ goes with $\lambda_i$ for $i = 1, \ldots, p$.

5 A set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ is orthogonal if $\mathbf{x}_i'\mathbf{x}_j = 0$ whenever $i \neq j$ and orthonormal $\mathbf{x}_i'\mathbf{x}_i = 1$. Orthogonal basis is defined as the basis where the independent collection of vectors is orthogonal $\mathbf{x}_i'\mathbf{x}_j = 0$ where $i \neq j$. In fact, given any orthonormal basis, the matrix whose rows are that basis is an orthogonal matrix. It is automatically the case that the columns are another orthonormal basis. A real square matrix $\mathbf{A}$ is called an orthogonal matrix if $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{U}\mathbf{U}' = \mathbf{I}$. Appendix A on Orthogonality can also be consulted on this subject.
Conversely, it must also be shown that if $\mathbf{M}$ is positive definite then all $\lambda_i > 0$.

Thus for any $\mathbf{x} \neq \mathbf{0}$

$$\mathbf{x}^\top \mathbf{M} \mathbf{x} > 0.$$ 

In particular, the previous equation must hold for $\phi_1, \ldots, \phi_p$. But

$$0 < \phi_i^\top \mathbf{M} \phi_i = \lambda_i, \quad i = 1, \ldots, p.$$ 

Hence, $\lambda_i > 0, i = 1, \ldots, p$. Thus, a positive definite matrix will have all positive eigenvalues.

**Theorem 4.2-4.**

Gershgorin’s theorem states that the eigenvalues of $\mathbf{A} \in \mathbb{C}^{\times n}$ lie in the union of $p$ disks in the complex plane. The $i$-th disk is given by

$$D_i = \left\{ z \in \mathbb{C} : |z - a_i| \leq \sum_{j=1, j \neq i}^p |a_j| \right\}, \quad i = 1, \ldots, p.$$ 

Since the correlation matrix is symmetric all its eigenvalues are real so we have for an eigenvalue $\lambda_i$,

$$|\lambda_i - 1| \leq p - 1$$

But also $\mathbf{A}$ is positive definite, so its eigenvalues are nonnegative and

$$0 < \lambda_i \leq p, \quad i = 1, \ldots, p.$$ 

Furthermore, since $\text{trace}(\mathbf{A}) = \sum_i \lambda_i$

$$\sum_i \lambda_i = p.$$  

**Theorem 4.2-5.**

If $\mathbf{A} \in \mathbb{C}^{p \times p}$ is symmetric and positive definite, then there exists a lower triangular (non-singular) matrix $\mathbf{B} \in \mathbb{C}^{p \times p}$ with positive diagonal entries such that $\mathbf{A} = \mathbf{B} \mathbf{B}'$. The decomposition $\mathbf{A} = \mathbf{B} \mathbf{B}'$ is known as the Cholesky Decomposition and $\mathbf{B}$ is referred as the Cholesky triangle.

**Theorem 4.2-6.**

If an arbitrary matrix $\mathbf{B}$ has full rank, then the matrix $\mathbf{A} = \mathbf{B} \mathbf{B}'$ is always positive definite.

**Theorem 4.2-7.**

If an arbitrary matrix $\mathbf{B}$ has independent columns, then the matrix $\mathbf{A} = \mathbf{B} \mathbf{B}'$ is always positive definite. To determine if $\mathbf{B}$ has independent columns, the rank of matrix $\mathbf{B}$ could be calculated. If this is the case, the matrix $\mathbf{B}$ has the full rank.

---

6 A square matrix $\mathbf{A}$ is said to be singular if determinant of $\mathbf{A}$ is zero. If determinant of $\mathbf{A}$ is nonzero, it is said to be nonsingular (one that has a matrix inverse).

7 Appendix C on the Cholesky Decomposition can be consulted on this subject.

8 The rank of a matrix $\mathbf{A}$, denoted as $r(\mathbf{A})$, is defined as the maximum number of linearly independent rows (or columns) of $\mathbf{A}$. In other words, the rank of a matrix is the order of the largest square sub-matrix of $\mathbf{A}$ whose determinant is not zero. The matrix $\mathbf{A}$ is of full rank if its rank equals its order, meaning that a square $p \times p$ matrix $\mathbf{A}$ is of full rank if rank $r(\mathbf{A}) = p$. 
Chapter 4 Analyzing Covariance and Correlation Matrices

4.3. Positive Definiteness Limits Covariance and Correlation Matrices

Covariance matrix \( \text{Cov}[\mathbf{X}] = \mathbf{C} \) of \( p \)-variate random vector is having the variances on the diagonal and the covariance’s off diagonal
\[
\mathbf{C} = \begin{cases} 
\sigma_i^2 & i = j \\
\sigma_{ij} & i \neq j 
\end{cases}.
\]
If the variances are chosen to be of unit value and the covariance’s are chosen to be the same for all \( p \) random variables, then the covariance matrix becomes the correlation matrix with 1’s on the diagonal and the same correlation coefficient, \( \rho_{ij} \), off diagonal.
\[
\mathbf{R} = \begin{cases} 
1 & i = j \\
\rho_{ij} & i \neq j 
\end{cases}.
\]
The set of \( p \) random variables is said to possess *intraclass correlation*, which limits the correlation coefficient range. The *intraclass correlation coefficient* ranges
\[
-1 < \rho_{\text{ICC}} < 1,
\]
where \( n \) is the number of *equally correlated variables*. In the above-presented structure \( p \) random variables are having the same variance and the covariance. Therefore, the number of equally correlated variables is \( n = p \). Thus, if \( p \) variables are equally correlated, the largest possible magnitude of this correlation is
\[
-1 < \rho < 1.
\]
In this manner the correlation coefficient is constrained by the intraclass correlation coefficient. The boundaries on the correlation coefficient are required if the correlation matrix has to be a positive definite matrix.

4.4. Solving Positive Definiteness with Nearest Correlation Matrix Problem [7]

The most important property of the correlation matrix is its positive definiteness. In the previous sections, boundaries on the correlation matrix to be positive definite are found. On contrary, in this section the numerical method for finding the nearest correlation matrix, which is always positive definite, will be presented. *Appendix E* includes the MATLAB M-programming code, while *Appendix F* includes C-programming code for finding nearest correlation matrix. The nearest correlation matrix problem is the problem of finding the nearest matrix with the same properties as the given correlation matrix. Where the given correlation matrix fails to satisfy the properties of the valid correlation matrix, the nearest correlation matrix will be used to simulate the QoS complexity.
Chapter 4 Analysing Covariance and Correlation Matrices

The nearest correlation problem consist of finding a nearest member of some given class of matrices for an arbitrary symmetric matrix \( A \in \mathbb{R}^{n \times n} \) or to compute the distance
\[
\gamma(A) = \min \left\{ \| A - X \|_F : X \text{ is a correlation matrix} \right\}
\]
and a matrix achieving this minimum distance by applying the Alternating Projection Programming (APP) Algorithm. The norm is a weighted version of the Frobenius norm
\[
\| A \|_F = \sum_{i,j} a_{ij}^2
\]
and is the easiest norm to work with for this problem. Defining
\[
\langle A, B \rangle = \text{trace}(A^T B),
\]
which is an inner product that induces the \( F - \text{norm} \), there are two different weighted Frobenius norms. The first and the most commonly used in numerical mathematics, is
\[
\| A \|_W = \left\| W^{1/2} A W^{1/2} \right\|_F,
\]
where \( W \) is a symmetric positive definite matrix. The second weighted norm is
\[
\| A \|_H = \left\| H \circ A \right\|_F,
\]
where \( H \) is a symmetric matrix of positive weights and \( \circ \) denotes the Hadamard product
\[
A \circ B = (a_{ij} b_{ij}).
\]
Further two following sets are defined
\[
S = \left\{ Y = Y' \in \mathbb{R}^{p \times p} : Y > 0 \right\},
\]
\[
U = \left\{ Y = Y' \in \mathbb{R}^{p \times p} : y_{ii} = 1, i = 1 : p \right\},
\]
where \( Y > 0 \) means positive definite. The problem searches for a matrix in the intersection of \( S \) and \( U \) that is closest to \( A \). Since \( S \) and \( U \) are closest convex sets, so it is their intersection. It thus follows from standard results in approximation theory that the minimum in \( \gamma(A) \) is achieved and that it is achieved at a unique matrix \( X \).

**Theorem 4.4-1.**
The correlation matrix \( X \) solves \( \gamma(A) \) if and only if
\[
X = A + W^{-1} (VDV' + \text{diag}(\theta)) W^{-1},
\]
where \( V \in \mathbb{R}^{p \times p} \) has orthonormal columns spanning \( \text{null}(X) \), \( D = \text{diag}(d) > 0 \), and the \( \theta_i \) are arbitrary. An analogue of Theorem 4.5-1 holds for \( H - \text{norm} \), with
\[
X = A + (VDV' + \text{diag}(\theta_i)) \circ (h^2).
\]

In sequel, the Alternative Projection Programming (APP) will be investigated. The idea of the APP is to project \( A \) on the sets \( S \) and \( U \) alternatively, which is repeated until the projection converge
\[
A \leftarrow P_U (P_S (A)),
\]
where \( P_U \) is the projection onto \( U \) and \( P_S \) the projection onto \( S \).
Chapter 4 Analysing Covariance and Correlation Matrices

**Theorem 4.4-2. The projection of \( A \) onto \( U \)**
For the \( W - \)norm
\[
P_U(A) = A - W^{-1} \text{diag}(\theta) W^{-1},
\]
where \( \theta = [\theta_1, \ldots, \theta_n]^T \) is the solution of the linear system
\[
(W^{-1} \circ W^{-1}) \theta = \text{diag}(A - I).
\]
In the case where \( W \) is diagonal the following holds
\[
P_U(A) = (p_{ij}), \quad \begin{cases} 
  a_{ij}, & i \neq j \\
  1, & i = j
\end{cases}
\]
For the \( H - \)norm, the above-equation is the projection onto \( U \) for all \( H \).

**Theorem 4.4-3. The projection of \( A \) onto \( S \)**
For a symmetric \( A \in \mathbb{R}^{n \times n} \) with spectral decomposition \( A = QDQ' \), where \( D = \text{diag}(\lambda_i) \) and \( Q \) is orthogonal, let
\[
A_\lambda = Q \text{diag}(\max(\lambda_i, 0)) Q', \quad A_\alpha = Q \text{diag}(\min(\lambda_i, 0)) Q'.
\]
For the \( W - \)norm
\[
P_S(A) = W^{-1/2} ((W^{1/2} A W^{1/2})_+) W^{-1/2}.
\]
Moreover,
\[
\text{diag}(P_S(A)) \geq \text{diag}(A).
\]
For the \( H - \)norm, no closed formula is known.

**Algorithm 4.4-1. Alternative Projection Programming Algorithm**
Given a symmetric matrix \( A \in \mathbb{R}^{n \times n} \), this algorithm computes the nearest correlation matrix to \( A \) in the \( W - \)norm.

\[
S_0 = 0 \\
Y_0 = A
\]

for \( k = 1, 2, \ldots \)
\[
R_k = Y_{k-1} - S_{k-1} \\
X_k = P_S(R_k) \\
S_k = X_k - R_k \\
Y_k = P_U(X_k)
\]
end

Figure 4.5-1 Meta-code of Alternative Projection Programming Algorithm.
4.5. Solving Positive Definiteness with Eigenvalues and Eigenvectors

In the previous sections the method for finding the nearest correlation matrix with the same properties as the given correlation matrix is deduced. This section presents a different method for finding the valid correlation matrix. The idea is to find the valid correlation matrix with the basis consisting of the random eigenvalues and its random eigenvectors. To construct the valid correlation matrix the following steps are performed:

- A $p \times p$ diagonal matrix $\Lambda$ is constructed whose $p$ diagonal elements are random and positive eigenvalues $\lambda_1 \ldots \lambda_p$.
- A $p \times p$ random matrix $U$ is constructed. Run the matrix $U$ through Singular Value Decomposition (SVD). The constructed matrix $U$ is a $p \times p$ matrix whose columns are ordered mutually orthogonal unit eigenvectors.
- A $p \times p$ random matrix $M$ is the product of a $p \times p$ column-orthogonal matrix $U$, $p \times p$ diagonal matrix $\Lambda$ and the transpose of a column-orthogonal matrix $U$. Section 4.2 can also be consulted on this subject.
- The constructed random matrix $M$ is a $p \times p$ positive definite correlation matrix.
- Where the generated correlation matrix fails to satisfy the given properties of the valid correlation matrix, the correlation matrix created from the random eigenvalues and its random eigenvectors will be used to simulate the QoS complexity.
5. Generating Correlated Vector Random Variables

In this chapter methods for generating vectors of random variables with specified covariance matrices will be presented. Also, methods for generating vectors of jointly Gaussian random variables will be discussed.


Gaussian random variables are very important because they show up in nearly every area of science and engineering. In this section the case of \( N \) random variables will be introduced. A random variable is called Gaussian if its density function has the form

\[
f_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-(x-a_X)^2/2\sigma_X^2}
\]

where the \( \sigma_X > 0 \) and \( -\infty < a_X < \infty \) are the real constants.

\( N \) random variables \( X_1, X_2, \ldots, X_N \) are called jointly Gaussian if their joint density function can be written as

\[
f_{X_1,\ldots,X_N}(x_1,\ldots,x_N) = \frac{1}{(2\pi)^{N/2}} \exp \left\{ -\frac{1}{2} \left[ \mathbf{x} - \mathbf{E}[\mathbf{X}] \right] \mathbf{C}^{-1} \left[ \mathbf{x} - \mathbf{E}[\mathbf{X}] \right]^T \right\}
\]

where the following matrices are defined

\[
\mathbf{x} - \mathbf{E}[\mathbf{X}] = \begin{bmatrix}
x_1 - \mathbf{E}[X_1] \\
x_2 - \mathbf{E}[X_2] \\
\vdots \\
x_N - \mathbf{E}[X_N]
\end{bmatrix}
\]

\[
\mathbf{C} = \begin{bmatrix}
C_{11} & C_{12} & \cdots & C_{1N} \\
C_{21} & C_{22} & \cdots & C_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
C_{N1} & C_{N2} & \cdots & C_{NN}
\end{bmatrix}
\]

The notation \( \mathbf{E}[\mathbf{X}] \) is used for the expected value of \( \mathbf{X} \), \( \mathbf{X}^T \) is used for matrix transpose, \( \mathbf{C}^{-1} \) for the matrix inverse and \( |\mathbf{C}| \) for the determinant. The joint density function is often called the \( N \)-variate Gaussian density function.

Elements of the covariance matrix \( \mathbf{C} \) of \( N \) random variables, are given by

\[
C_{ij} = \mathbf{E}\left[(X_i - \mathbf{E}[X_i])(X_j - \mathbf{E}[X_j])\right] = \begin{cases}
\sigma_{X_i}^2 & i = j \\
\sigma_{X_iX_j} & i \neq j
\end{cases}
\]

where \( \sigma_{X_i}^2 \) ( \( \sigma_{X_j}^2 \) ) is used for the notation of the variance of \( X_i \) ( \( X_j \) ) and \( \sigma_{X_iX_j} \) is the covariance of \( X_i \) and \( X_j \).
Some properties of Gaussian random variables are:

- Gaussian random variables are completely defined through only their first- and second-
  order moments; that is, by their means, variances and covariance’s. This fact is readily
  apparent since only these quantities are needed to completely determine joint density
  function of \( N \) jointly Gaussian random variables.
- If the random variables are uncorrelated, they are also statistically independent.
- Random variables produced by a linear transformation of \( N \) random variables
  \( X_1, X_2, \ldots, X_N \) will also be Gaussian.
- Any \( k \)-variate marginal density function obtained from the \( N \)-variate density function by
  integrating out \( N-k \) random variables will be Gaussian. If the variables are ordered so
  that \( X_1, \ldots, X_k \) occur in the marginal density and \( X_{k+1}, \ldots, X_N \) are integrated out, then the
  covariance matrix of \( X_1, \ldots, X_k \) is equal to the leading \( k \times k \) sub-matrix of the covariance
  matrix of \( X_1, \ldots, X_N \).
- The conditional density \( f_{X_{k+1}, \ldots, X_N} (x_{k+1}, \ldots, x_N \mid X_1 = x_1, \ldots, X_N = x_N) \) is Gaussian. This hold
  for any \( k < N \).


Let \( \mathbf{X} = (X_1, \ldots, X_N) \) be a vector of \( N \) zero-mean, unit-variance, and uncorrelated random
variables. We are interested in generating a zero-mean vector \( \mathbf{Y} = (Y_1, Y_2, \ldots, Y_N) \) with some
specified covariance matrix \( \mathbf{C} \). \( \mathbf{C} \) is a valid covariance matrix if it is a positive definite\(^9\)
symmetric matrix that is real-valued, symmetric matrix with nonnegative eigenvalues. It can
be accomplished by letting \( \mathbf{Y} = \mathbf{AX} \), where \( \mathbf{A} \) is a \( N \times N \) matrix. The mapping \( \mathbf{Y} = \mathbf{AX} \) can also be used to transform a vector \( \mathbf{X} \) with covariance \( \mathbf{C}_X \) into a vector \( \mathbf{Y} \) with a diagonal
covariance matrix, that is, into a set of orthogonal random variables.

Let \( a_{kj} \) be the element in the \( k \)-th row and \( j \)-th column of \( \mathbf{A} \). The \( k \)-th element of \( \mathbf{Y} \) is then
\[
Y_k = \sum_{j=1}^{n} a_{kj} X_j.
\]
Clearly, the mean of \( Y_k \) is zero since the means of the \( X_j \)’s are zero. The covariance between
the elements of \( \mathbf{Y} \) is then given by
\[
E[Y_k Y_L] = E \left[ \sum_{j=1}^{n} a_{kj} X_j \sum_{j=1}^{n} a_{Lj} X_j \right] = \sum_{j=1}^{n} \sum_{j=1}^{n} a_{kj} a_{Lj} E[X_j X_j].
\]
Since the \( X_j \)’s are uncorrelated and have zero mean and unit variance, we have that
\[
E[X_j X_j] = 1 \quad \text{if} \quad j = j' \quad \text{and equals zero otherwise. Therefore all terms in the above double}
\quad \text{summation are zero except when} \quad j = j', \quad \text{and}
\]
\[
E[Y_k Y_L] = \sum_{j=1}^{n} a_{kj} a_{Lj}.
\]
The above equation shows that the \( k, k' \) element of the covariance matrix of \( \mathbf{Y} \) is equal to the
dot product between the \( k \)-th row of the matrix \( \mathbf{A} \) and the \( k' \)-th column of the matrix \( \mathbf{A}' \),

\(^9\) Appendix B on Positive Definite Matrices can be scrutinized for the additional information on this subject.
Chapter 5 Generating Correlated Vector Random Variables

the transpose of the matrix \( \mathbf{A} \). In other words, if \( \mathbf{Y} = \mathbf{A} \mathbf{X} \), where \( \mathbf{X} \) consists of unit-variance, uncorrelated random variables, then the covariance matrix of \( \mathbf{Y} \) is \( \mathbf{C} = \mathbf{A} \mathbf{A}' \).

Next, a method of finding a matrix \( \mathbf{A} \), that satisfies the above equation for a given \( \mathbf{C} \), is presented. This problem is solved using elementary methods from linear algebra. Since \( \mathbf{C} \) is a symmetric matrix, it can be expressed in the form
\[
\mathbf{C} = \mathbf{P} \Lambda \mathbf{P}'
\]
where \( \mathbf{A} \) is a diagonal matrix that consists of the eigenvalues of \( \mathbf{C} \), and \( \mathbf{P} \) is a matrix whose columns consist of an orthonormal set of eigenvectors of \( \mathbf{C} \). If \( \Lambda^{1/2} \) is defined to be a diagonal matrix whose elements are the square root of the elements of \( \Lambda \) and if \( \mathbf{A} = \mathbf{P} \Lambda^{1/2} \) is permitted, then
\[
\mathbf{A} \Lambda' = \mathbf{P} \Lambda^{1/2} \Lambda^{1/2} \mathbf{P}' = \mathbf{P} \Lambda \mathbf{P}' = \mathbf{C}.
\]
Thus, \( \mathbf{A} \) yields \( \mathbf{Y} \) with the desired covariance matrix.

Furthermore, if \( \mathbf{X} \) is defined to be a zero-mean vector, with covariance \( \mathbf{C}_X \), and if a vector \( \mathbf{Y} = \mathbf{A} \mathbf{X} \) of orthogonal random variables, that is \( E[Y_i Y_j] = 0 \) if \( i \neq j \), wants to be obtained, the following is applicable. The equation, which gives the \( (k,k') \) entry of the covariance matrix \( \mathbf{C}_Y \), implies, rewritten in the matrix form, that
\[
\mathbf{C}_Y = \mathbf{A} \mathbf{C}_X \mathbf{A}'
\]
where \( \mathbf{C}_X \) is the covariance matrix of \( \mathbf{X} \) with entries \( E[X_i X_j] \). Pre-multiplying the equation \( \mathbf{C} = \mathbf{P} \Lambda \mathbf{P}' \) by \( \mathbf{P}' \) and post-multiplying it by \( \mathbf{P} \), the equation \( \mathbf{A} = \mathbf{P} \mathbf{C}_Y \mathbf{P} \) could be obtained. By comparing the above two equations the following can be concluded: if \( \mathbf{A} = \mathbf{P}' \) and \( \mathbf{K}_Y = \mathbf{A} \) are allowed, the random variables in \( \mathbf{Y} \) are orthogonal\(^{10} \) as required.


If \( \mathbf{X} \) is a vector of jointly Gaussian random variables with covariance \( \mathbf{C}_X \), then \( \mathbf{Y} = \mathbf{A} \mathbf{X} \) is also jointly Gaussian with covariance matrix \( \mathbf{C}_Y = \mathbf{A} \mathbf{C}_X \mathbf{A}' \). If we assume that \( \mathbf{X} \) consists of zero-mean, unit-variance, uncorrelated random variables, then \( \mathbf{C}_X = \mathbf{I} \) is the identity matrix and therefore \( \mathbf{C}_Y = \mathbf{A} \mathbf{A}' \). Thus, jointly Gaussian vector \( \mathbf{Y} \) with an arbitrary covariance matrix \( \mathbf{C}_Y \) could be generated as follows:

- Find a matrix \( \mathbf{A} \) for any desired covariance matrix \( \mathbf{C}_Y \), such that \( \mathbf{C}_Y = \mathbf{A} \mathbf{A}' \). The Cholesky Decomposition can be used to generate \( \mathbf{A} \mathbf{A}' \) efficiently. For a detailed description on the Cholesky method, Appendix C on the Cholesky decomposition can be consulted.
- Generate \( \mathbf{X} \) consisting of \( N \) zero-mean, unit-variance and uncorrelated Gaussian random variables.
- Let \( \mathbf{Y} = \mathbf{A} \mathbf{X} \).
- Linear transformation of \( N \) Gaussian random variables produces \( N \) Gaussian random variables with mean values
  \[
  E[Y_i] = \sum_{j=1}^{N} a_{ij} E[X_j], \quad i = 1, \ldots, N.
  \]

\(^{10}\) Appendix A on Orthogonality can be scrutinized for additional information on this subject.
Chapter 5 Generating Correlated Vector Random Variables

The following example shows how a vector $\mathbf{Y}$ of two jointly Gaussian random variables, with an arbitrary $2 \times 2$ covariance matrix $\mathbf{C}_y$, could be generated:

- $\mathbf{X}$ is a vector of two jointly Gaussian random variables, which are zero-mean, unit-variance and uncorrelated random variables. $\mathbf{C}_x$ is then the identity matrix.
- $\mathbf{Y}$ is also a jointly Gaussian vector, with $2 \times 2$ covariance matrix, $\mathbf{C}_y = \mathbf{A}\mathbf{A}'$. By applying Cholesky Decomposition, matrix $\mathbf{A}$ can be determined, easily.

$$
\mathbf{C}_y = \begin{bmatrix}
\sigma_{y_1}^2 & \rho_{y_1y_2}\sigma_{y_1}\sigma_{y_2} \\
\rho_{y_1y_2}\sigma_{y_1}\sigma_{y_2} & \sigma_{y_2}^2
\end{bmatrix}
$$

$$
\mathbf{A} = \begin{bmatrix}
\sigma_{y_1} & 0 \\
\rho_{y_1y_2}\sigma_{y_2} & \sigma_{y_2}\sqrt{1-\rho_{y_2}^2}
\end{bmatrix}
$$

- Now, $\mathbf{Y} = \mathbf{A}\mathbf{X}$ can be generated. The expression for a jointly Gaussian vector $\mathbf{Y}$ with an arbitrary covariance matrix $\mathbf{C}_y$ is

$$
Y_1 = \sigma_{y_1}X_1
$$

$$
Y_2 = \rho_{y_1y_2}\sigma_{y_2}X_1 + \left(\sigma_{y_2}\sqrt{1-\rho_{y_2}^2}\right)X_2.
$$

- If $\mathbf{Y}$ vector consists of two jointly Gaussian random variables, which are zero-mean, unit-variance and correlated random variables, then its covariance matrix becomes the correlation matrix with 1’s on the diagonal and the correlation coefficient, $\rho_{y_i}$, off diagonal. The above-equations will change as follows

$$
Y_1 = X_1
$$

$$
Y_2 = \rho_{y_1}X_1 + \left(\sqrt{1-\rho_{y_1}^2}\right)X_2
$$

- Gaussian random variables can become negative what is undesirable for the problems considered in this thesis. To prevent this, the mean values are added to variables of jointly Gaussian vector, $\mathbf{Y}$. This is also the case for the vector of $N$ jointly Gaussian random variables.

$$
Y_1 = \sigma_{y_1}X_1 + \mu_1
$$

$$
Y_2 = \rho_{y_1y_2}\sigma_{y_2}X_1 + \left(\sigma_{y_2}\sqrt{1-\rho_{y_2}^2}\right)X_2 + \mu_2
$$
Chapter 6 Configuring Network Topology for Simulations

6. Configuring Network Topology for Simulations

In this chapter the configuration of a two-dimensional lattices network topology will be discussed. The configuration of a network topology is necessary for the simulations that will appear subsequently in this thesis. This includes:

- Overview of the proposed path correlation classes. The path correlation classes in a two-dimensional lattice network topology are generated in two different ways. The most important classes of path correlation are those where the correlation in the network topology is generated via the correlation matrix. The other one is the path correlation class, where the correlation in the network topology is generated via underlying chain topologies, creating non-dominated and simultaneously correlated paths.
- Overview of the simulation input and output parameters.
- Overview of two different network topology configurations, where the path and the link correlation vectors are either consecutively or randomly configured.

6.1. Proposed Path Correlation Classes

Finding a path correlation in a two-dimensional lattice network topology, which may lead to NP-complete behaviour of Multi-Constrained QoS routing, is the most important step in successfully accomplishing the goal of the thesis. Inducing the complexity analysis presented in Chapter 3, the analysis of correlation matrices presented in Chapter 4 and the generation of correlated vector random variables presented in Chapter 5, different path correlation classes are generated. In the first subsection, the classes of path correlation will be presented where the correlation is generated via the correlation matrix. For these classes, the analysis presented in Chapter 4 and 5 is crucial. Subsequently, the classes of path correlation will be presented, where the correlation is generated via underlying chain topology. For this class, the analysis presented in Chapter 3 is fundamental.

6.1.1 Generating Path Correlation Classes via Correlation Matrix

The positive definiteness property limits the correlation coefficient range of the following correlation matrix structure

\[
R = \begin{cases} 
1 & i = j \\
\rho_{ij} & i \neq j 
\end{cases}
\]

where the correlation coefficient has the same value for all \( i \neq j \). The range of the correlation coefficient in such correlation matrix structures is

\[
\frac{-1}{p-1} \leq \rho \leq 1,
\]

meaning that for negative correlation coefficient range the correlation matrix is not solvable with the Cholesky decomposition. Therefore, the following structure of the correlation matrix is proposed.
Class 1 of Path Correlation

The correlation matrix is the \( p \times p \) Toeplitz\(^{11} \) matrix with 1’s on the main diagonal and the correlation coefficient, \( \rho_{ij} \), off diagonal if \( \rho_{ij} \geq 0 \). If \( \rho_{ij} < 0 \), then a diagonal filled with the negative correlation coefficient, \( -\rho_{ij} \), follows a diagonal that is filled with the positive correlation coefficient, \( \rho_{ij} \). The inspected range of the correlation coefficient is \([-1,1]\). Appendix G includes C-programming code on path correlation of the Class 1, while Appendix D includes C-programming code on the Cholesky decomposition algorithm.

\[
\begin{bmatrix}
1 & |\rho_{12}| & \rho_{13} & \cdots & |\rho_{1p}|

|\rho_{21}| & 1 & |\rho_{23}| & \cdots & |\rho_{2p}|

|\rho_{31}| & |\rho_{32}| & 1 & \cdots & |\rho_{3p}|

\vdots & \vdots & \vdots & \ddots & \vdots

|\rho_{p1}| & |\rho_{p2}| & |\rho_{p3}| & \cdots & 1
\end{bmatrix}
\]

\( \rho_{ij} = \rho \), \( i, j = 1, \ldots, p \)

The Class 2 of path correlation includes the correlation matrix, which is a counterpart of the Class 1 correlation matrix structure.

Class 2 of Path Correlation

The correlation matrix is the \( p \times p \) matrix with 1’s on the diagonal and the same value of the correlation coefficient, \( \rho_{ij} \), off diagonal if \( \rho_{ij} \geq 0 \). However, if \( \rho_{ij} < 0 \), this path correlation structure is opposite to the Class 1 structure. The inspected range of the correlation coefficient is \([-1,1]\). Appendix G includes C-programming code on path correlation of the Class 2, while Appendix D includes C-programming code on the Cholesky decomposition algorithm.

\[
\begin{bmatrix}
1 & \rho_{12} & |\rho_{13}| & \rho_{14} & \cdots & |\rho_{1p}|

|\rho_{21}| & 1 & \rho_{23} & |\rho_{24}| & \cdots & |\rho_{2p}|

|\rho_{31}| & |\rho_{32}| & 1 & \rho_{34} & \cdots & |\rho_{3p}|

\vdots & \vdots & \vdots & \vdots & \ddots & \vdots

|\rho_{p1}| & |\rho_{p2}| & |\rho_{p3}| & \cdots & 1
\end{bmatrix}
\]

\( \rho_{ij} = \rho \), \( i, j = 1, \ldots, p \)

The simulation results, which were performed on the Class 1 and Class 2 and will appear later in the thesis, show that an increase of number of nodes does not result in NP-complete behaviour of Multi-Constrained QoS routing. Therefore, the random chosen values for the correlation coefficient are used subsequently to construct the correlation matrix that is completely opposite to the matrix presented in Class 1.

---

\(^{11}\) Matrices whose entries are constant along each diagonal are called Toeplitz matrices. Formally, matrix \( T \in \mathbb{R}^{p \times p} \) is Toeplitz if there exist scalars \( r_{p+1}, \ldots, r_{p-1} \) such that \( a_{ij} = r_{i-j} \) for all \( i \) and \( j \).
**Chapter 6 Configuring Network Topology for Simulations**

**Class 3 of Path Correlation**

The correlation matrix is the \( p \times p \) matrix with 1’s on the diagonal and randomly chosen values of the correlation coefficient, \( \rho_{\text{rand}} \), off diagonal. Thus, the correlation matrix has \( \sum_{i=1}^{p} i - 1 \) random values of the correlation coefficient, \( \rho_{\text{rand}} \), to \( \rho_{\text{rand}} \). The inspected range of the correlation coefficient is \([-1,1]\).

\[
\text{Class 3} = \begin{pmatrix}
1 & \rho_{\text{rand}} & \rho_{\text{rand}} & \cdots & \rho_{\text{rand}} \\
\rho_{\text{rand}} & 1 & \rho_{\text{rand}} & \cdots & \rho_{\text{rand}} \\
\rho_{\text{rand}} & \rho_{\text{rand}} & 1 & \cdots & \rho_{\text{rand}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho_{\text{rand}} & \rho_{\text{rand}} & \rho_{\text{rand}} & \cdots & 1
\end{pmatrix}
\]

The Cholesky decomposition of the completely randomised correlation matrix may not be possible because the random correlation matrix is hardly ever a positive definite matrix. For that reason, this structure of the correlation matrix is transformed into the nearest correlation matrix structure where the Cholesky decomposition is surely achievable. Appendix F includes C-programming code on the Nearest Correlation Matrix Problem, while Appendix D includes C-programming code on the Cholesky decomposition algorithm. Appendix G includes C-programming code on path correlation of the Class 3.

The correlation matrices that appear subsequently in the Class 4 and the Class 5 are an alternative for the random correlation matrix presented in the Class 3.

**Class 4 of Path Correlation**

The correlation matrix is the \( p \times p \) matrix with the main diagonal, which is followed with \( p - 1 \) diagonals. The main diagonal is filled with 1’s, while other \( p - 1 \) diagonals are filled with the same but random value of the correlation coefficient, \( \rho_{\text{rand}} \). Thus, the correlation matrix has \( p - 1 \) different values of the correlation coefficient, \( \rho_{\text{rand}} \), to \( \rho_{\text{rand}} \). The inspected range of the correlation coefficient is \([-1,1]\).

\[
\text{Class 4} = \begin{pmatrix}
1 & \rho_{\text{rand}} & \rho_{\text{rand}} & \cdots & \rho_{\text{rand}} \\
\rho_{\text{rand}} & 1 & \rho_{\text{rand}} & \cdots & \rho_{\text{rand}} \\
\rho_{\text{rand}} & \rho_{\text{rand}} & 1 & \cdots & \rho_{\text{rand}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho_{\text{rand}} & \rho_{\text{rand}} & \rho_{\text{rand}} & \cdots & 1
\end{pmatrix}
\]

This structure of the correlation matrix is transformed into the nearest correlation matrix structure for the reason that the Cholesky decomposition of this randomised matrix may also not be possible. Appendix F includes C-programming code on the nearest correlation matrix problem, while Appendix D includes C-programming code on the Cholesky decomposition algorithm. Appendix G includes C-programming code on path correlation of the Class 4.

**Class 5 of Path Correlation**
Chapter 6 Configuring Network Topology for Simulations

The correlation matrix is the \( p \times p \) matrix with 1’s on the main diagonal, which is followed by \( p-1 \) diagonals. Every other diagonal is filled with the same random value of the correlation coefficient, \( \rho_{\text{rand}} \). In between are diagonals with the same value of the correlation coefficient as in the previous diagonal but with the sign that is alternated. Thus, the correlation matrix has \( p/2 \) different values of the correlation coefficient, \( \rho_{\text{rand}_1} \) to \( \rho_{\text{rand}_{p/2}} \).

The inspected range of the correlation coefficient is \([-1,1]\).

\[
\text{Class 5} = \begin{pmatrix}
1 & \rho_{\text{rand}_1} & -\rho_{\text{rand}_1} & \rho_{\text{rand}_2} & \cdots & -\rho_{\text{rand}_{p/2}} \\
\rho_{\text{rand}_1} & 1 & \rho_{\text{rand}_1} & -\rho_{\text{rand}_1} & \cdots & -\rho_{\text{rand}_{p/2}} \\
-\rho_{\text{rand}_1} & \rho_{\text{rand}_1} & 1 & \rho_{\text{rand}_1} & \cdots & -\rho_{\text{rand}_{p/2}} \\
\rho_{\text{rand}_2} & -\rho_{\text{rand}_2} & \rho_{\text{rand}_2} & 1 & \cdots & -\rho_{\text{rand}_{p/2}} \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\rho_{\text{rand}_{p/2}} & -\rho_{\text{rand}_{p/2}} & \rho_{\text{rand}_{p/2}} & -\rho_{\text{rand}_{p/2}} & \cdots & 1
\end{pmatrix}
\]

This structure of the correlation matrix is also transformed into the nearest correlation matrix structure where the Cholesky decomposition is surely achievable. Appendix F includes C-programming code on the nearest correlation matrix problem, while Appendix D includes C-programming code on the Cholesky decomposition algorithm. Appendix G includes C-programming code on path correlation of the Class 5.

The simulation results, which were performed on the Class 3, Class 4 and Class 5 and will appear later in the thesis, show that an increase of number of nodes does not result in NP-complete behaviour of Multi-Constrained QoS routing. Above all, the calculation of the nearest correlation matrix problem is extremely time-consuming. Therefore, the Class 6 of path correlation is purposed. The idea behind the correlation matrix in the Class 6 is to create the structure that simultaneously will give a positive definite matrix. The method is described in Section 4.5.

Class 6 of Path Correlation

The correlation matrix is the \( p \times p \) matrix with 1’s on the diagonal followed by randomly chosen value of the correlation coefficient, \( \rho_{\text{rand}} \), off diagonal. Thus, the correlation matrix has \( \sum_{i=1}^{p} \) random values of the correlation coefficient, \( \rho_{\text{rand}_1} \) to \( \rho_{\text{rand}_{p/2}} \). The inspected range of the correlation coefficient is \([-1,1]\). Multiplying the following matrices, abovementioned correlation matrix is generated:

- Matrix with \( p \) random eigenvalues, \( \lambda_{\text{rand}_1} \) to \( \lambda_{\text{rand}_p} \), on the main diagonal, ordered randomly in the range \((0,1)\).
- Matrix whose columns are the ordered mutually orthogonal unit eigenvectors. That is \( \Phi_{\text{rand}_i} \) goes with \( \lambda_{\text{rand}_i} \) for \( i=1,\ldots,p \). Running a random correlation matrix through the SVD algorithm generates the matrix with the ordered mutually orthogonal unit eigenvectors.
- Transpose of the matrix whose columns are the ordered mutually orthogonal unit eigenvectors.
The Cholesky decomposition of this randomised correlation matrix is surely achievable because this matrix is a positive definite matrix, where the eigenvalues are always nonnegative. Appendix D includes C-programming code on the Cholesky decomposition algorithm. Appendix G includes C-programming code on path correlation of the Class 6. It also includes C-programming code on the SVD algorithm.

6.1.2 Generating Path Correlation Classes via Underlying Chain Topologies

Class 7 of Path Correlation

In this class, path correlation in the network topology is generated via underlying chain topologies. In detail, a two-dimensional lattice network topology has several chain topologies enclosed in its structure. The paths from a source node to a destination node in those underlying chain topologies are created to be non-dominated. Satisfying the Property 3.3-1 the link weights between succeeding nodes are correlated with each other. For a detailed description on this particular path correlation class, Section 3.3 can be consulted. Recall that the correlation coefficients, ρ and ρlink, do not have any influence on the simulation results. Appendix G includes C-programming code on path correlation of the Class 7.

\[
\begin{pmatrix}
\lambda_{\text{rand}_1} & 0 & \cdots & 0 \\
0 & \lambda_{\text{rand}_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_{\text{rand}_p}
\end{pmatrix}
\times
\begin{pmatrix}
\varphi_{\text{rand}_1} \\
\varphi_{\text{rand}_2} \\
\vdots \\
\varphi_{\text{rand}_p}
\end{pmatrix}
\]

Figure 6.1.2-1 Example of path correlation of the class 7 in a two-dimensional lattice network topology, where the paths in underlying main chain topology are non-dominated. In the same way, in other underlying chain topologies, the paths are non-dominated, creating correlated links.

6.2 Simulation Parameters
Chapter 6 Configuring Network Topology for Simulations

In this section an overview of the simulation input and output parameters will be given. Simulation input parameters are used to configure a two-dimensional lattice network topology. For each simulation the following input parameters are used:

- **Correlation coefficients:**
  - Path correlation coefficient (\( \rho \)) is used in the range \([-1, 1]\) with the steps of 0.1 to show how strongly the path weights are correlated.
  - Link correlation coefficient (\( P_{\text{link}} \)) is used in the range \([-1, 1]\) with the steps of 0.1 to show how strongly the link weights are correlated.
- **Number of nodes \((N)\):** 49, 100 and 169. In a two-dimensional lattice network topology the square root of the number of nodes must be an integer.
- **Number of iterations \((\text{iter})\):** In one simulation a two-dimensional lattice network topology, with different two-dimensional link weight vectors on each link, is generated \(10^4\) times.

Thus, all simulations consisted of generating a two-dimensional lattice network topology \(10^4\) times and in each network topology the optimal path is computed satisfying constraints via the SAMCRA algorithm. Using small constraints leads to a very high probability that there is no path within the constraints. On the contrary, using very large constraints leads to a very high probability of easily finding a path within the constraints (this is only true if SAMCRA was only solving the MCP problem). In the thesis the constraints are chosen to be large, such that all possible paths in a network topology could satisfy them. The reason for choosing the large constraints is that SAMCRA now must search for the optimal path in the largest possible search-space. Recall that SAMCRA, besides the MCP problem, also solves the MCOP problem. This means that SAMCRA retrieves not only the feasible paths in a network but also the optimal paths. The optimal path is thus the path with smallest length from the set of all feasible paths.

Simulation output parameters demonstrate the complexity of QoS routing of the simulated network topology. For each simulation the following output parameters will be generated and stored by the simulation program:

- **\(K_{\text{min}}\)** is the queue-size parameter, needed to find a feasible path within the constraints. \(K_{\text{min}}\) can grow as a factorial if NP-complete behaviour is encountered and therefore it is related to the complexity of Multi-Constrained QoS routing. Each simulation consists of \(10^4\) repeated generation of a two-dimensional lattice network topology. Each generation results in one observation of the \(K_{\text{min}}\) queue-size parameter. After \(10^4\) generations, \(10^4\) sample values of \(K_{\text{min}}\) queue-size parameter are generated.
- **Expected value, \(E\),** or mean of the queue-size variable \(K_{\text{min}}\)

\[
E[K_{\text{min}}] = \sum_{k_{\text{min}}} k_{\text{min}} P_{K_{\text{min}}}[k_{\text{min}}],
\]

where \(k_{\text{min}}\) is one of the possible values of the \(K_{\text{min}}\) queue-size parameter and \(P_{K_{\text{min}}}[k_{\text{min}}]\) is the probability that the \(K_{\text{min}}\) queue-size parameter will have that specific \(k_{\text{min}}\) value.

- **Variance, \(\text{VAR}\),** of the queue-size variable \(K_{\text{min}}\)

\[
\text{VAR}[K_{\text{min}}] = E[(K_{\text{min}} - E[K_{\text{min}}])^2] = E[K_{\text{min}}^2] - (E[K_{\text{min}}])^2.
\]

- **Maximum, \(\text{MAX}\),** of the queue-size variable \(K_{\text{min}}\)

\[
\text{MAX}[K_{\text{min}}] = \max[K_{\text{min}}],
\]
where \( \max[K_{min}] \) is the maximum \( K_{min} \) value from \( 10^4 \) generated values of the \( K_{min} \) queue-size parameter.

6.3. Configuring Two-Dimensional Lattice Network Topology

6.3.1. Configuring Link Weights Consecutively

To construct a two-dimensional lattice network topology, where the path and the link correlation vectors consecutively are linked to the first and the second link weights, the following programming approach is used:

1. The correlation matrix represents the path correlation or the correlation between paths weights in a network. Therefore, the correlation matrix is generated using one of the proposed Class structures. For a detailed description on different correlation matrices, Section 6.1.1 on “Generating Path Correlation Classes via Correlation Matrix” could be scrutinized.

2. From the generated correlation matrix, the Gaussian path correlation vector \( \mathbf{Z} \) is generated and the corresponding parameters refer to the first link weight of each link in a network. This means that the first link weight is a Gaussian random variable, which is actually generated from an uncorrelated Gaussian random variable.

3. The second link weight is correlated to the first link weight using the link correlation coefficient \( \rho_{\text{link}} \). Consequently, the Gaussian link correlation vector \( \mathbf{X} \) is created and the corresponding parameters refer to the second link weight of each link in a network.

4. Linking of the parameters of the path correlation vector \( \mathbf{Z} \) to the first link weight is done consecutively, meaning that the first weight of the first link correspond to the first parameter of the path correlation vector \( \mathbf{Z} \), the first weight of the second link corresponds to the second parameter of the path correlation vector \( \mathbf{Z} \), etc. In the same way is the linking of the parameters of the link correlation vector \( \mathbf{X} \) to the second link weight completed, although here the second weight of the first link corresponds to the first parameter of the link correlation vector \( \mathbf{X} \), the second weight of the second link corresponds to second parameter of the link correlation vector \( \mathbf{X} \), etc.

5. As a result a two-dimensional lattice network topology is generated where the components of a two-dimensional link weight vector are Gaussian correlated random variables (Figure 6.3.1-1).

**Figure 6.3.1-1** Consecutive configuration of link and path vectors.
Using the above procedure the following simulations for a two-dimensional lattice network topology are completed:

1. Varying $\rho$ while $\rho_{\text{link}}$ is constant. 
   - $\rho$ is changed in the range $[-1,1]$ with the steps of 0.1.
   - $\rho = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

2. Varying $\rho_{\text{link}}$ while $\rho$ is constant.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
   - $\rho = 1.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

3. Varying $\rho_{\text{link}}$ while $\rho$ is constant.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
   - $\rho = -1.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

To construct a two-dimensional lattice network topology with on each link a two-dimensional link weight vector, where the first and second weights represent correlated paths via underlying chain topologies, the following programming approach is used:

A two-dimensional lattice network topology has several chain topologies enclosed in its structure. Applying the non-dominance property on all underlying chain topologies ensures the creation of non-dominated paths. The creation of non-dominated paths ensures in turn that the link weights between succeeding nodes in those underlying chain topologies are correlated with each other. Variables of two-dimensional link weight vectors are Uniformly distributed random variables. For a detailed description of this method, where path correlation is created using the non-dominance property, Section 3.3 on “Non-Dominance Property in Two-Dimensional Lattice Network Topology” could be scrutinized.

Using the above procedure the following simulation for a two-dimensional lattice network topology is also completed:

4. 
   - $\text{Class } 7$ of path correlation is used.
   - $\rho$ and $\rho_{\text{link}}$ have no influence.
   - $\text{iter} = 10000$, $N = 49, 100$ and $169$.

### 6.3.2. Configuring Link Weights Randomly

To construct a two-dimensional lattice network topology with on each link a two-dimensional link weight vector, where the path and the link correlation vectors are linked randomly to the first and the second link weights, the following programming approach is used:

1. The correlation matrix represents the path correlation or the correlation between path weights in a network. Therefore, the correlation matrix is generated using one of the proposed Class structure. For a detailed description on different correlation matrices, Section 6.1.1 on “Generating Path Correlation Classes via Correlation Matrix” could be scrutinized.

2. From the generated correlation matrix, the Gaussian path correlation vector $\mathbf{Z}$ is generated and the corresponding parameters refer to the first link weight of each link in a network. This means that that the first link weight is a Gaussian random variable, which is actually generated from an uncorrelated Gaussian random variable.
3. The second link weight is correlated to the first link weight using the link correlation coefficient \( \rho_{\text{link}} \). Consequently, the Gaussian link correlation vector \( \mathbf{X} \) is created and the corresponding parameters refer to the second link weight of each link in a network.

4. Linking of the parameters of the path correlation vector \( \mathbf{Z} \) to the first link weight is done randomly, meaning that the first weight of the first link corresponds to the first randomly chosen parameter of the path correlation vector \( \mathbf{Z} \), the first weight of the second link corresponds to the second randomly chosen parameter of \( \mathbf{Z} \), etc.

In the same way is the linking of the parameters of the link correlation vector \( \mathbf{X} \) to the second link weight completed, although here the second weight of the first link corresponds to the first randomly chosen parameter of the link correlation vector \( \mathbf{X} \), the second weight of the second link corresponds to the second randomly chosen parameter of \( \mathbf{X} \), etc.

5. As a result, a two-dimensional lattice network topology is generated where the components of a two-dimensional link weight vector are Gaussian correlated random variables (Figure 6.3.2-1).

![Figure 6.3.2-1](image)

Using the above procedure, the following simulations for a two-dimensional lattice network topology were completed:

1. Varying \( \rho \) while \( \rho_{\text{link}} \) is constant. Class 1 of path correlation is used.
   - \( \rho \) is changed in the range \([-1, 1]\) with the steps of 0.1.
     - \( \rho = 0.0, \text{iter} = 10000, N = 49, 100 \) and 169.

2. Varying \( \rho \) while \( \rho_{\text{link}} \) is constant. Class 1 of path correlation is used.
   - \( \rho \) is changed in the range \([-1, 1]\) with the steps of 0.1.
     - \( \rho_{\text{link}} = -1.0, \text{iter} = 10000, N = 49, 100 \) and 169.

3. Varying \( \rho \) while \( \rho_{\text{link}} \) is constant. Class 2 of path correlation is used.
   - \( \rho \) is changed in the range \([-1, 1]\) with the steps of 0.1.
     - \( \rho_{\text{link}} = 0.0, \text{iter} = 10000, N = 49, 100 \) and 169.

4. Varying \( \rho \) while \( \rho_{\text{link}} \) is constant. Class 2 of path correlation is used.
   - \( \rho \) is changed in the range \([-1, 1]\) with the steps of 0.1.
     - \( \rho_{\text{link}} = -1.0, \text{iter} = 10000, N = 49, 100 \) and 169.

5. Varying \( \rho_{\text{link}} \) while \( \rho \) is constant; Class 1 of path correlation is used.
   - \( \rho_{\text{link}} \) is changed in the range \([-1, 1]\) with the steps of 0.1.
     - \( \rho = -1.0, \text{iter} = 10000, N = 49, 100 \) and 169.
Chapter 6 Configuring Network Topology for Simulations

6. Varying $\rho_{\text{link}}$ while $\rho$ is constant. Class 1 of path correlation is used.
   - $\rho_{\text{link}}$ is changed in the range [-1,1] with the steps of 0.1.
   - $\rho = 1.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

7. Class 3 of path correlation is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

8. Class 4 of path correlation is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

9. Class 5 of path correlation is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

10. Class 6 of path correlation is used.
    - Varying $\rho$ has no influence.
    - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$. 
7. Simulation Results on Complexity of Multi-Constrained QoS Routing

In the previous chapter the configuration of a two-dimensional lattice network topology was provided and discussed. In the remainder of the thesis, focus will be given on the evaluation of the simulation results. The simulation results, based on the specific path correlation structures, show the performance of Multi-Constrained QoS routing.

7.1. Simulation Results for the Consecutive-Configuration of Link Weights

1. Varying $\rho$ while $\rho_{\text{link}}$ is constant. *Class 1* of path correlation is used.
   - $\rho$ is changed in the range [-1,1] with the steps of 0.1.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and 169.

![Expected kmin queue-size](image)

Figure 7.1-1. Expected kmin queue-size for the *Class 1* with consecutively configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.1-2. Maximum in kmin queue-size for the Class 1 with consecutively configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{link} = 0.0$ is constant.
For the here-presented network configuration, which makes use of path correlation of the Class 1 with the consecutively configured link weights, the following is observed from the Figures 7.1-1 to 7.1-3:

Varying $\rho$ while $\rho_{\text{link}}$ is constant ($\rho_{\text{link}} = 0.0$) does not lead to NP-complete behaviour for the three used number of nodes. The expected kmin queue-size is very small and hardly increases with $N$. It reaches its maximum value when $\rho = 0.0$. This means that the expected kmin queue-size reaches its maximum when the path correlation in the network topology is zero. At the same time the maximum is reached when the link correlation is zero too. Thus, the maximum value of the expected kmin queue-size is encountered when there is no correlation in the network topology. However, the expected kmin queue-size in the whole range is close to one, similar to the complexity of Dijkstra’s algorithm. Therefore, the conclusion can be made that this specific path correlation structure has no influence on the complexity of Multi-Constrained QoS routing.
2. Varying $\rho_{\text{link}}$ while $\rho$ is constant. Class 1 of path correlation is used.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
     - $\rho = 1.0$, $\text{iter} = 10 000$, $N = 49$, 100 and 169.

For the here-presented network configuration, which makes use of path correlation of the Class 1 and the consecutively configured link weights, the following is observed:

Varying $\rho_{\text{link}}$ while $\rho$ is constant ($\rho = 1.0$) does not lead to NP-complete behaviour for the three used number of nodes. This type of network configuration can be seen as an extreme case of positive path correlation because the $i$-th weights of all links are correlated to each other with the path correlation coefficient $\rho = 1.0$ (see Property 2.7 from Section 2.7). Consequently, the conclusion can be made that this specific path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing. Recall that the simulation figures are not presented here but they are available on the CD, which is provided with the thesis.

3. Varying $\rho_{\text{link}}$ while $\rho$ is constant. Class 1 of the path correlation is used.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
     - $\rho = -1.0$, $\text{iter} = 10 000$, $N = 49$, 100 and 169.

For the here-presented network configuration, which makes use of path correlation of the Class 1 and the consecutively configured link weights, the following is observed:

Varying $\rho_{\text{link}}$ while $\rho$ is constant ($\rho = -1.0$) leads to the NP-complete behaviour for the number of nodes of 100 and 169. In detail, this network configuration is an example of negative path correlation structure that already is observed in [9]. There, the conclusion is made that negative path correlation structure induces NP-complete behaviour, what indeed is observed here. Nevertheless, this specific path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing because the probability that such extreme negative values will appear in practice is very small. For details on this specific path correlation structure [9] can be consulted. Recall that the simulation figures are not presented here but they are available on the CD, which is provided with the thesis.
4. *Class 7* of path correlation is used.
   - $\rho$ and $\rho_{\text{link}}$ have no influence.
   - $\text{iter} = 10\,000$, $N = 49, 100$ and 169.

![Graph showing Expected kmin-size](image)

**Figure 7.1-4.** Expected kmin queue-size for the *Class 7* with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{link}}$ do not have influence on the simulation results.
Figure 7.1-5. Maximum in kmin queue-size for the Class 7 with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{dla}}$ do not have influence on the simulation results.
Figure 7.1-6. Variance in kmin queue-size for the Class 7 with consecutively configured vectors of Jointly Uniform Random Variables, where $\rho$ and $\rho_{\text{link}}$ do not have influence on the simulation results.

For the here-presented network configuration, which makes use of path correlation of the Class 7, the following is observed from the Figures 7.1-4 to 7.1-6:

For the three used number of nodes NP-complete behaviour is not encountered. The expected kmin queue-size increases linearly until the value of 20 is reached. Recall that the path and the link coefficients do not have any influence on the simulation results because this class is generated via underlying chain topologies, creating non-dominated paths. Furthermore, the weights on each link are uniformly distributed what leads to somewhat worse performance than in the Gaussian case. With uniformly distributed link weights, there is higher probability that the link weight vectors are dissimilar, leading to a larger variability and therefore to a somewhat worse performance than in the Gaussian case. However, the complexity is not NP-complete for the three inspected number of nodes. If this complexity can be extrapolated for large $N$, the conclusion can be made that this specific path correlation structure has no influence on the complexity of Multi-Constrained QoS routing.
7.2. Simulation Results for Random-Configuration of the Link Weights

1. Varying $\rho$ while $\rho_{\text{link}}$ is constant. Class 1 of path correlation is used.
   - $\rho$ is changed in the range [-1,1] with the steps of 0.1.
     $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and $169$.

Figure 7.2-1. Expected kmin queue-size for the Class 1 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-2. Maximum in $k_{\text{min}}$ queue-size for the $Class_1$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-3. Variance in kmin queue-size for the Class 1 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.

For the here-presented network configuration, which makes use of path correlation of the Class 1 and the randomly configured link weights, the following is observed from the Figures 7.2-1 to 7.2-3:

Varying $\rho$ while $\rho_{\text{link}}$ is constant ($\rho_{\text{link}} = 0.0$) does not lead to NP-complete behaviour for the three used number of nodes. The expected kmin queue-size hardly increases with $N$ and it reaches its maximum value when $\rho = 0.0$. This means that the expected kmin queue-size reaches its maximum when the path correlation in the network topology is zero. At the same time the maximum is reached when the link correlation is zero too. Thus, the maximum value of the expected kmin queue-size is encountered when there is no correlation in the network topology. However, the expected minimum queue-size in the whole path correlation coefficient range is close to one, similar to the complexity of Dijkstra’s algorithm. Therefore, the conclusion can be made that this specific path correlation structure gives the same simulation results as the path correlation structure presented in Section 7.1. To conclude, different configurations of the Class 1 structure do not have negative influence on the complexity of the Multi-Constrained QoS routing.
2. Varying $\rho$ while $\rho_{\text{link}}$ is constant. Class 1 of path correlation is used.
   - $\rho$ is changed in the range $[-1, 1]$ with the steps of 0.1.
   - $\rho_{\text{link}} = -1.0$, $\text{iter} = 10\,000$, $N = 49, 100$ and $169$.

![Expected kmin queue-size](image)

Figure 7.2-4. Expected kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.
Figure 7.2-5. Maximum in kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.
Figure 7.2-6. Variance in kmin queue-size for the Class 1 and $N = 49$ with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.

For the here-presented network configuration, which makes use of path correlation of the Class 1 and the randomly configured link weights, the following is observed from the Figures 7.2-4 to 7.2-6:

Varying $\rho$ while $\rho_{\text{link}}$ is constant ($\rho_{\text{link}} = -1.0$) for 49 nodes does not lead to NP-complete behaviour. The figures show that in the range of positive path correlation the expected kmin slightly increases until the correlation between the path weights is zero. From that point the QoS complexity is constant for the entire range of negative path correlation. On the other hand, for 100 and 169 nodes and in the path correlation range $-1.0 \leq \rho \leq 0.9$ this network configuration does lead to NP-complete behaviour. For these two numbers of nodes simulations could not be finished, due to an exponential increase in kmin queue-size. This network configuration induces NP-complete behaviour because it can be seen as an extreme case of negative link correlation, meaning that the link weights are correlated with the link correlation coefficient $\rho_{\text{link}} = -1.0$. Consequently, the conclusion can be made that this specific correlation structure has indeed negative influence on the complexity of the QoS routing.
3. Varying $\rho$ while $\rho_{\text{link}}$ is constant. Class 2 of path correlation is used. 
   - $\rho$ is changed in the range [-1,1] with the steps of 0.1.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49, 100$ and 169.

Figure 7.2-7. Expected kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-8. Maximum in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.
Chapter 7 Simulation Results on Complexity of Multi-Constrained QoS Routing

Figure 7.2-9. Variance in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = 0.0$ is constant.

For the here-presented network topology, which makes use of path correlation of the Class 2 and the randomly configured link weights, the following is observed from the Figures 7.2-7 to 7.2-9:

Varying $\rho$ while $\rho_{\text{link}}$ is constant ($\rho_{\text{link}} = 0.0$) does not lead to NP-complete behaviour for 49 nodes. For 100 and 169 nodes and in the path correlation range $-1.0 \leq \rho \leq -0.1$ this structure gives the Cholesky decomposition failure, meaning that for this region no results on the complexity are provided. The strongest increase in the expected kmin queue-size is generally encountered when the negative correlation occurs, inducing NP-complete behaviour. In the here-presented network configuration the link correlation coefficient is zero, meaning that the link weights are not correlated with each other. Furthermore, the complexity of correlated path weights is almost constant for the entire inspected range. Therefore, the conclusion could be made that for the entire path correlation coefficient range this specific correlation structure probably does not give negative results for the complexity of Multi-Constrained QoS routing.
Chapter 7 Simulation Results on Complexity of Multi-Constrained QoS Routing

4. Varying $\rho$ while $\rho_{\text{link}}$ is constant. *Class 2* of path correlation is used.
   - $\rho$ is changed in the range [-1,1] with the steps of 0.1.
     $\rho_{\text{link}} = -1.0$, $\text{iter} = 10\,000$, $N = 49, 100$ and 169.

![Expected kmin-size](image)

Figure 7.2-10. Expected kmin queue-size for the *Class 2* with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.
Figure 7.2-11. Maximum in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.
Figure 7.2-12. Variance in kmin queue-size for the Class 2 with randomly configured vectors of Jointly Gaussian Random Variables, where $\rho$ is variable while $\rho_{\text{link}} = -1.0$ is constant.

For the here-presented network configuration, which makes use of path correlation of the Class 2 and the randomly configured link weights, the following is observed from the Figures 7.2-10 to 7.2-12:

The network configuration presented earlier in this section differs from this network configuration in the structure of the correlation matrix. However, varying $\rho$ while $\rho_{\text{link}}$ is constant ($\rho_{\text{link}} = -1.0$) leads also to NP-complete behaviour for the three used number of nodes. In detail, for 49 nodes and in the path correlation coefficient range $-1.0 \leq \rho \leq 0.0$ NP-complete behaviour is encountered. Furthermore, for 100 and 169 nodes and in the path correlation coefficient range $-1.0 \leq \rho \leq 0.9$ NP-complete behaviour is encountered too. For these three numbers of nodes simulations could not be finished, due to an exponential increase in kmin queue-size. This network configuration induces NP-complete behaviour because it can be seen as an extreme case of negative link correlation, meaning that the link weights on each link are correlated with the link correlation coefficient $\rho_{\text{link}} = -1.0$. For that reason, the conclusion can be made that this specific correlation structure has indeed negative influence on the complexity of the Multi-Constrained QoS routing.
Chapter 7 Simulation Results on Complexity of Multi-Constrained QoS Routing

5. Varying $\rho_{\text{link}}$ while $\rho$ is constant. \textit{Class 1} of path correlation structure is used.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
     
     \[ \rho = -1.0, \text{iter} = 10000, N = 49, 100 \text{ and } 169. \]

For the here-presented network configuration, which makes use of path correlation of the \textit{Class 1} and randomly configured link weights, the following is observed:

Varying $\rho_{\text{link}}$ while $\rho$ is constant ($\rho = -1.0$) does not lead to NP-complete behaviour for the three used number of nodes, except for extreme negative values of the link and the path correlation coefficient. This network configuration can be seen as an example of negative path correlation structure because the $i$-th weights of all links are correlated with $\rho = -1.0$. The comparable network configuration presented in Section 7.1 has resulted in very similar simulation results. Therefore, the same conclusion can be made here. Recall that the simulation figures are not presented but they are available on the CD, which is provided with the thesis.

6. Varying $\rho_{\text{link}}$ while $\rho$ is constant. \textit{Class 1} of path correlation structure is used.
   - $\rho_{\text{link}}$ is changed in the range $[-1,1]$ with the steps of 0.1.
     
     \[ \rho = 1.0, \text{iter} = 10000, N = 49, 100 \text{ and } 169. \]

For the here-presented network configuration, where path correlation of the \textit{Class 1} is used and the configuration of the weights on each link is random, the following is observed:

Varying $\rho_{\text{link}}$ while $\rho$ is constant ($\rho = 1.0$) does not lead NP-complete behaviour for all the three number of nodes. This type of network configuration can be seen as an extreme case of positive path correlation, meaning that the $i$-th weights of all links are correlated to each other with the path correlation coefficient $\rho = 1.0$ (see Property 2.7-2 from Section 2.7). The same simulation results are obtained from the network configuration presented in Section 7.1. Hence, this path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing. Recall that the simulation figures are not presented but they are available on the CD, which is provided with the thesis.
7. Class 3 of the path correlation structure is used.
   • Varying $\rho$ has no influence.
   • $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49,100$ and 169.

![Expected kmin queue-size](image)

Figure 7.2-13. Expected kmin queue-size for the Class 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-14. Maximum in kmin queue-size for the \textit{Class} 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{ink}} = 0.0$ is constant.
Chapter 7  Simulation Results on Complexity of Multi-Constrained QoS Routing

Figure 7.2-15.  Variance in kmin queue-size for the Class 3 with randomly configured vectors of Jointly Gaussian Random Variables, where varying \( \rho \) has no influence and \( \rho_{\text{link}} = 0.0 \) is constant.

For the here-presented network configuration, which makes use of path correlation of the Class 3 and the randomly configured link weights, the following is observed from the Figures 7.2-13 to 7.2-15:

The Class 3 of path correlation does not result in an exponential increase in the expected kmin queue-size for the three used number of nodes. Recall that the path correlation coefficient does not have any influence on the simulation results because the coefficients in the correlation matrix are chosen to be random. Also, the nearest correlation matrix problem is used to solve the positive definiteness property. If this behaviour can be extrapolated to large \( N \), the conclusion can be made that this specific path correlation structure does not have negative influence on the complexity of the Multi-Constrained QoS routing.

- 69 -
8. *Class 4* of the path correlation structure is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10\,000$, $N = 49, 100$ and $169$.

![Expected kmin queue-size](image)

Figure 7.2-16. Expected kmin queue-size for the *Class 4* with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-17. Maximum in kmin queue-size for the $Class 4$ with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-18. Variance in kmin queue-size for the Class 4 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.

For the here-presented network configuration, which makes use of path correlation of the Class 4 and randomly configured link weights, the following is observed from the Figures 7.2-16 to 7.2-18:

The Class 4 of path correlation does not lead to NP-complete behaviour for the three used number of nodes. The expected kmin queue-size increases linearly with $N$ but does not exceed the value of 5. If this behaviour can be extrapolated to large $N$, the conclusion can be made that this specific path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing. Moreover, the Class 3 and the Class 4 of path correlation show similar behaviour. This behaviour is possibly caused by the nearest correlation matrix problem, which finds the same nearest positive definite matrix for both structures, resulting in the same QoS performance. Recall that the path correlation coefficient does not have any influence on the simulation results because the coefficients in the correlation matrix are chosen to be random.
9. *Class 5* of the path correlation structure is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10\,000$, $N = 49,100$ and 169.

![Figure 7.2-19. Expected kmin queue-size for the *Class 5* with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.](image-url)
Figure 7.2-20. Maximum in kmin queue-size for the Class 5 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Chapter 7 Simulation Results on Complexity of Multi-Constrained QoS Routing

![Variance in kmin queue-size](image)

Figure 7.2-21. Variance in kmin queue-size for the Class 5 with randomly configured vectors of Jointly Gaussian Random Variables, where varying \( \rho \) has no influence and \( \rho_{\text{link}} = 0.0 \) is constant.

For the here-presented configuration, where the path correlation of the Class 5 is used and the configuration of the weights on each link is random, the following is observed from the Figures 7.2-19 to 7.2-21:

The expected kmin queue-size does not increase exponentially for the used number of nodes. In addition, it remains small with increased number of nodes. If this behaviour can be extrapolated to large \( N \), the conclusion can be made that this specific path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing. Moreover, the Classes 3, 4 and 5, which have the randomised correlation matrix structure, show similar behaviour. The only difference is that the worst-case values (MAX[kmin]) have a little decrease for 100 nodes and a little increase for 169 nodes. The same behaviour is possibly caused by the nearest correlation matrix problem, which finds the similar nearest correlation member for all three structures, resulting in the same QoS performance. Recall that the path correlation coefficient does not have any influence on the simulation results because the coefficients in the correlation matrix are chosen to be random.
10. *Class 6* of the path correlation is used.
   - Varying $\rho$ has no influence.
   - $\rho_{\text{link}} = 0.0$, $\text{iter} = 10000$, $N = 49$, 100 and 169.

![Expected kmin-size](image)

**Figure 7.2-22.** Expected kmin queue-size for the *Class 6* with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2. Maximum in kmin queue-size for the Class 6 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.
Figure 7.2-24. Variance in kmin queue-size for the Class 6 with randomly configured vectors of Jointly Gaussian Random Variables, where varying $\rho$ has no influence and $\rho_{\text{link}} = 0.0$ is constant.

For the here-presented network configuration, where path correlation of the Class 6 is used and the configuration of the weights on each link is random, the following is observed from the Figures 7.2-22 to 7.2-24:

The Class 6 of path correlation does not lead to NP-complete behaviour for the three used number of nodes. The expected kmin queue-size is increasing with $N$ but it does not display an exponential increase in kmin. This network configuration shows similar QoS behaviour as the network configuration of randomised path correlation Classes. If this behaviour can be extrapolated for the large $N$, the conclusion can be made that this specific path correlation structure does not have negative influence on the complexity of Multi-Constrained QoS routing. Recall that the path correlation coefficient does not have any influence on the simulation results because the correlation matrix is generated from the matrix with the randomly ordered positive eigenvalues and the matrix whose columns are the ordered mutually orthogonal unit eigenvectors. At the same time the link correlation coefficient is zero, meaning that the weights on each link are uncorrelated.
8. Conclusions and Recommendations

One of the basic problems in QoS routing is how to find a path in a network with constraints on multiple QoS metrics. This problem is also often referred to as the Multi-Constrained Path (MCP) problem. The MCP problem has been proven to be NP-complete, indicating that guaranteed QoS routing is impossible. However, the MCP problem is not strong NP-complete. This property suggests that the MCP problem is solvable with pseudo-polynomial time algorithms, such as SAMCRA.

The goal of this M.Sc. thesis was to evaluate and analyse the influence of path correlation on the complexity of Multi-constrained QoS routing. The analysis is made through the proof of NP-completeness, which suggests that specific correlation structures between the link weights have influence on the complexity of Multi-Constrained QoS routing. With the aim of confirming the theory behind the NP-complete proof, two different path correlation structures are generated. The first path correlation structure uses the correlation matrix to create the path correlation or the correlation between the path weights. For the reason that the correlation matrix is a valid correlation matrix if and only it is a positive definite, symmetric matrix, many correlation structures were scrutinised. Furthermore, the nearest correlation matrix problem and the eigenvalues and eigenvectors problem were invoked to deal with the positive definiteness. Finally, the two components of the link weight vector are correlated Gaussian distributed random variables. The second path correlation structure uses underlying chain topologies to create non-dominated and simultaneously correlated paths. Hence, the two components of the link weight vector are correlated uniformly distributed random variables. The complexity of QoS routing was evaluated through simulations via an exact QoS routing algorithm, declared as SAMCRA. All simulation consisted of generating $10^4$ times a two-dimensional lattice network topology with the above-described path correlation structures.

The simulations results on the complexity of Multi-Constrained QoS routing for a two-dimensional lattice network topology, have displayed the following complexity behaviour:

- The NP-completeness is encountered for the Class 1 and Class 2 of path correlation with randomly configured vectors of Jointly Gaussian random variables. This correlation structure can be seen as an extreme case of negative link correlation, where the weights on each link are correlated to each other with the link correlation coefficient $\rho_{\text{link}} = -1.0$.
- The NP-completeness is encountered for the Class 1 of path correlation with consecutively and randomly configured vectors of Jointly Gaussian random variables. However, NP-completeness is only encountered for extremely negative values of the path and the link correlation coefficients ($\rho = -1.0$ and $\rho_{\text{link}} = -1.0$). This correlation structure can be seen as an extreme case of negative path correlation, where the weights on each link are correlated to each other with the link correlation coefficient $\rho = -1.0$.

Conclusively, the simulation results indicate that the scrutinized classes of path correlation only display NP-complete behaviour if the used values of $\rho$ and $\rho_{\text{link}}$ are extremely negative. Similar performance is already observed in [9]. The possibility that the weights will display such a negative correlation is very small in practice. In summary, the simulation results implicate that in practice an exact QoS routing algorithm, such as SAMCRA, can work in polynomial-time and therefore can make guaranteed Multi-Constrained QoS routing to have a promising future.
Future study should be focused on a more thorough analysis of the non-dominance property, applied in a two-dimensional lattice network topology. Also, generation of a network topology with $m$-dimensional link weight vector on each link could be a chance to examine other correlation structures that could lead to NP-complete behaviour. However, it is essential to analyse the correlation matrix problem in some other way because positive definiteness brings restrictions on the correlation matrix and its correlation coefficient. Finally, a more thorough analysis of the correlation matrix, which is generated from the eigenvalues and its eigenvectors, would be the right method to get to the bottom of the problem.
Appendix A

Orthogonality [6]

A set of vectors \( \{x_1, \ldots, x_p\} \) in \( \sim^m \) is **orthogonal** if \( x_i'x_j = 0 \) whenever \( i \neq j \) and **orthonormal** if \( x_i'x_j = \delta_{ij} \). Intuitively, orthogonal vectors are maximally independent for they point in totally different directions.

A collection of subspaces \( S_1, \ldots, S_p \) in \( \sim^m \) is **mutually orthogonal** if \( x'y = 0 \) whenever \( x \in S_i \) and \( y \in S_j \) for \( i \neq j \). The **orthogonal complement** of a subspace \( S \subseteq \sim^m \) is defined by

\[
S^\perp = \{ y \in \sim^m : y'x = 0 \text{ for all } x \in S \}
\]

and it is not hard to show that \( \text{ran}(A)^\perp = \text{null}(A') \). The vectors \( v_1, \ldots, v_k \) form an **orthonormal** basis for a subspace \( S \subseteq \sim^m \) if they are orthonormal and span \( S \).

A matrix \( Q \in \sim^{m \times m} \) is said to be **orthogonal** if \( Q'Q = I \). If \( Q = [q_1, \ldots, q_m] \) is orthogonal, then the \( q_i \) form an orthonormal basis for \( \sim^m \). It is always possible to extend such a basis to a full orthonormal basis \( \{v_1, \ldots, v_m\} \) for \( \sim^m \):

**Theorem A-1.**

If \( V_1 \in \sim^{n \times r} \) has orthonormal columns, then there exist \( V_2 \in \sim^{n \times (n-r)} \) such that \( V = [V_1 V_2] \) is orthogonal. Note that \( \text{ran}(V_1)^\perp = \text{ran}(V_2) \).
Appendix B

Positive Definite Matrices [3]

Let \( A \) denote a \( p \times p \) matrix, and define a function \( Q_A : \mathbb{R}^p \to \mathbb{R} \) by
\[
Q_A(x) = x'Ax = \sum_{i=1}^{p} \sum_{j=1}^{p} a_{ij}x_i x_j.
\]
This function is called a quadratic form. If \( A \) is not symmetric, we can define a symmetric matrix \( B = (A + A')/2 \), then \( Q_A(x) = Q_B(x) \) because \( x'Ax = (x'Ax)' = x'A'x \). Thus we can always assume without loss of generality that the matrix in a quadratic form is symmetric. The quadratic form \( Q_A(x) \), or equivalently the matrix \( A \), is called positive definite if \( x'Ax > 0 \) for all nonzero \( x \in \mathbb{R}^p \) (\( x \neq 0 \)), and positive semidefinite if \( x'Ax \geq 0 \) for all \( x \in \mathbb{R}^p \). The abbreviation \( pds \) is used for positive definite symmetric matrices.

Suppose the portioned matrix \( A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \) is \( pds \). Then both \( A_{11} \) and \( A_{22} \) are \( pds \). This follows by taking vectors \( x = \begin{pmatrix} x_1 \\ 0 \end{pmatrix} \) and \( x = \begin{pmatrix} 0 \\ x_2 \end{pmatrix} \), respectively. Suppose the dimension of \( A_{11} \) is \( k \times k \). Writing \( C = \begin{pmatrix} I_k & 0 \\ -A_{21}A_{11}^{-1} & I_{p-k} \end{pmatrix} \) and noticing that
\[
C^{-1} = \begin{pmatrix} I_k & 0 \\ A_{21}A_{11}^{-1} & I_{p-k} \end{pmatrix},
\]
the following is obtained
\[
x'Ax = x'C^{-1}(CAC')(C')^{-1}x = y'\begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}y,
\]
where \( y = (C')^{-1}x \). It follows that \( A_{22,1} \) is \( pds \), and similarly that \( A_{11,2} \) is \( pds \).

More useful in practice are the following properties, which are equivalent to above explained requirement.

- The matrix \( A \) is positive definite if all its principal minors \( A_1, A_2, \ldots, A_n \) have strictly positive determinants. The determinant of a positive definite matrix is positive but the converse is not necessarily true.
- The matrix also is positive definite if all his eigenvalues are positive.
- If the determinants are nonzero and alternate in sign, starting with \( \det(A_1) < 0 \), then the matrix \( A \) is negative definite.
- If the determinants are all nonnegative, then the matrix is positive semidefinite.
- If the determinant alternate in sign, starting with \( \det(A_1) \leq 0 \), then the matrix is negative semidefinite.
Appendix C

The Cholesky Decomposition

Suppose $A$ is positive definite symmetric ($pd$s) of dimension $p \times p$. Then there exist a unique lower triangular matrix $C$ with positive diagonal entries such that $A = CC^\prime$. This is called the Cholesky decomposition (factorisation) of $pd$s matrix. First the existence of such matrix will be proved and then some computational aspects will be discussed.

The proof is induction on the dimension $p$. For $p = 1$ the matrix $A$ is real number, and the Cholesky decomposition is $A = c^2$ where $c = \sqrt{A}$.

Assume now that $p > 1$ and the Cholesky decomposition has been shown for dimension $p - 1$. Partition $A$ as

$$A = \begin{pmatrix} A_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},$$

where $a_{22}$ is scalar. By the induction hypothesis there exists a unique lower triangular matrix $C_{p-1}$ of dimension $(p - 1) \times (p - 1)$ with positive diagonal entries such that $A_{11} = C_{p-1}C_{p-1}'$.

Define

$$C = \begin{pmatrix} C_{p-1} & 0 \\ c & c_{pp} \end{pmatrix},$$

where $c \in \mathbb{R}^{p-1}$ and $c_{pp} > 0$ have yet to be determined. $A = CC'$ is wanted, i.e.,

$$C_{p-1}C_{p-1}' = A_{11},$$

$$C_{p-1}c = a_{12},$$

$$c'c + c_{pp}^2 = a_{22}.$$

Since $C_{p-1}$ is non-singular, it follows that we have to choose $c = C_{p-1}^{-1}a_{12}$ and

$$c_{pp} = (a_{22} - c'c)^{1/2} = \left(a_{22} - a_{21}A_{11}^{-1}a_{12}\right)^{1/2} = \sqrt{a_{22,1}}.$$

The Cholesky decomposition is of great value in numerical analysis because it can be computed precisely without iteration. As the above proof by induction shows, we can compute the factorisation by first finding the factorisation of the leading $1 \times 1$ matrix, then the factorisation of the leading $2 \times 2$ matrix, and so on. The crucial calculation in each step is solving the equation $C_{p-1}c = a_{12}$ for $c$; but since $C_{p-1}$ is lower triangular this as an easy task.

Since the determinant of a triangular matrix is the product of its diagonal elements, it follows that

$$\det(A) = \prod_{i=1}^{p} c_{ii}^2.$$

If $A = CC'$ is the Cholesky decomposition of a $pd$s matrix $A$, then $A^{-1} = (C')^{-1}C^{-1}$.

Putting $y = C^{-1}x$ it follows that $x'Ax = y'y > 0$ for all $y \in \mathbb{R}^p (y \neq 0)$. That is if $A$ is $pd$s, then $A^{-1}$ is $pd$s as well.
Appendix D

C-code for Cholesky Decomposition Algorithm [23]

```c
int chol_dec_new(a,n,p)
double **a,p[];
int n;

/*
Computes Cholesky decomposition of a given matrix a[1..n][1..n]. On input, only the upper triangle of a has to
be given; it is not modified. The Cholesky factor is returned in the lower triangle of the matrix a except for its
diagonal elements which are returned in p[1..n].
*/
{
    int i,j,k;
    double sum;

    for (i=1;i<=n;i++) {
        for (j=i;j<=n;j++) {
            for (sum=a[i][j],k=i-1;k>=1;k--)
                sum -= a[i][k]*a[j][k];
            if (i == j){
                if (sum <= 0.0){
                    return 0;
                }
                p[i]=sqrt(sum);
                else a[j][i]=sum/p[i];
            }
        }
    }
    return 1;
}
```

Appendix E

MATLAB M-code for Nearest Correlation Matrix Problem [7,16]

% The main function
function X=Near_CorM(A,W,tol)
[m,n]=size(A);
S=zeros(n);
Y=A;
Z=Y;

R=Y-S;
X=Ps(R,W);
S=X-R;
Y=Pu(X,W);
while cond1(Z,Y)>tol;
    Z=Y;
    R=Y-S;
    X=Ps(R,W);
    S=X-R;
    Y=Pu(X,W);
end;

% The function of projection A onto S, a positive definite symmetric matrix
function Xk=Ps(Rk, W)
V=real(sqrtm(W));
B=V*Rk*V;
[m,n]=size(B);
[Q,lemda]=eig(B);
for i=1:m
    if lemda(i,i)<0
        d(i)=0.001;
    else d(i)=lemda(i,i);
    end
end
Bplus=Q*diag(d)*Q';
C=inv(V);
Xk=C*Bplus*C;

% The function of projection Xk onto U, a symmetric unit diagonal matrix
function Yk=Pu(Xk,W)
[m,n]=size(Xk);
U=inv(W);
B=U.*U;
b=diag(Xk-eye(n));
x=B \ b;
Yk=Xk-U*diag(x)*U;

% The functions computing the loop condition
function c=cond1(Y,X)
Delta=X-Y;
NormDelta=Max_Norm(Delta);
NormX=Max_Norm(X);
c=NormDelta/NormX;

% The function computing the infinite norm of a matrix
function max=Max_Norm(X)
Appendix E MATLAB M-code for Nearest Correlation Matrix Problem

[m,n]=size(X);
max=0;
for i=1:m
    sum(i)=0;
    for j=1:n
        sum(i)=sum(i)+abs(X(i,j));
    end;
    if sum(i)>max
        max=sum(i);
    end;
end;

% Compute the W norm of matrix A
function x=W_Norm(A,W)
    B=sqrtm(W);
    C=B*A*B;
    x=F_Norm(C);

% Compute the F_norm of a matrix A
function x=F_Norm(A)
    [m,n]=size(A);
    x=0;
    for i=1:m
        for j=1:n
            x=x+abs(A(i,j))^2;
        end
    end
    x=sqrt(x);
Appendix F

C-code for Nearest Correlation Matrix Problem

void nearest_corr_matrix(a,n)
double **a;
int n;
/* make nearest correlation matrix from a given matrix a[1..n][1..n] in the F-norm where w=I */
{
    int i,j,k,iter;
    double *d;
    double **r, **s, **v, **w, **x, **y;
    double **d_temp, **v_temp, **x_temp;
    void jacobi_new();

d = dvector (1,n);
r = dmatrix (1,n,1,n);
s = dmatrix (1,n,1,n);
v = dmatrix (1,n,1,n);
w = dmatrix (1,n,1,n);
x = dmatrix (1,n,1,n);
y = dmatrix (1,n,1,n);
d_temp = dmatrix (1,n,1,n);
v_temp = dmatrix (1,n,1,n);
x_temp = dmatrix (1,n,1,n);

for(i=1;i<=n;i++){
    for(j=1;j<=n;j++){
        r[i][j]=0.0;
        s[i][j]=0.0;
        v[i][j]=0.0;
        x[i][j]=0.0;
        y[i][j]=0.0;
        d_temp[i][j]=0.0;
        x_temp[i][j]=0.0;
        v_temp[i][j]=0.0;
    }
}
/* initialization */
for(i=1;i<=n;i++){
    for(j=1;j<=n;j++){
        w[i][j]=0.0;
    }
    w[i][i]=1.0;
}
/* w = I */
for(i=1;i<=n;i++){
    for (j=1;j<=n;j++){
        y[i][j]=a[i][j];
    }
}
/* y = a */
for (iter=1;iter<=10;iter++){
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            x[i][j]=0.0;
            x_temp[i][j]=0.0;
        }
    }
    /* x = 0 and x_temp = 0 */
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            if (x[i][j] != 0.0) {
                r[i][j] = y[i][j] - s[i][j];
            } else {
                r[i][j] = y[i][j] - s[i][j];
            }
        }
    }
    /* r = y-s */
}
Appendix F

C-code for Nearest Correlation Matrix Problem

/* following functions project matrix a onto s (in fact a=r) */
/* s is a positive definite symmetric matrix: x=Ps(r) */
/* ---------------------------------------------------------- */

/* jacobi algorithm to determine eigenvalues and eigenvectors */

jacobi_new(r,n,d,v); /* for the c code see end of this section */
for (i=1;i<=n;i++){
    for (j=i+1;j<=n;j++){
        r[i][j]=r[j][i];
    }
    for (i=1;i<=n;i++){
        if (d[i]<=0.0000) d[i]=0.0001;
        for (j=1;j<=n;j++){
            if (i==j) d_temp[i][j]=d[i];
        }
    }
    // making eigenvalues positive and the diagonal matrix of eigenvalues
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            for (k=1;k<=n;k++){
                x_temp[i][j]+=v[i][k]*d_temp[k][j];
            }
        }
    }
    // x_temp = v*d_temp
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            v_temp[i][j]=v[j][i];
        }
    }
    // v_temp = v'
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            for (k=1;k<=n;k++){
                x[i][j]+=x_temp[i][k]*v_temp[k][j];
            }
        }
    }
    // x = x_temp*v_temp = v*d_temp*v'
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            s[i][j]=x[i][j]-r[i][j];
        }
    }
    // s = x-r
    /* following functions projects matrix a onto u (in fact a=x) */
    /* u is a symmetric unit diagonal matrix: y=Pu(x) */
    /* ---------------------------------------------------------- */
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            if (i==j) x_temp[i][j]=x[i][j]-w[i][j];
            else x_temp[i][j]=0.0;
        }
    }
    // x_temp = diagonal(x-w)
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            y[i][j]=x[i][j]-x_temp[i][j];
        }
    }
    // y = x-x_temp
    for (i=1;i<=n;i++){
        for (j=1;j<=n;j++){
            a[i][j]=x[i][j];
        }
    }
    // a = x
Appendix F C-code for Nearest Correlation Matrix Problem

```c
free_dvector (d,1,n);
free_dmatrix (r,1,n,1,n);
free_dmatrix (s,1,n,1,n);
free_dmatrix (v,1,n,1,n);
free_dmatrix (w,1,n,1,n);
free_dmatrix (x,1,n,1,n);
free_dmatrix (y,1,n,1,n);
free_dmatrix (d_temp,1,n,1,n);
free_dmatrix (v_temp,1,n,1,n);
free_dmatrix (x_temp,1,n,1,n);
}

The Jacobi Algorithm [23]
void jacobi_new(a,n,d,v)
/* Computes all eigenvalues and eigenvectors of a real symmetric matrix a[1..n][1..n]. On output, elements
   of a above the diagonal are destroyed. d[1..n] returns the eigenvalues of a. v[1..n][1..n] is a matrix whose
   columns contain, on output, the normalized eigenvectors of a. */
double **a,**v;
double d[];
int n;
{
    int j,iq,ip,i;
    double tresh,theta,tau,sm,s,h,g,c,*b,*z;
    b = dvector(1,n);
    z = dvector(1,n);
    for (ip=1;ip<=n;ip++){
        for (iq=1;iq<n;iq++) v[ip][iq]=0.0;
        v[ip][ip]=1.0;
    }
    for (ip=1;ip<=n;ip++)
        b[ip]=d[ip]=a[ip][ip];
    z[ip]=0.0;
    for (i=1;i<=50;i++)
    {
        sm=0.0;
        for (ip=1;ip<=n-1;ip++)
            for (iq=ip+1;iq<=n;iq++)
                sm +=fabs(a[ip][iq]);
        if (sm == 0.0){
            free_vector(z,1,n);
            free_vector(b,1,n);
            return;
        }
        if (i < 4)
            tresh=0.2*sm/(n*n);
        else
            tresh=0.0;
        for (ip=1;ip<=n-1;ip++)
            for (iq=ip+1;iq<=n;iq++)
                sm +=fabs(a[ip][iq]);
        if (sm == 0.0){
            free_vector(z,1,n);
            free_vector(b,1,n);
            return;
        }
        if (i < 4)
            tresh=0.2*sm/(n*n);
        else
            tresh=0.0;
        for (ip=1;ip<=n-1;ip++)
            for (iq=ip+1;iq<=n;iq++)
                g=100.0*fabs(a[ip][iq]);
            if (i > 4 && (float)(fabs(d[ip])+g) == (float)fabs(d[ip]) &&
                (float)(fabs(d[iq])+g) == (float)fabs(d[iq]))
                a[ip][iq]=0.0;
            else if (fabs(a[ip][iq]) > tresh) {
                h=d[iq]-d[ip];
                if ((float)(fabs(h)+g) == (float)fabs(h))
                    t=(a[ip][iq])/h;
                else {
                    theta=0.5*h/(a[ip][iq]);
                    t=1.0/(fabs(theta)+sqrt(1.0+theta*theta));
                    if (theta < 0.0) t = -t;
                }
```

93
\[ c = \frac{1.0}{\sqrt{1+t^2}}; \]
\[ s = t \cdot c; \]
\[ \tau = \frac{s}{1.0+c}; \]
\[ h = t \cdot a[ip][iq]; \]
\[ z[ip] := h; \]
\[ z[iq] += h; \]
\[ d[ip] := h; \]
\[ d[iq] += h; \]
\[ a[ip][iq] = 0.0; \]
\[ \text{for (j=1;j<=ip-1;j++)} \]
\[ \quad \text{ROTATE}(a,j,ip,iq) \]
\[ \text{for (j=ip+1;j<=iq-1;j++)} \]
\[ \quad \text{ROTATE}(a,ip,j,j,iq) \]
\[ \text{for (j=iq+1;j<=n;j++)} \]
\[ \quad \text{ROTATE}(a,ip,j,iq,j) \]
\[ \text{for (j=1;j<=n;j++)} \]
\[ \quad \text{ROTATE}(v,j,ip,j,iq) \]
\[ \}
\[ \text{for (ip=1;ip<=n;ip++)} \]
\[ \quad b[ip] += z[ip]; \]
\[ \quad d[ip] = b[ip]; \]
\[ \quad z[ip] = 0.0; \]
\[ \}
\[ \text{nrerror("Too many iterations in routine jacobi");} \]
\[ \text{free_dvector(b,1,n);} \]
\[ \text{free_dvector(z,1,n);} \]
Appendix G

Part of Correlation.C Program where Path Correlation Classes are generated

Path Correlation of the Class 1

```c
int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */

/* generates correlation matrix a[1..E][1..E] of the class 1, 
with rho in the range [-1,1] */
if(rho<1.0&&rho>-1.0){
  if(rho>=0.0){
    for(i=1;i<=E;i++){
      for(j=i+1;j<=E;j++){
        a[i][j] = rho;
        a[j][i]=a[i][j];
      }
      a[i][i]= 1.0;
    }
  }
  else{
    for(i=1;i<=E;i++){
      for(j=i+1;j<=E;j+=2){
        a[i][j] = rho;
        a[j][i]=a[i][j];
      }
      for(j=i+2;j<=E;j+=2){
        a[i][j] = -1*rho;
        a[j][i]=a[i][j];
      }
      a[i][i]= 1.0;
    }
  }

  /* computes the chol_decomp. of the correlation matrix */
  chol_dc_new(a, E, p_chol);

  /* generates the lower triangular chol_decomp. */
  for(i=1;i<=E;i++){
    a[i][i]=p_chol[i];
    for(j=i+1;j<=E;j++){
      a[i][j]=0.0;
    }
  }

  /* makes correlated (rho) links with first weight Z[i] */
  if(rho>=1.0){
    for(i=1;i<=E;i++){
      X[i]=gasdev(&idum);
      Z[i]=X[1];
    }
  }
  else{
    for(i=1;i<=E;i++){
      X[i]=gasdev(&idum);
      if(i%2==1)
        Z[i]=X[1];
      else
        Z[i]=-1*X[i];
    }
  }
  if(rho<1.0&&rho>-1.0){
    ...
```
Appendix G  Part of Correlation.C Program where Path-Correlation Classes are generated

```c
for(i=1;i<=E;i++){
    X[i]=gasdev(&idum);
    Z[i]=0.0;
    for(j=1;j<=E;j++){
        Z[i]+=a[i][j]*X[j];
    }
}
/* makes second link weight that is correlated to the first with rholink */
for(i=1;i<=E;i++){
    X[i]=rholink*Z[i] + sqrt(1-rholink*rholink)*(gasdev(&idum));
}
/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);
```

Path Correlation of the Class 2

```c
int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */

/* generates correlation matrix a[1..E][1..E] of the class 2, 
with rho in the range [-1,1] */
if(rho<1.0&&rho>-1.0){
    if(rho>=0.0){
        for(i=1;i<=E;i++){
            for(j=i+1;j<=E;j++){
                a[i][j] = rho;
                a[j][i]=a[i][j];
            }
            a[i][i]= 1.0;
        }
    }
    else{
        for(i=1;i<=E;i++){
            for(j=i+1;j<=E;j+=2){
                a[i][j] = -1*rho;
                a[j][i]=a[i][j];
            }
            for(j=i+2;j<=E;j+=2){
                a[i][j] = rho;
                a[j][i]=a[i][j];
            }
            a[i][i]= 1.0;
        }
    }
}
/* computes the cholesky decomp. of the correlation matrix */
chol_dc_new(a, E, p_chol);
/* generates the lower triangular cholesky matrix */
for(i=1;i<=E;i++){
    a[i][i]=p_chol[i];
    for(j=i+1;j<=E;j++){
        a[i][j]=0.0;
    }
}
/* makes correlated (rho) links with first weight Z[i] */
if(rho>=1.0){
    for(i=1;i<=E;i++){
        X[i]=gasdev(&idum);
        Z[i]=X[1];
    }
}
```
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

if(rho<=-1.0){
    for(i=1;i<=E;i++){
        X[i]=gasdev(&idum);
        if(i%2==1)
            Z[i]=X[1];
        else
            Z[i]=-1*X[i];
    }
}
if(rho<1.0&&rho>-1.0){
    for(i=1;i<=E;i++)
        X[i]=gasdev(&idum);
    Z[i]=0.0;
    for(j=1;j<=E;j++)
        Z[i]+=a[i][j]*X[j];
}
/* makes second link weight that is correlated to the first with rhoLink */
for(i=1;i<=E;i++)
    X[i]=rhoLink*Z[i] + sqrt(1-rhoLink*rhoLink)*(gasdev(&idum));

/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);

Path Correlation of the Class 3

int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */
void nearest_corr_matrix(); /* see Appendix F */

/* generates correlation matrix a[1..E][1..E] of the class 3, 
with random elements in the range [-1,1] */
for(i=1;i<=E;i++)
    for(j=i+1;j<=E;j++)
        a[i][j]= 2*ran4(&idum) - 1;
    a[j][i]= a[i][j];
    a[i][i]= 1.0;
/* computes the cholesky decomposition of the correlation matrix if 
the correlation matrix is positive definite*/
chol=0;
chol = chol_dec_new(a,E,p_chol);
while (chol==0)
    for(i=1; i<=E; i++)
        for (j=i+1; j<=E; j++)
            a[j][i] = a[i][j];
    a[i][i] = 1.0;
/* computes the nearest correlation matrix followed by the cholesky decomposition 
of the nearest correlation matrix. */
nearest_corr_matrix(a,E);
chol = chol_dec_new(a,E,p_chol);
/* generates the lower triangular cholesky matrix */
for(i=1;i<=E;i++)
    a[i][i]=p_chol[i];
    for(j=i+1;j<=E;j++)
        a[i][j]=p_chol[j];
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

```c
a[i][j]=0.0;
}

/* makes correlated (rho) links with first weight Z[i] */
for(i=1;i<=E;i++){
    X[i]=gasdev(&idum);
    Z[i]=0.0;
    for(j=1;j<=E;j++){
        Z[i]+=(a[i][j]*X[j]);
    }
}

/* makes second link weight that is correlated to the first with rholink */
for(i=1;i<=E;i++){
    X[i]=rholink*Z[i] + sqrt(1-rholink*rholink)*(gasdev(&idum));
}

/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);
```

Path Correlation of the Class 4

```c
int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */
void nearest_corr_matrix(); /* see Appendix F */

/* generate correlation matrix a[1..E][1..E] of the class 4, 
with random elements in the range [-1,1] */
for(i=2; i<=E; i++){
    a[1][i]= 2*ran4(&idum) - 1;
    a[i][1]= a[1][i];
}
for(i=2; i<=E; i++){
    for(j=i+1; j<=E; j++){
        a[i][j]= a[i-1][j-1];
        a[j][i]= a[i][j];
    }
}
for(i=1; i<=E; i++){
    a[i][i]= 1.0;
}

/* compute cholesky decomposition of the correlation matrix
   if the correlation matrix is positive definite */
chol=0;
chol = chol_dec_new(a,E,p_chol);
while (chol==0){
    for(i=1; i<=E; i++){
        for (j=i+1; j<=E; j++){
            a[j][i] = a[i][j];
        }
    }
    a[i][i] = 1.0;
}

/* computes the nearest correlation matrix followed by cholesky decomposition
   of the nearest correlation matrix */
nearest_corr_matrix(a,E);
chol = chol_dec_new(a,E,p_chol);
```

/* generates the lower triangular cholesky matrix */
for(i=1;i<=E;i++){
    a[i][i]=p_chol[i];
    for(j=i+1;j<=E;j++){
        a[i][j]=0.0;
    }
}
```
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

} /* makes correlated (rho) links with first weight Z[i] */
for(i=1;i<=E;i++){
    X[i]=gasdev(&idum);
    Z[i]=0.0;
    for(j=1;j<=E;j++){
        Z[i]+=a[i][j]*X[j];
    }
}
/* makes second link weight that is correlated to the first with rholink */
for(i=1;i<=E;i++){
    X[i]=rholink*Z[i] + sqrt(1-rholink*rholink)*(gasdev(&idum));
}
/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);

Path Correlation of the Class 5

int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */
void nearest_corr_matrix(); /* see Appendix F */
/* generate correlation matrix a[1..E][1..E] of the class 5,
   with random elements in the range [-1,1] */
for(i=2; i<=E; i=i+2){
    a[1][i]= 2*ran4(&idum) - 1;
    a[i][1]= a[1][i];
    a[1][i+1]=-a[1][i];
    a[i+1][1]=a[1][i+1]
}
for(i=2; i<=E; i=i+2){
    for(j=i+1; j<=E; j++){
        a[i][j]= a[i-1][j-1];
        a[j][i]= a[i][j];
    }
}
/* compute cholesky decomposition of the correlation matrix
   if the correlation matrix is positive definite */
chol=0;
chol = chol_dec_new(a,E,p_chol);
while (chol==0){
    for(i=1; i<=E; i++){
        for (j=i+1; j<=E; j++){
            a[j][i] = a[i][j];
        }
        a[i][i] = 1.0;
    }
    /* computes the nearest correlation matrix followed by cholesky decomposition
       of the nearest correlation matrix */
    nearest_corr_matrix(a,E);
    chol = chol_dec_new(a,E,p_chol);
}
/* generates the lower triangular cholesky matrix */
for(i=1;i<=E;i++){
    a[i][i]=p_chol[i];
    for(j=i+1;j<=E;j++){
        a[i][j]=0.0;
    }
}
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

```c
} /* makes correlated (rho) links with first weight Z[i] */
for(i=1;i<=E;i++)
    X[i]=gasdev(&idum);
    Z[i]=0.0;
    for(j=1;j<=E;j++)
        Z[i]+=(a[i][j]*X[j]);
}
/* makes second link weight that is correlated to the first with rholink */
for(i=1;i<=E;i++)
    X[i]=rholink*Z[i] + sqrt(1-rholink*rholink)*(gasdev(&idum));
/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);

Path Correlation of the Class 6

int chol_dc_new(); /* see Appendix D */
void sort_random(); /* see Appendix H */
void svdcmp_new(); /* see Appendix F */

/* generates correlation matrix a[1..E][1..E] with random eigenvalues and eigenvectors in the range [-1,1] */
for (i=1;i<=E;i++)
    for (j=1;j<=E;j++)
        eig_val_matrix[i][j] = 0.0;
    }
eig_val_matrix[i][i] = 2*ran4(&idum) - 1;
    if (eig_val_matrix[i][i]<=0.0)
        eig_val_matrix[i][i]=
    eig_val_matrix[i][i]=
/* generates matrix with E random eigenvalues */
for (i=1;i<=E;i++)
    for (j=1;j<=E;j++)
        eig_vec_matrix[i][j] = 2*ran4(&idum) - 1;
    }
eig_vec_matrix[i][i] = 1.0;
/* generates random correlation matrix */
/* constructs orthonormal basis by running random correlation matrix through SVD algorithm or constructs an orthogonal matrix */
svdcmp_new(eig_vec_matrix,E,E); /* for the c code see end of this section */
for (i=1;i<=E;i++)
    for (j=1;j<=E;j++)
        for (k=1;k<=E;k++)
            a_temp[i][j]+=eig_vec_matrix[i][k]*eig_val_matrix[k][j];
/* a_temp = eig_vec_matrix*eig_val_matrix */
for (i=1;i<=E;i++)
    for (j=1;j<=E;j++)
        eig_vec_trns[i][j]=eig_vec_matrix[j][i];
/* eig_vec_trns = eig_vec */
for (i=1;i<=E;i++)
    for (j=1;j<=E;j++)
        for (k=1;k<=E;k++)
            a[i][j]+=a_temp[i][k]*eig_vec_trns[k][j];
/* a=a_temp*eig_vec_trns=eig_vec_matrix*eig_val_matrix*eig_vec_matrix' */
/* normalization of the correlation matrix */
```

100
Appendix G  Part of Correlation.C Program where Path-Correlation Classes are generated

```c
a_temp[1][1]=a[1][1];
for (i=1;i<=E;i++)
{
  for (j=1;j<=E;j++)
  {a[i][j]=a[i][j]/a_temp[1][1];
  }
}
/* computes cholesky decomposition of the correlation matrix if correlation matrix is positive definite */
chol=0;
while (chol==0)
{
  chol = chol_dec_new(a,E,p_chol);
}
/* generates the lower triangular cholesky matrix */
for(i=1; i<=E; i++)
{
  a[i][i] = p_chol[i];
  for(j=i+1; j<=E; j++)
  {a[i][j] = 0.0;
  }
}
/* makes correlated (rho) links with first weight Z[i] */
for(i=1;i<=E;i++)
{
  X[i]=gasdev(&idum);
  Z[i]=0.0;
  for(j=1;j<=E;j++)
  {Z[i]+=a[i][j]*X[j];
  }
}
/* makes second link weight that is correlated to the first with rho link */
for(i=1;i<=E;i++)
{
  X[i]=rholink*Z[i] + sqrt(1-rholink*rholink)*(gasdev(&idum));
}
/* random sorting of Z[i] and X[i] */
sort_random(&idum,E,Z);
sort_random(&idum,E,X);
```

The Singular Value Decomposition (SVD) Algorithm [23]

```c
void svdcmp_new(a,m,n)
/* Given matrix a[1..m][1..n], this routine computes its singular value decomposition.
The matrix U replaces a on output. The columns of the matrix U are desired orthonormal basis vectors. */
double **a;
int m,n;
{
  double pythag_new(double a,double b);
  int flag,i,its,j,jj,k,l,nm;
  double anorm,c,f,g,h,s,scale,x,y,z,*rv1;
  double *w,**v;
  rv1=dvector(1,n);
  w=dvector(1,n);
  v=dmatrix(1,n,1,n);
  g=scale=anorm=0.0;
  for (i=1;i<=n;i++)
  {
    l=i+1;
    rv1[i]=scale*g;
    g=s=scale=0.0;
    if(i<m)
    {
      for(k=i;k<=m;k++) scale+=fabs(a[k][i]);
      if(scale)
      {for(k=i;k<=m;k++)
         {a[k][i]/=scale;
          s+=a[k][i]*a[k][i];
         }
      f=s;}
```
Appendix G  Part of Correlation.C Program where Path-Correlation Classes are generated

```c
g=-SIGN(sqrt(s),f);
h=f*g*s;
a[i][j]=f*g;
for(i=1;i<=n;i++){
    for(s=0.0,k=1;k<=m;k++) s+=a[k][i]*a[k][j];
    f=s/h;
    for(k=1;k<=m;k++) a[k][j]+=f*a[k][i];
    for(k=1;k<=m;k++) a[k][i]*=scale;
}
}
w[i]=scale*g;
g=s=scale=0.0;
for(i=1;i<=n;i++)
    if(scale){
        for(k=1;k<=n;k++) a[i][k]=a[i][k]/scale;
        s+=a[i][k]*a[i][k];
    }
    f=a[i][1];
g=-SIGN(sqrt(s),f);
h=f*g*s;
a[i][1]=f*g;
for(i=1;i<=n;i++)
    for(s=0.0,k=1;k<=n;k++) s+=a[i][k]*v[k][i];
    for(k=1;k<=n;k++) a[i][k]+=s*v[k][i];
    for(k=1;k<=n;k++) a[i][k]*=scale;
}
}
anorm=FMAX(anorm,(fabs(w[i])+fabs(rv1[i])));
}
for(i=n;i>=1;i--){
    if(i<n){
        if(g){
            for(j=1;j<=n;j++) v[j][i]=(a[i][j]/a[i][i])/g;
        }
        for(j=1;j<=n;j++) v[j][i]=v[j][i]/=1.0;
        g=rv1[i];
        l=i;
    }
}
for(i=IMIN(m,n);i>=1;i--){
    l=i+1;
    g=w[i];
    for(j=1;j<=n;j++) a[i][j]=0.0;
    if(g){
        g=1.0/g;
        for(j=1;j<=n;j++)
            for(s=0.0,k=1;k<=m;k++) s+=a[k][i]*a[k][j];
            f=(s/a[i][i])*g;
            for(k=1;k<=m;k++) a[k][j]+=f*a[k][i];
    }else for(j=1;j<=m;j++) a[j][i]=0.0;
    ++a[i][i];
```
for(k=n;k>=1;k--){
    for(its=1;its<=30;its++){
        flag=1;
        for(l=k;l>=1;l--){
            nm=l-1;
            if((double)(fabs(rv1[l])+anorm)==anorm){
                flag=0;
                break;
            }
            if((double)(fabs(w[nm])+anorm)==anorm) break;
        }
        if(flag){
            c=0.0;
            s=1.0;
            for(i=l;i<=k;i++){
                f=s*rv1[i];
                rv1[i]=c*rv1[i];
                if((float)(fabs(f)+anorm)==anorm) break;
                g=w[i];
                h=pythag(f,g);
                w[i]=h;
                h=1.0/h;
                c=g*h;
                s=-f*h;
                for(j=1;j<=m;j++){
                    y=a[j][nm];
                    z=a[j][i];
                    a[j][nm]=y*c+z*s;
                    a[j][i]=z*c-y*s;
                }
            }
        } else {
            z=w[k];
            if(l==k){
                if(z<0.0){
                    w[k]=-z;
                    for(j=1;j<=n;j++) v[j][k]=-v[j][k];
                    break;
                }
            }
        }
    }
    if(its==30) nrerror("no convergence in 30 svdcmp iterations");
    x=w[l];
    nm=k-1;
    y=w[nm];
    g=rv1[nm];
    h=rv1[k];
    f=((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
    g=pythag(f,1.0);
    f=((x-z)*(x+z)+h*((y/(f+SIGN(g,f))-h))/x;
    c=s=1.0;
    for(j=1;j<=nm;j++){
        i=j+1;
        g=rv1[i];
        y=w[i];
        h=s*g;
        g=c*g;
        z=pythag(f,h);
        rv1[j]=z;
        c=f/z;
        s=h/z;
        f=x*c+g*s;
        g=g*c-x*s;
    }
}
Part of Correlation.C Program where Path-Correlation Classes are generated

```c
h = y * s;
y = c;
for (jj = 1; jj <= n; jj ++)
{
x = v[j][j];
z = v[i][j];
v[j][j] = x * c + z * s;
v[i][j] = z * c - x * s;
}
z = pythag(f, h);
w[j] = z;
if (z)
{
  z = 1.0 / z;
c = f * z;
s = h * z;
}
f = c * g + s * y;
x = c * y - s * g;
for (jj = 1; jj <= m; jj ++)
{
y = a[j][j];
z = a[i][j];
a[j][j] = y * c + z * s;
a[i][j] = z * c - y * s;
}
rv1[l] = 0.0;
rv1[k] = f;
w[k] = x;
}
free_dvector(rv1, 1, n);
free_dvector(w, 1, n);
free_dmatrix(v, 1, n, 1, n);
}

double pythag_new(double a, double b)
{
  double absa, absb;
  absa = fabs(a);
  absb = fabs(b);
  if (absa == absb)
    return absa * sqrt(1.0 + SQR(absb / absa));
  else
    return (absb == 0.0 ? 0.0 : absb * sqrt(1.0 + SQR(absb / absa)));
}

Path Correlation of the Class 7

iter = 1;
k = 2 * sqrt(n) - 1;
l = 2 * sqrt(n) + 1;
for (i = 1; i < E; i++)
{
  Z[i] = 0.5 * ran4(&idum);
  Z[i + 1] = 0.5 + (0.5 * ran4(&idum));
  for (j = i + 2; < iter * k; j += 2)
  {
    if (iter > 1 && l >= 2 * sqrt(n) + 1) l = (i + 1) + k;
    Z[j] = 0.5 * ran4(&idum);
    Z[l] = 0.5 + (0.5 * ran4(&idum));
l += k;
  }
l = 2 * sqrt(n) + 1;
iter ++;
} /* correlates two links between first sqrt(n) rows and its perpendicular sqrt(n) columns */
```
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

iter=sqrt(n)-1;
j=(k*iter)+1;
for(i=k;i<E;i=i+k) {
  Z[i]=0.5*ran4(&idum);
  Z[j]=0.5+(0.5*ran4(&idum));
  j++;
} /* correlates two links between the last column and the last row */

iter=1;
temp=2;
l=2*sqrt(n);
for(i=4;i<k;i=i+2) {
  Z[i]=0.5*ran4(&idum);
  Z[l]=0.5+(0.5*ran4(&idum));
  for(j=i+k;j<iter*k;j=j+k) {
    l=l+2;
    Z[j]=0.5*ran4(&idum);
    Z[l]=0.5+(0.5*ran4(&idum));
    if(l==((iter*k)+temp+1)) {
      l=l-temp;
      temp=temp+2;
    }
  }
  l=2*sqrt(n)+1;
  iter++;
} /* correlates two links starting with the second column and its perpendicular row
 and ends with the next to the last column and its perpendicular row */

iter=1;
k=2*sqrt(n)-1;
l=2*sqrt(n)+1;
for(i=1;i<E;i=i+l) {
  X[i]=0.5+(0.5*ran4(&idum));
  X[i+1]=0.5*ran4(&idum);
  for(j=i+2;j<iter*k;j=j+2) {
    if(iter>1 && l==2*sqrt(n)+1) l=(i+1)+k;
    X[j]=0.5+(0.5*ran4(&idum));
    X[l]=0.5*ran4(&idum);
    l=l+k;
  }
  l=2*sqrt(n)+1;
  iter++;
} /* correlates two links of the first sqrt(n) rows and its perpendicular sqrt(n) columns */

iter=sqrt(n)-1;
j=(k*iter)+1;
for(i=k;i<E;i=i+k) {
  X[i]=0.5+(0.5*ran4(&idum));
  X[j]=0.5*ran4(&idum);
  j++;
} /* correlates two links between the last column and the last row */

iter=1;
temp=2;
l=2*sqrt(n);
for(i=4;i<k;i=i+2) {
  X[i]=0.5+(0.5*ran4(&idum));
  X[l]=0.5*ran4(&idum);
  for(j=i+k;j<iter*k;j=j+k) {
    l=l+2;
    X[j]=0.5+(0.5*ran4(&idum));
Appendix G Part of Correlation.C Program where Path-Correlation Classes are generated

```c
X[l]=0.5*ran4(&idum);
if(l==((iter*k)+temp+1)){
    l=l-temp;
    temp=temp+2;
}
l=l+k;
iter++;
}/* correlates two links starting with the second column and its perpendicular row
    ands end with the next to the last column and its perpendicular row */
```
Appendix H

Correlation.C Program

#include <string.h>
#include <stdio.h>
#include <math.h>
#include "/home/kuipersf/nr/include/nr.h"
#include "/home/kuipersf/nr/include/nrutil.h"
#define INF 9.9e99
#define TINY 1.0e-20
#define ROTATE(a,i,j,k,l) g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);a[k][l]=h+s*(g-h*tau);

/* two-dimensional lattice network topology */
void correlatedgrid(seed,n,Adj,numAdj,datAdj,E,rho,rholink,a)
int n,E,**Adj,*numAdj;
long int *seed;
double ***datAdj,rho,rholink;
double **a;
{
int i,j,k,l;
long int idum;
int temp,chol;
double root;
double rand;
double *X;
double *Z;
double *p_chol;

int chol_de_new(); /* for the c-code see Appendix D */
void sort_random(); /* for the c-code see end of this Appendix */
idum = *seed;
p_chol = dvector(1,E);
X = dvector(1, E);
Z = dvector(1, E);

/* generates correlation matrix a[1..E][1..E] of the 7 different classes*/
/* see Appendix G for different c-codes */
/* makes grid topology with E link and 2 link weights */
root = sqrt(n);
temp = (int) root;
for(i=1;i<=n;i++)
    numAdj[i] = 0;
if(temp*temp!=n)
    fprintf(stderr,"the square root of the number of nodes must be an integer\n");
else{
k=1;
for(i=1;i<n;i++)
    j=i+1;
    if(i%temp!=0&&j<n)
        Adj[i][++numAdj[i]]= j;
        Adj[j][++numAdj[j]]= i;
        if(Z[k]+5.0>0.0)
            datAdj[1][i][numAdj[i]]=datAdj[1][j][numAdj[j]]= Z[k]+5.0;
else
The Random Sorting Algorithm [23]

void sort_random(idum, n, arr) {
    int n;
    long int *idum;
    double *arr;
    /* Sorts a given array arr[1..n] in random manner. */
    {
        int i, j, x;
        int *r_arr;
        double *r_temp;
        long int seed;

        r_arr = ivec(1, n);
        r_temp = dvec(1, n);

        seed = *idum;
        x = (int) (n*ran4(&seed) + 1);
        r_arr[1] = x;
        for (i = 2; i <= n; i++) {
            x = (int) (n*ran4(&seed) + 1);
            r_arr[i] = x;
            for (j = 1; j < i; j++) {
                if (r_arr[i] == r_arr[j]) {
                    j = i;
                    i--;
                }
            }
        }
        for (i = 1; i <= n; i++) {
            r_temp[i] = arr[r_arr[i]];
        }
    }
}

The Rando
m Sorting Algorithm [23]
Appendix H Correlation.C Program

```c
arr[i]=r_temp[i];
}
free_ivector(r_arr,1,n);
free_dvector(r_temp,1,n);
```
References

• [16] Su, Q. and Zhao, W., “The Nearest Correlation Matrix and its Applications”, Univesity of Texas, Department of Computational and Applied Mathematics.
  1. Singular Value Decomposition (2.6)
  2. Cholesky Decomposition (2.9)
  3. Jacobi Transformations of a Symmetric Matrix (11.1)
  4. Quicksort (8.2)