MODELLING AND SIMULATION FRAMEWORK INCORPORATING REDUNDANCY AND FAILURE PROBABILITIES FOR EVALUATION OF A MODULAR AUTOMATED MAIN DISTRIBUTION FRAME

by

Marthinus Ignatius Botha

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Electronic Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology
UNIVERSITY OF PRETORIA

October 2012

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

SUMMARY

MODELLING AND SIMULATION FRAMEWORK INCORPORATING REDUNDANCY AND FAILURE PROBABILITIES FOR EVALUATION OF A MODULAR AUTOMATED MAIN DISTRIBUTION FRAME

by

Marthinus Ignatius Botha

Promoters:	Mr H. Grobler, Mr J. H. van Wyk
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Electronic Engineering)
Keywords:	Automated Main Distribution Frame; Stochastic Simulation; Neural Network; Genetic Algorithm; Hybrid Intelligent Algorithm; OMNeT++; Redundancy Optimisation; Redundancy Allocation Problem; Expected Lifetime; System Reliability;

Maintaining and operating manual main distribution frames is labour-intensive. As a result, Automated Main Distribution Frames (AMDFs) have been developed to alleviate the task of maintaining subscriber loops. Commercial AMDFs are currently employed in telephone exchanges in some parts of the world. However, the most significant factors limiting their widespread adoption are cost-effective scalability and reliability. Therefore, an impelling incentive is provided to create a simulation framework in order to explore typical implementations and scenarios. Such a framework will allow the evaluation and optimisation of a design in terms of both internal and external redundancies.

One of the approaches to improve system performance, such as system reliability, is to allocate the optimal redundancy to all or some components in a system. Redundancy at the system or component levels can be implemented in one of two schemes: parallel redundancy or standby redundancy. It is also possible to mix these schemes for various components. Moreover, the redundant elements may or may not be of the same type. If all the redundant elements are of different types, the redundancy optimisation model is implemented with component mixing. Conversely, if all the redundant

© University of Pretoria

components are identical, the model is implemented without component mixing.

The developed framework can be used both to develop new AMDF architectures and to evaluate existing AMDF architectures in terms of expected lifetimes, reliability and service availability. Two simulation models are presented. The first simulation model is concerned with optimising central office equipment within a telephone exchange and entails an environment of clients utilising services. Currently, such a model does not exist. The second model is a mathematical model incorporating stochastic simulation and a hybrid intelligent evolutionary algorithm to solve redundancy allocation problems.

For the first model, the optimal partitioning of the model is determined to speed up the simulation run efficiently. For the second model, the hybrid intelligent algorithm is used to solve the redundancy allocation problem under various constraints. Finally, a candidate concept design of an AMDF is presented and evaluated with both simulation models.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

OPSOMMING

MODELLERING- EN SIMULASIERAAMWERK WAT OORTOLLIGHEID- EN ONDERBREKINGSPROBLEME VIR DIE EVALUERING VAN 'N MODULÊRE GEOUTOMATISEERDE HOOFVERSPREIDINGSRAAM INSLUIT

deur

Marthinus Ignatius Botha

Promotors:	Mnr H. Grobler
Departement:	Elektriese, Elektroniese en Rekenaar-Ingenieurswese
Universiteit:	Universiteit van Pretoria
Graad:	Magister in Ingenieurswese (Elektroniese Ingenieurswese)
Sleutelwoorde:	Geoutomatiseerde Hoofverspreidingsrame; Stogastiese Simulasie; Neurale Netwerk; Genetiese Algoritme; Hibriede Intelligente Algoritme; OMNeT++; Oortolligheidsoptimering; Oortolligheidstoekening-probleem; Verwagte Lewensduur; Stelsel Betroubaarheid;

Die instandhouding en bedryf van handmatige hoofverspreidingsrame is arbeidsintensief. Gevolglik, het geoutomatiseerde hoofverspreidingsrame ontwikkel ten einde die onderhoudsfunksie van kliëntelyne, te verlig. Kommersiële geoutomatiseerde hoofverspreidingsrame word reeds in telefoonsentrales in sekere dele van die wêreld gebruik. Die hoof faktore wat egter die wydverspreide aanvaarding van die tegnologie beperk, is koste-effektiewe skalering en betroubaarheid. Dit was die vernaamste dryfveer om 'n simulasieraamwerk te skep ten einde tipiese toepassings en scenario's te ondersoek. So 'n raamwerk sal die evaluering en optimering van 'n ontwerp in terme van beide interne en eksterne oortolligheid toelaat.

Een van die benaderings wat gevolg kan word om stelseluitvoering te verbeter, byvoorbeeld stelselbetroubaarheid, is om optimale oortolligheid aan alle, of sommige, komponente wat deel uitmaak van 'n stelsel, toe te ken. Komponentoortolligheid op die stelsel- of komponentvlakke kan deur middel van een van twee skemas geïmplementeer word, naamlik paralleloortolligheid of bystandoortolligheid. Dit is egter ook moontlik om hierdie skemas te vermeng vir verskillende komponente. Bowendien mag die oortolligheidselemente van dieselfde soort wees, al dan nie. Indien alle oortolligheidsele-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

mente verskil, word die oortolligheid-optimeringsmodel met komponentvermenging geïmplementeer. Aan die ander kant, indien die komponente identies is, word die model sonder komponentvermenging geïmplementeer.

Twee simulasiemodelle word aangebied. Die ontwikkelde raamwerk kan gebruik word om beide nuwe hoofverspreidingsraamargitekture te ontwikkel en bestaande outomatiese hoofverspreidingsraam argitekture te evalueer in terme van verwagte lewensduur, betroubaarheid en diensbeskikbaarheid. Die eerste simulasiemodel is gemoeid met die optimering van sentrale toerusting binne in ’n telefoonsentrale en behels ’n omgewing van kliënteverbruikersdienste. Tans bestaan daar nie so ’n model nie. Die tweede model is a wiskundige model wat stogastiese simulasie en ’n hibriede intelligente evolusionêre algoritme inkorporeer om oortolligheidstoekenning-probleme op te los.

Vir die eerste model word die optimale verdeling van die model bepaal ten einde die spoed van die simulasie-lopie effektief te verhoog. Vir die tweede model word die hibriede intelligente algortime gebruik om die oortolligheidstoekenning-probleem op te los, gegewe verskeie beperkings. Ten slotte word ’n moontlike konsepontwerp vir ’n geoutomatiseerde hoofverspreidingsraam voorgestel en met behulp van beide simulasie modelle evalueer.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

LIST OF ABBREVIATIONS

ADSL	Asymmetric Digital Subscriber Line
AMDF	Automated Main Distribution Frame
APL	Academic Public License
CO	Central Office
CEL	Current Events List
CM	Component Mixing
CPU	Central Processing Unit
DCS	Digital Cross-connect System
DES	Discrete Event Simulator
DPST	Double-Pole-Single-Throw
DSL	Digital Subscriber Line
EA	Evolutionary Algorithm
EMR	Electromagnetic Relay
FEL	Future Events List
FES	Future Events Set
FTTB	Fibre-to-the-Building
FTTH	Fibre-to-the-Home
GA	Genetic Algorithm

© University of Pretoria

GUI	Graphical User Interface
IDE	Integrated Development Environment
IDF	Intermediate Distribution Frame
ISDN	Integrated Services Digital Network
LMS	Loop Management System
MDF	Main Distribution Frame
MTTR	Mean Time to Repair
MTTF	Mean Time to Failure
MEMS	Micro Electromechanical System
MOEA	Multi-objective Evolutionary Algorithm
MOGA	Multi-objective Genetic Algorithm
MOO	Multi-objective Optimisation
MPI	Message Passing Interface
NED	Network Description
NEMS	Nano Electro-Mechanical System
NMA	Null Message Algorithm
NN	Neural Network
NOC	Network Operations Centre
NPGA	Niched-Pareto Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm

© University of Pretoria

OO	Object-oriented
OSS	Operations Support System
PDES	Parallel Distributed Event Simulation
POTS	Plain Old Telephone Service
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAP	Redundancy Allocation Problem
REVM	Redundancy Expected Value Model
REVMOP	Redundancy Expected Value Multi-Objective Programming
RNG	Random Number Generator
RF	Radio Frequency
SSR	Solid-state Relay
TCP	Transmission Control Protocol
TEMS	Transparent Embedded Magnetic Switch
VoIP	Voice over Internet Protocol
XML	Extensible Markup Language

© University of Pretoria

LIST OF TABLES

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

LIST OF FIGURES

© University of Pretoria

© University of Pretoria

TABLE OF CONTENTS

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

© University of Pretoria

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The most significant cost associated with operating and maintaining a telecommunications service-provider business is labour [1]. In wireline telecommunication companies copper management is especially labour intensive, since it typically involves several manual processes. A large portion of the labour entails performing moves, additions and changes to services at last-mile copper plants. These plants in the telecommunication service network provided by Telkom S.A. Ltd in South Africa are known as Telkom exchanges [2]. Employed in telecommunication, the Main Distribution Frame (MDF) is a passive device which terminates cables, allowing arbitrary interconnections to be made. Turning up (installing and activating) a service, for instance, requires on-site technicians to connect/disconnect jumper cables manually in a connection matrix in order to establish connectivity to a subscriber. Such a connection matrix board in the MDF may contain millions of cross-connection points [3, 4].

1.2 SCOPE

Complex systems often require the development of a software model to simulate the behaviour of the system under various conditions. Simulation tools have been widely used in network research, especially during the development of new systems to ensure that the optimal solution is chosen.

The scope of this research is the development of a simulation framework to evaluate the behaviour and effectiveness of an Automated Main Distribution Frame (AMDF) design. Specifically, the scalability effectiveness is paramount in the design of an MDF or AMDF. High availability and reliability

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

are also major concerns regarding Quality of Service (QoS) in any communication network. These properties should be kept in mind when dealing with the Loop Management System (LMS).

1.3 MOTIVATION

The main motivation is to provide a simulation framework for telecommunication service-provider companies to effectively design and employ AMDFs in telephone exchanges. More importantly, in terms of equipment card failures and faults, a sense of high service availability must be introduced to the AMDF by exploring and evaluating the effects of redundant equipment cards on failover when connected cards fail. Furthermore, redundancy at system and component levels within the AMDF itself must be thoroughly explored to ensure mission-critical operation.

In order to evaluate the effectiveness or feasibility of a concept design, simulation models are usually created to assess the performance of a configuration. At present, no such simulation models exist for the development of an AMDF. Comprehensive simulation models incorporating failure probabilities and redundancies at system and component levels must be realised to evaluate and compare various probabilistic activity profiles.

1.4 OBJECTIVES

The first important objective is to create fully reconfigurable simulation models of a modular AMDF. The first model must be able to exhibit the behaviour of a standard AMDF (i.e. automatic jumpering upon client request), with the addition of the redundancy features. The framework allows various configurations of a modular AMDF to be evaluated. The different configurations could entail, among others, various numbers of clients, clients per AMDF module, redundancy percentages, failover delays, link delays, equipment lifetimes and technician visitation times. The model also allows statistics to be collected throughout the simulation run, including total and average equipment failures per time interval, total and average technician visitations and repair times, as well as the total and average downtime experienced by clients. Because of the size of the model and number of simulation modules, it was required that the workload be evenly distributed over multiple Central Processing Unit (CPU) cores to speed up simulation times. This can be accomplished by partitioning the model into several segments where each segment utilises a separate CPU core.

The second main objective is the creation of a multi-objective optimisation model for the internal

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

design of an AMDF. Developers and researchers are prompted to enter design-specific characteristics via a Graphical User Interface (GUI). Such a model will aid developers/researchers in determining the optimal set of design parameters under various design constraints.

1.5 RESEARCH APPROACH

During the first phase, a flexible simulation model was developed to evaluate the behaviour of an AMDF using OMNeT++. The model is highly reconfigurable (including number of clients, redundancy percentages, modularity, technician visits, random card failure rates, etc.) in order to explore typical real life profiles. The actual number of redundant equipment cards is determined by obtaining the downtime that each client experiences. The second phase entailed the generation of a mathematical model of the system incorporating failure probabilities. This model can be used to determine the optimal redundancy at the system and component levels.

1.5.1 Hypothesis Formulation

The availability of the service provided by telecommunication companies that clients experience can be greatly increased by implementing an AMDF that makes provision for redundant equipment cards. To this end, the main limitations to contend with in providing such high availability include cost and power consumption. A simulation framework helps with the evaluation and comparison of several concept designs addressing specific scenarios, subject to specific design constraints, clients demands and expectations and failure rates.

1.5.2 Research Questions

The main question is whether a modular AMDF can be modelled to effectively evaluate various aspects of its operation ideally. A fully configurable simulation framework is required to simulate various configurations of any number of clients and AMDF modules with reconfigurable parameters. The models can be used to evaluate:

- the optimal equipment card redundancy to achieve a desired service availability percentage for cost, power consumption, failover delays and complexity,
- the optimal system and component level redundancy that can be applied to AMDF modules for guaranteed operation, and

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- the architecture of the modular AMDF.

1.6 WHERE THE MODELS FIT IN

The first model simulates the environment of clients interacting with the access network within a Central Office (CO). The purpose is to determine the service availability, as a percentage, that can be obtained for a certain redundancy percentage of CO equipment cards. The second model evaluates internal redundancies at system and component levels. Given specific constraints (such as cost, power, etc.), the model determines the optimal component and / or system redundancy allocation. In other words, the first model evaluates external redundancies (with respect to the AMDF), whereas the second model evaluates internal redundancies.

1.7 MAIN CONTRIBUTIONS

The primary contributions of this research are:

- The development of a flexible simulation framework for modelling AMDFs and the evaluation of their behaviour. The user is able to reconfigure all relevant aspects to construct an AMDF conforming to any specifications.
- From an application perspective, the developed framework can be used both to develop new AMDF architectures and to evaluate existing AMDF architectures in terms of expected lifetimes, reliability and service availability.
- Candidate concepts designs incorporate redundancy to facilitate high availability of services for clients in telecommunication networks.

The secondary contributions of this research are:

- A full paper was submitted to the 2011 Proceedings of the Southern Africa Telecommunication Networks and Applications (SATNAC) Conference. The paper was accepted and the work was presented at this conference [60].
- A full paper was submitted to the SAIEE Africa Research Journal. No feedback was received prior to the submission of this dissertation.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1.8 STRUCTURE OF THIS DISSERTATION

Chapter 2 provides a general overview of MDFs and current AMDFs. Chapter 3 describes the OMNeT++ environment in detail and its features are compared to the features of other existing network simulators. Chapter 4 provides a description of the implementation of the modular AMDF model. Chapter 5 provides a brief overview of the essential mathematics behind reliability and multi-objective optimisation, including stochastic simulation, neural networks and genetic algorithms. Several numerical example evaluations are performed. Chapter 6 combines the models presented in Chapter 4 and Chapter 5 in a unified conceptual design. Chapter 7 concludes with the discussion of the simulation models.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 2

LITERATURE STUDY

2.1 CHAPTER OBJECTIVES

An overview of MDFs currently employed in telephone exchanges in South Africa is provided. The benefits of automating these devices, possible switching technologies, as well as the requirements of automation, are discussed. The Redundancy Allocation Problem (RAP) is discussed and possible methods for solving them are provided.

2.2 MAIN DISTRIBUTION FRAMES

2.2.1 Role and Structure of the Main Distribution Frames

An MDF acts as a buffer where permanent outside telephone lines terminate within a telephone exchange [1]. At this termination point, all the outside lines are interconnected to specific telecommunication equipment to provide individual subscribers with their appropriate telecommunications service, such as Digital Subscriber Line (DSL) or voice switches. In larger COs, apart from MDFs, Intermediate Distribution Frames (IDFs) are used to distribute lines further. IDFs also serve as isolation points for troubleshooting. Figure 2.1 displays a graphical view of a typical MDF / IDF configuration.

The conventional MDF is a double-sided steel structure containing a switching matrix with terminal strips on both sides [3, 4]. These are commonly referred to as *horizontals* and *verticals* because of their orientation. The horizontal side is the part where the subscriber lines are terminated, whereas the vertical side terminates office equipment cables. Possible termination equipment in a CO include

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

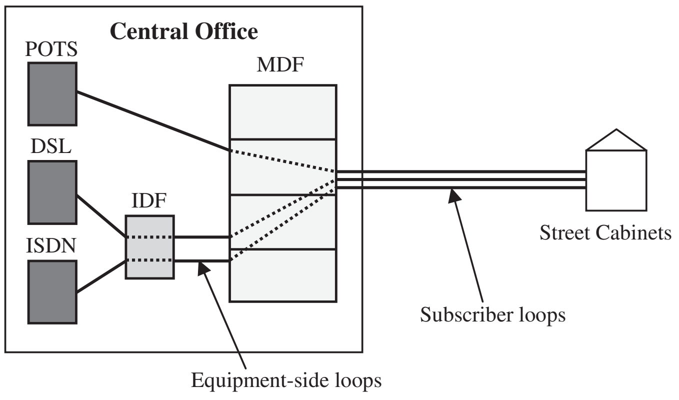


Figure 2.1: Example of a telephone exchange terminating outside plant facilities and interconnecting subscribers to termination equipment. Adapted from [1].

a Public Switched Telephone Network (PSTN), also referred to as the Plain Old Telephone Service (POTS), Voice over Internet Protocol (VoIP), DSL service equipment, such as Asymmetric Digital Subscriber Line (ADSL), Integrated Services Digital Network (ISDN) and in some cases, dial-up Internet facilities.

2.2.2 Problems with Existing Main Distribution Frames

Maintaining distribution frames is very labour-intensive as jumper cross-connects must be completed manually. To this end, technicians are dispatched to the MDF or IDF to make changes. Advanced scheduling and coordination between different service-provider groups are required [1]. Moreover, manual operations are prone to human errors, such as improper insertions (punch-down) and incorrect cross-connects. The problems can drastically delay the turn-up of new services and can even result in the loss of existing services, which, in turn, may cause dissatisfaction with the customer and increase the time-to-revenue for the service or result in unnecessary operational expenditures.

Another problematic aspect is line testing [1, 3, 4]. Line testing in itself can be time-consuming, as technicians are required to disconnect subscribers/office equipment cables from the MDF or IDFs and connect measurement devices at the ports. Normally, a hand-held testset is connected to a newly configured circuit to verify the accuracy and correctness of the installation.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.2.3 Requirements for Automation of Main Distribution Frames

As mentioned before, the primary goal of an AMDF is to alleviate the manual processes involved with MDFs, as well as enhancing the accuracy of copper management in order to reduce cost and increase revenue. In addition, the following requirements must generally be met for copper automation [1, 5]:

1. *Any-to-Any Non-Blocking Connectivity*: The AMDF must have the ability to switch any facility pair (wires in a cable) to any equipment pair regardless of any existing connections or port utilisation in the system. With any-to-any switching, Operations Support System (OSS) components making provisioning assignments or fault-management decisions should be able to perform their functions successfully.
2. *Scalability*: Telephone exchanges vary greatly in size across the world. Smaller, rural areas may require a few thousand copper pair terminations whereas larger, urban areas may need to terminate the connection of several hundred thousand subscribers. An AMDF must scale cost-effectively to meet different-sized requirements.
3. *Reliability*: As distribution frames are the lifeline of telecommunication companies, an AMDF must provide an extremely high degree of confidence that connections are successfully made upon request. Low failure rates and long expected lifetimes are paramount.
4. *OSS Integration*: An AMDF must be integrated directly with any other service activation, inventory, and workflow systems so that cross-connections automatically take place as a software-driven flow. By attaching existing fault management systems to AMDFs, problems can be isolated and resolved pro-actively. A typical OSS is the testing equipment within a telephone exchange.
5. *Automated Connection Verification and Continuous Testing*: An AMDF must have the ability to attach any terminating pair remotely from either subscription or equipment side to a testing system to verify that a connection has been securely and properly connected.

It is important that the design of an AMDF allows it to be scaled linearly to keep pace with subscriber growth rates. In other words, the AMDF must cost-effectively scale with an increase in the number of subscribers. Hence, the ratio of frame ports on the AMDF to the number of subscribers should remain fairly constant as modules are added. To increase the effectiveness of scalability, a modular design of an AMDF is suggested. As user demands (number of subscribers) increase, more modules can be added to the existing frame to increase the capacity of the system.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.2.4 Benefits of Copper Automation

Automated copper switching, remote testing and verification, and OSS integration of an AMDF, can provide the following benefits to service providers [1]:

1. *Reduced Labour Costs*: Automated distribution frames minimise the requirement for on-site personnel to physically perform cross-connections. Workforce efficiency is improved, since technician call-outs to replace faulty hardware are also minimised.
2. *Faster Service Delivery*: New services or changes to existing services can be provided almost instantly. Service activation can take place in minutes as opposed to days.
3. *Improved Accuracy*: Automation significantly reduces errors, eliminating human errors. In addition, testing and verification tools ensure that all changes are applied accurately.
4. *Universal Test Access*: The any-to-any connectivity of the AMDF offers a logical interface for testing all copper pairs (subscriber and equipment side) even at remote sites.
5. *Improved Loop Records*: Service providers can ensure that loop records consistently remain accurate throughout the expected lifetime of operation.
6. *Improved Service Assurance*: The inclusion of automated fault recovery techniques enables nearly perfect network reliability. This, in conjunction with integrated test access, improves the ability to isolate and correct errors, ultimately reducing Mean Time to Repair (MTTR).

2.2.5 Connection Matrix of the Main Distribution Frames

The pin-board matrix of an MDF contains the physical cross-connects between subscriber lines and office equipment lines. In the case where a pure mechanical frame is used, the orientation and layout of the cross-connect holes along with the horizontals and verticals are crucial for a compact design. For the automated copper designs in [3, 4, 6], a similar configuration was used for the matrix board. The configuration is shown in Figure 2.2.

The matrix has X_i (A,B) and Y_j (A,B) connective patterns that are arranged in a perpendicular fashion to each other on different insulated layers. The collective set of subscriber cables are denoted by X, and index i refers to a specific copper pair in the cable. Similarly, the collective set of office equipment cables are denoted by Y, and index j refers to a specific copper pair. Wires A and B (not shown in the figure) are the twisted-pair copper wires in cables coming from a subscriber or going to office equipment. Cross-point holes are arranged in three regions according to their function. X_i (A) is

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

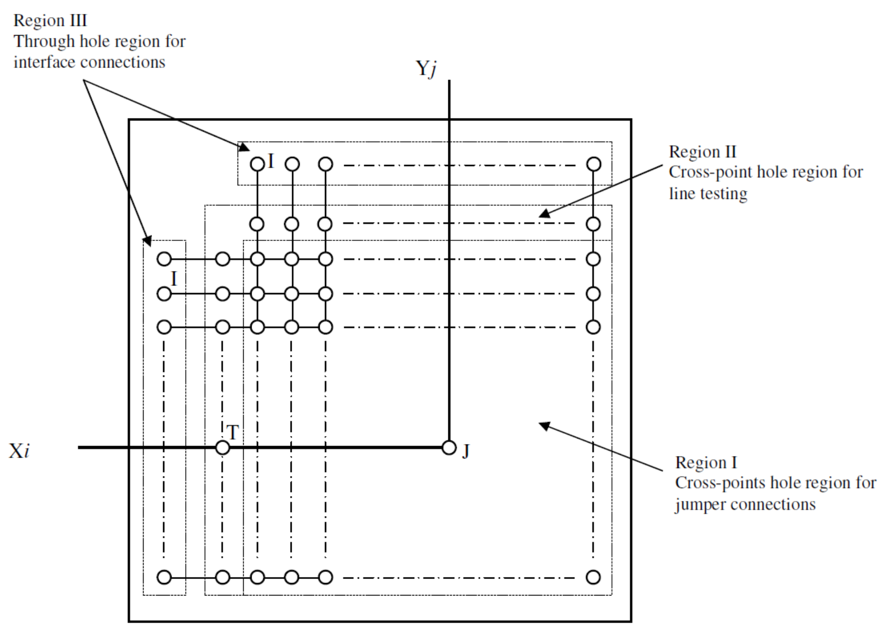

Region III
Through hole region for interface connections
Yj
Region II
Cross-point hole region for line testing
I
I
T
Xi
J
Region I
Cross-points hole region for jumper connections

Figure 2.2: Basic configuration of cross-connect matrix board. Adapted from [3]. ©IEEE 1992.

connected to Y_j (A) and X_i (B) to Y_j (B). Region II holes (T) establish connections for line testing for either X_i (A,B) or Y_j (A,B). Holes (J) in Region I are used for cross-point jumpering to complete the circuit between X_i (A,B) and Y_j (A,B) once a connection pin has been inserted. A third region, Region III, is prepared for interface connectors that connect link cables, subscriber cables and office equipment cables. Current MDFs should be perceived as cross-connect switching systems, which means that MDFs switch circuits, not packets in the data lines. In the digital domain, these devices are known as Digital Cross-connect Systems (DCSs). The first requirement of an AMDF states that any-to-any non-blocking connectivity is required and the described matrix makes provision for any-to-any connectivity. Such a cross-connect matrix, with actuators at each cross-point, is known as a crossbar switch [7, 8].

2.3 CURRENT AMDFS

This section provides a brief overview of the AMDF and its history. The cross-point switching mechanisms and architectures are also discussed.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.3.1 History of the AMDF

Even though the concept of AMDFs is not new, very little research regarding the automation thereof has been conducted to date. The late 1980s to the mid 1990s marked an epoch for researching AMDFs. Using the technology available at the time, robotic solutions described in [1, 3, 4, 9, 10] were the first attempts to automate the copper wire MDF. A robotic arm with a laser-tracking sensor was implemented to insert/remove connection pins physically into/from cross-point holes in a high-density pin-board matrix, similar to human technician operations. A different approach was used for automated optical MDFs where small grooves are implemented as switches (acting as the connection pins at cross-points) that reflect incoming optical signals from subscribers to office equipment and back once filled with viscous oil [11, 12]. The cross-points are said to be *ON* or *OFF* depending on whether the groove is filled with oil or not. The matrix waveguide is, however, considerably smaller than the high-density pin-board matrix since a microfluidics implementation is employed. A similar switching approach is described in [7] for copper MDFs, where electromechanical switches push/pull connecting pins into/from cross-point holes.

The aforementioned concepts are feasible implementations of an AMDF. However, because of the purely mechanical design, robotic and general electromechanical solutions have reliability and maintenance issues due to moving parts. Scalability of such a precise robotic system also comes at a high price for this concept, since very precise calibrations are required for varying-sized systems. Furthermore, existing copper and optical AMDF implementations merely address the routing, or jumpering, aspect of an AMDF. A complete automated MDF should inherit some form of fault detection and correction, as stipulated in the previous section.

2.3.2 Switching Media

As far as automated switching is concerned, various switching media can be considered. Possible switches include Solid-state Relays (SSRs), Electromagnetic Relays (EMRs), microrelays or stepper motors. Recent developments in cross-connect switching arrays entailed the exploitation of Micro Electromechanical Systems (MEMSs), specifically for telecommunication networks [13, 14]. MEMS switches are an attractive alternative to solid-state switches because of their near zero power consumption with electrostatic actuation, low cost due to high-volume semiconductor manufacturing methods, and good signal isolation due to air gaps between switch contacts. Furthermore, MEMSs fill the

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

gap between SSRs and EMRs by offering true ohmic switching, desirable miniaturisation and consequently good signal properties over large bandwidths [13]. Recently, promising technologies that emerged in the form of Nano Electro-Mechanical Systems (NEMSs) [15] and Transparent Embedded Magnetic Switches (TEMSs) [16] have provided an alternative for feasible switching media.

Regardless of the actual switches, a Double-Pole-Single-Throw (DPST) switching matrix configuration, as depicted in Figure 2.3, ensures that any-to-any connectivity is achieved. In effect, two commonly controlled switches are required for each cross-point. That is, if n subscriber ports ($2n$ physical wires) and m equipment ports ($2m$ physical wires) are provided, a total of $2n \times m$ switches are required. For relatively large switches/relays, such as stepper motors, a considerable amount of space would be required to implement such a switching matrix. Moreover, as the number of incoming subscriber lines increases, the number of DPST switches increases and consequently also the number of control signal lines. In this case, it would be feasible to cascade these switching units to form a multicast switching network [14, 17]. Such an arrangement is shown in Figure 2.4.

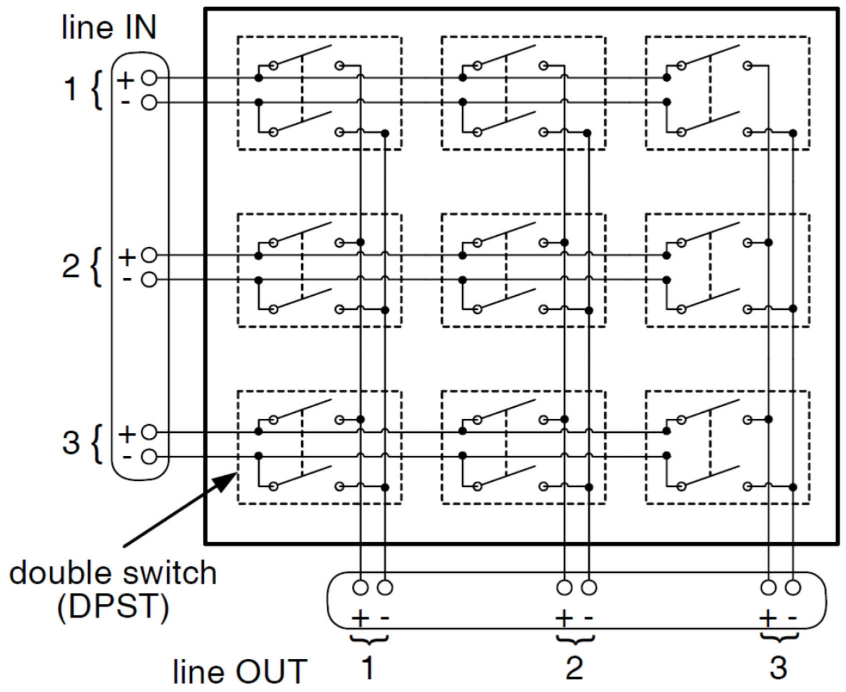


Figure 2.3: Electrical schematic diagrams of a switch unit, consisting of an array of 3×3 double-switches. Adapted from [13]. ©IEEE 2007.

Upon scrutinising Figure 2.3 and Figure 2.4, it is evident that the diagram shown in Figure 2.3 is used to implement each switching unit in the hierarchical layer in Figure 2.4. Specifically, Figure 2.4 shows how a multi-stage network is employed to establish any-to-any connections without interrupting ex-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

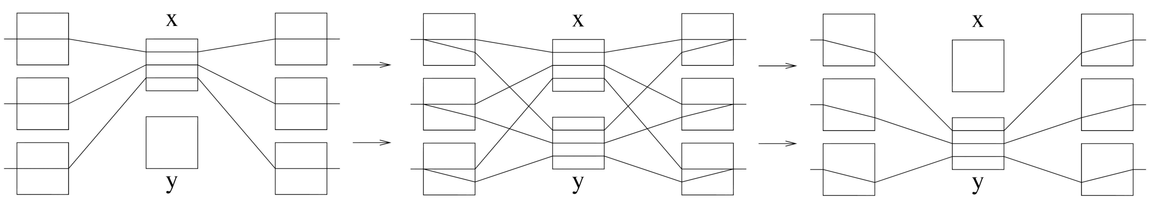


Figure 2.4: Arrangement of switch units on a higher hierarchical layer, forming a multicast switching network and allowing for the re-configuration of a larger number of lines [17]. [1]

isting connections. Various-sized MEMS switch packages for telecommunication networks have been introduced, including 2×2, 3×3, 5×5 and 20×20 arrays [13,14]. By utilising these MEMS packages, larger matrices can be created by implementing multi-stage switching networks.

2.4 REDUNDANCY AND HIGH AVAILABILITY

In order to provide reliable services to customers, the availability of the service should be of such a nature that a customer is ideally able to access the service at any time without experiencing downtimes. To this end, it is required to reduce network downtimes of service-providing equipment. Studies suggest that *the five nines of availability* or 99.999 % of network uptime is considered to be adequate [18, 19]. This translates to a mere five minutes of downtime per year. However, the result of recent market analysis (2009) shows that 50 % of subscribers expect at least 99.99 % availability [20]. Consequently, this research focused on achieving 99.99 % service availability.

2.4.1 Redundancy Strategies

There are two types of redundancy strategies, namely *standby* and *active* (*parallel*). In an active redundancy arrangement, all redundant units are simultaneously active from time $t = 0\ s$. As a result, the active redundant units are subjected to operational stresses at an early stage. As the name suggests, a standby redundancy arrangement entails redundant units being placed in standby mode when not in operation. The redundant units are used sequentially; a failover requires that a component in standby mode be switched on. A system with two standby redundant units are shown in Figure 2.5. If the primary component fails (component 1), then the switching unit immediately switches over to one of the redundant units (components 2 and 3). There are three variations of standby redundancy, namely

[1]Reprinted from International Conference on High Performance Switching and Routing, D. Cuda, P. Giaccone, and M. Montalto, Design and Control of Next Generation Distribution Frames, pp. 115-120, Copyright 2011, with permission from Elsevier.

© University of Pretoria

cold, *warm* and *hot*. In cold standby redundancy, the component does not fail before it operates, to save the component from operation stresses. In warm standby redundancy, the component is more prone to failure before operation. In hot standby redundancy, the failure pattern of the component does not depend whether the component is in standby or in operation. Cold standby means that the redundant units are powered down until needed, whereas warm and hot standby the redundant units are activated. The likelihood of failure of redundant units not in operation is very low, and is assumed to be zero in most cases [21]. Therefore, only active and cold standby redundancy are considered.

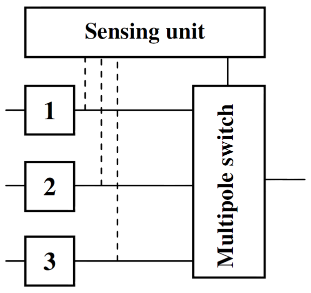


Figure 2.5: Standby system with two standby components.

2.4.2 Expected Lifetime of Systems

When modelling components in simulation, it is often required to define expected component lifetimes. These lifetimes are typically stochastically generated from statistical distributions. The normal and exponential/Weibull distributions are often used due to their mathematical tractability [22]. For a parallel or active redundant system the lifetime of the components are [23]:

$$T_i(\mathbf{x},\boldsymbol{\xi}) = \max_{1\le j\le x_i} \xi_{i,j}, \qquad i = 1,2,\ldots,n. \tag{2.1}$$

where $T_i(\mathbf{x},\boldsymbol{\xi})$ is the lifetime of component i in the decision vector $\mathbf{x}$, and $\boldsymbol{\xi}$ is the observational vector containing expected lifetimes of the redundant units. $\xi_{i,j}$ is the lifetime of redundant unit j of component i in the decision vector. x_i is the number of redundant units assigned for component i. Put simply, the component lifetime is equal to the largest lifetime of all the redundant units. The corresponding component lifetime of a standby redundant system is given by:

$$T_i(\mathbf{x},\boldsymbol{\xi}) = \sum_{j=1}^{x_i} \xi_{i,j}, \qquad i = 1,2,\ldots,n. \tag{2.2}$$

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

For a standby configuration, the component lifetime is simply the sum of the lifetimes of each of the redundant units on standby. It is clear that the expected lifetime, $E[T_i(\mathbf{x},\boldsymbol{\xi})]$, of a component will be much larger for a standby configuration. However, implementing a standby redundant system is much more complex and expensive due to the necessity to detect failures as they occur and to activate redundant units [21]. Furthermore, the inclusion of the failover mechanism adds additional failure points in the system (see Figure 2.5).

2.5 RELIABILITY

One way of increasing the reliability of a system is to incorporate redundancy scheme(s) into the design. Implementing redundancy in a design increases the expected lifetime of the system, thus increasing its reliability. During the early stages of conceptual design, the ability to predict reliability is very limited. It can be difficult to predict reliability of the concept design accurately prior to testing of a prototype in an experimental environment, or without sufficient field data. Various publications discuss analytical approaches to modelling, evaluating and optimising the reliability of a concept design [24–27].

For a pure series arrangement of n components/subsystems, the overall system reliability (R) is defined by [28]:

$$R = \prod_{i=1}^{n} R_i \tag{2.3}$$

where R_i is the reliability of component/subsystem i in the series arrangement. It is assumed that each component/subsystem fails independently, i.e. the failure events are mutually exclusive [28]. In the case of a parallel configuration, components that normally perform identical functions are placed in parallel. A failure only occurs if all the components connected in parallel have failed. The probability that the entire parallel branch would fail ($Pr\{entire\ branch\ fails\} = F$) is given by [28]:

$$F = \prod_{i=1}^{n} F_i \tag{2.4}$$

where F_i is the probability that component i would fail, or in terms of reliability, since $R = 1 - F$ [28],

$$R_p = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{2.5}$$

where R_p is the overall reliability of the parallel branch, and R_i the reliability of component/subsystem i. After the reliability, R, has been determined for the entire system, the corresponding system cost

© University of Pretoria

can be calculated by [24]:

$$C = \sum_{i=1}^{n} c_i \cdot (a_i + s_i) + c_f(1 - R) \tag{2.6}$$

where c_i is the acquisition cost per component for subsystem i,

a_i is the number of active redundant units for component/subsystem i,

s_i is the number of standby redundant units for component/subsystem i, and

c_f is the cost of mission (system) failure.

$s_i = 0$ when active redundancy is used and $a_i = 0$ when standby redundancy is used. It is often assumed that the chosen redundancy scheme for each component is known *a priori*.

2.6 REDUNDANCY ALLOCATION PROBLEM

The RAP entails the simultaneous selection of components and a system-level design configuration that can collectively meet all design constraints in order to optimise specific objective functions, such as system cost and/or reliability [29]. In general, the RAP is known to be an NP-hard problem [30] that can be approximated by metaheuristic approaches. The RAP is divided into two groups [25]:

- *RAP with Component Mixing (CM):* A mix of components is allowed for redundant units in parallel. Using different, yet functionally similar components in a redundant configuration increases the chances of correct operation.
- *RAP without CM:* A single component type is available for redundant units.

The main problem with a pure reliability optimisation problem is that the mathematics involved in describing complex systems is reasonably complex; the analytical formulation can only be obtained with significant effort and iteration through variations becomes impractical. Instead of using an analytical approach, the concept of the *system structure function* used in redundancy optimisation is considered. In a system consisting of n components, the object is to find the optimal value of $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, where x_i denotes the number of redundant units for component i. Let y_i denote the state of component i, which is determined by the states, $y_{i,j}$, of the redundant elements for $j = 1, 2, \ldots, x_i$, $i = 1, 2, \ldots, n$. A fundamental hypothesis of the system structure function is presented in [31]: For any redundant system, there is a system structure function Ψ: $\{0, 1\}^n \rightarrow \{0, 1\}$ that assigns a system state $\Psi(\mathbf{y}) \in \{0, 1\}$ for each component state $\mathbf{y} \in \{0, 1\}^n$. $\Psi(\mathbf{y}) = 0$ signifies that a component failure has resulted in an overall system failure. Using the system structure function, one can systematically progress through time and evaluate the state of each component, and subsequently, the state of the

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

overall system at any instance. This process continues until the system structure function evaluation returns zero. The expected lifetime is approximated by the time instance where the overall system state changed.

As an example, consider the bridge system depicted in Figure 2.6. By definition, the bridge system will be operational if at least one path exists between the input and the output of the system. Visual inspection of Figure 2.6 indicates that the possible paths are: 0-3, 1-4, 0-2-4 and 1-2-3. The resulting system structure function is then given by:

$$\Psi(\mathbf{y}) = \{y_0 \cdot y_3, y_1 \cdot y_4, y_1 \cdot y_2 \cdot y_3, y_0 \cdot y_2 \cdot y_4\}$$

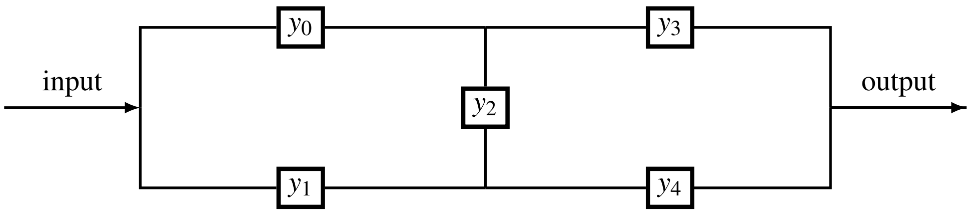


Figure 2.6: A bridge configuration.

2.6.1 Single-objective Optimisation

Evolutionary Algorithms (EAs) are of particular interest for solving the NP-hard class of problems. Popular examples of such metaheuristics used in reliability-redundancy optimisation problems include Genetic Algorithms (GAs), ant colony optimisation [27, 32] and recently, artificial bee colony algorithms [33, 34]. In single-objective optimisation, the problem is concerned with optimising a problem with respect to one objective.

2.6.2 Multi-objective Optimisation

Single-objective Optimisation is sufficient to optimise a design for a single objective, provided that other objectives are of little or no concern to the designer. As soon as multiple (and very often conflicting) objectives become paramount for a design, there is no single optimal solution, but rather an entire set of alternative solutions, each favouring a separate objective. Therefore, each solution in the set is optimal in some sense and no other solution in the search space is superior to them when all objectives are considered. The set is known as the Pareto-optimal solution set. Multi-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

dimensional search spaces are partially ordered, two solutions are related in one of two ways: either one dominates the other, or neither is dominated [35]. Formally, a multi-objective problem is typically defined as:

$$\begin{aligned} \text{Maximise} \quad & \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m)) \\ \text{where} \quad & \mathbf{x} = (x_1, \ldots, x_m) \in \mathbf{X} \\ & \mathbf{y} = (y_1, \ldots, y_n) \in \mathbf{Y} \end{aligned} \tag{2.7}$$

with an m-dimensional decision vector, $\mathbf{x}$, and n objectives in objective vector, $\mathbf{y}$. $\mathbf{X}$ is the parameter (decision-variable) space, and $\mathbf{Y}$ is the objective space.

In essence, there are two goals in multi-objective optimisation [36]:

1. To find a set of solutions as close as possible to the Pareto-optimal front (discussed in succeeding section).
2. To find a set of solutions as diverse as possible.

The first goal is essential for optimisation, and is common to the optimality goal of single-objective optimisation. The second goal, however, is specific only to multi-objective optimisation. Although it is required that the set of solutions must be close to the Pareto-optimal front, they should also be sparsely spaced in the Pareto-optimal region. A diverse set ensures a good set of trade-off solutions to choose from. In any optimisation problem, a search is performed in the decision-variable space. However, the search can be traced in the objective space and in some instances, the progress of the search in the objective space can be used to steer the search in the decision-variable space.

2.6.2.1 Dominance

It is said that decision vector $\mathbf{x}_1 \in \mathbf{X}$ dominates decision vector $\mathbf{x}_2 \in \mathbf{X}$ (written as $\mathbf{x}_1 \succ \mathbf{x}_2$) with a set K objective functions iff. [35]:

$$\begin{aligned} & f_i(\mathbf{x}_1) \geq f_i(\mathbf{x}_2) && \text{for } i = 1, 2, \ldots, K, \text{ and} \\ & f_j(\mathbf{x}_1) > f_j(\mathbf{x}_2) && \text{for at least one objective function } j, \end{aligned} \tag{2.8}$$

where $f_i(\mathbf{x}_1)$ donates the fitness of decision vector $\mathbf{x}_1$ for the ith objective. That is, $\mathbf{x}_1$ dominates $\mathbf{x}_2$ iff. it is as good as $\mathbf{x}_2$ regarding each objective, and there is at least one objective in which $\mathbf{x}_1$ is better than $\mathbf{x}_2$. For example, if solution x is better than solution y in the first objective, while solution x is worse than solution y in the second objective, it cannot be concluded that solution x dominates

© University of Pretoria

solution *y*, nor that solution y dominates solution *x*. It is customary to say that solutions *x* and *y* are non-dominated with respect to one another [36]. When both objectives are equally important, it is not possible to say which solution is better. In the end, it is expected to have a set of non-dominated solutions with respect to each other.

It is a common occurrence that multiple solutions equally satisfy one objective. In such a case it is said that one solution *weakly* dominates the other if it is better with respect to the second objective. On the other hand, if one solution *strongly* dominates another solution, it is better with respect to both objectives. Figure 2.7 illustrates strong and weak dominance between difference solutions. From the figure, it is evident that solution B weakly dominates solution A, since it is a better choice with respect to f_2. However, solution A and solution B strongly dominates solution C, since they are better choices with respect to both objectives. Globally, it is obvious that solution D dominates the entire set. Interestingly, it is noted that if one solution strongly dominates another solution, it weakly dominates the solution as well, but not vice versa (for instance, solutions A, B and D weakly dominate solution C with respect to f_1).

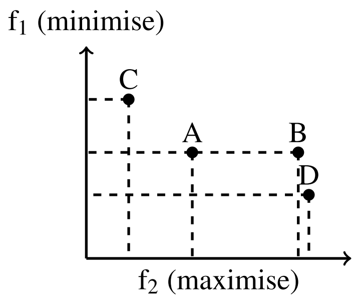
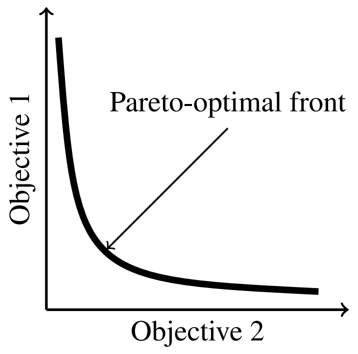


Figure 2.7: Illustration of dominance.

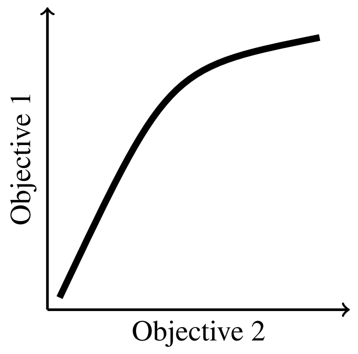
2.6.2.2 Pareto Optimality

If there is no solution vector in space $\mathbf{X}$ that dominates $\mathbf{x}$, then $\mathbf{x} \in \mathbf{X}$ is said to be *Pareto-optimal*. These solutions form the *Pareto-optimal set*. The ultimate goal of multi-objective optimisation is to identify solutions in the Pareto-optimal set $\mathbf{X}^* \subseteq \mathbf{X}$. Therefore, the Pareto-optimal set is the set of solutions that are not dominated by any other solution in the search space. For instance, in a 2D search space (two objectives), the elements of the Pareto-optimal set form a curve called the *Pareto-optimal front*, $\mathbf{Y}^* = \mathbf{f}(\mathbf{X}^*) \subseteq \mathbf{Y}$. Figure 2.8 illustrates possible Pareto-optimal fronts for problems with two objectives. The diversity of the solution set in each case is evident.

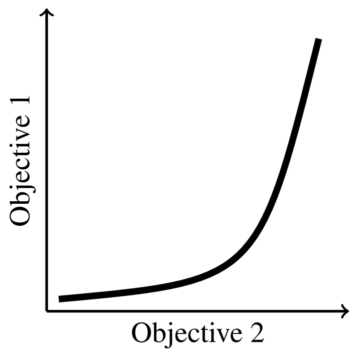
© University of Pretoria



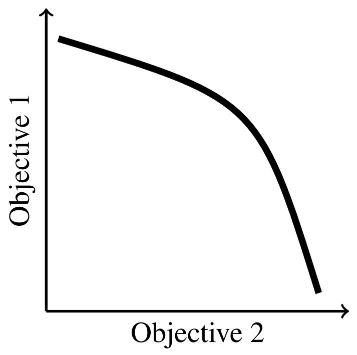
(a) Minimise-minimise problem



(b) Minimise-maximise problem



(c) Maximise-minimise problem



(d) Maximise-maximise problem

Figure 2.8: Possible Pareto-optimal fronts for a two-objective problem.

2.6.2.3 Multi-objective Evolutionary Algorithms

Because of the large search spaces encountered in Multi-objective Optimisation (MOO), GAs are very attractive options for solving MOO problems, since they offer a population-based approach. Also, multiple viable solutions make GA implementations more efficient than direct analytical approaches. Of the several Multi-objective Evolutionary Algorithms (MOEAs) that were developed, those most often represented in the literature are [37, 38]:

1. *Non-dominated Sorting Genetic Algorithm (NSGA)* [39]: Individuals (chromosomes in the populations) are classified and ranked on the basis of non-domination. All the non-dominated solutions are classified and categorised in a single category and made available for selection for the next generation. All the solutions in the first category (non-dominated) will get more copies in the next generation. This process is completed for each generation of the entire GA run. As a result the NSGA is not very efficient, since the Pareto ranking has to be repeated for every generation.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2. *Niched-Pareto Genetic Algorithm (NPGA)* [40]: The NPGA uses a tournament selection scheme based on Pareto dominance. Two random samples are selected and compared to a random subset of the entire population. If only one individual is considered to be non-dominated, and the other is not, then the non-dominated solution is passed to the next generation (including mutation and crossover operations). Any other scenario is considered to be a tie. A tie results in the tournament selection to be decided via fitness sharing. The population at the end of the run is considered to be the Pareto-optimal set.

3. *Multi-objective Genetic Algorithm (MOGA)* [41]: The rank of a possible solution is a function of the number of chromosomes by which it is dominated in a population. All the non-dominated solutions are assigned the highest fitness. This allows all non-dominated solutions to be sampled at the same rate. Fitness sharing is used to penalise dominated solutions in a certain region.

Moreover, the abovementioned MOEAs focus on simplicity of implementation. The methods are usually implemented by applying various weights to each objective function and converting it to a single-objective optimisation problem. However, a favoured approach is to use a GA along with Pareto optimality to find the global optimal solution.

2.7 EVOLUTIONARY ALGORITHMS

EAs are generic population-based metaheuristic optimisation algorithms, which use mechanisms inspired by biological evolution, such as natural selection. The working of EAs places a focus on the *survival of the fittest* principle. Candidate solutions are produced by the EA in a *population*, and the fitness of the candidate solution determines whether a solution will survive the current *generation*. During the transition to a new generation, the strongest parents are selected to hopefully produce new candidate solutions with a higher fitness. However, very large spaces of possible candidate solutions can be searched for a specific problem; it is not guaranteed that the theoretical optimal solution will ever be found, but attempts are made to find a good approximation. The most popular type of EAs is the GA.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.7.1 Genetic Algorithms

GAs are search heuristics that mimic the process of natural evolution and were proposed by J. H. Holland in 1977 [42]. In nature, it is known that weak and unfit species in a specific environment face possible extinction via natural selection. Stronger individuals survive to pass their genes on to future generations. Random changes in the make-up of the genes may occur during transitions between generations. With the correct combination of genes, the strongest will ultimately be dominant in their population. On the other hand, infeasible changes will be eliminated over time.

2.7.1.1 Terminology

In GA terminology, a solution is called an individual or *chromosome* in a solution space. The chromosomes consist of smaller, discrete units called *genes*. Typically, a gene represents the value of an element in the solution vector, which controls a certain feature of the chromosome. A GA operates on a set of chromosomes in a *population*. Each chromosome is assigned a *fitness* value that is proportional to its feasibility in the context of the problem. The first set of chromosomes in the population is called the *initial population*, which is usually randomly initialised. As the search for an optimal solution proceeds, the population consists of better (fitter) solutions. A new *generation* of the population is created by mixing and matching the chromosomes of the previous generation.

2.7.1.2 Crossover

The crossover operation is one of two mechanisms used during the transition to a new generation. During crossover, two chromosomes, called *parents*, are combined (also called recombination) to form two new chromosomes, called *children* or *offspring*. Each chromosome has a certain probability of undergoing the crossover operation, which can be set as a parameter in the GA, known as the crossover rate (P_c). Otherwise, it is simply passed to the next generation without any changes. It is desirable to select chromosomes in the population with the highest *fitness* to perform crossover. That is, the chromosomes with higher fitness should have better probabilities to be selected for crossover. In most instances, a fitter chromosome may be formed, but it is also possible to create a weaker one. Mathematically, the crossover operation is defined as follows: Consider the population with a chromosome vector set $\{V_1, V_2, V_3, \ldots, V_N\}$, where N is the size of the population. The chromosomes with higher fitness are selected more frequently and are grouped into pairs (V_1', V_2'), (V_3', V_4'), (V_5', V_6'),....

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

The crossover process is illustrated by considering the chromosome pair, (V_1', V_2'):

$$V_1' = \left(x_1^1, x_2^1, x_3^1, \ldots, x_n^1\right), \qquad V_2' = \left(x_1^2, x_2^2, x_3^2, \ldots, x_n^2\right).$$

Two randomly generated integers, n_1 and n_2, are generated such that $n_1 < n_2$. Then, the genes of V_1' and V_2' are exchanged between n_1 and n_2. The children are produced as follows [23]:

$$V_1'' = \left(x_1^1, x_2^1, \ldots, x_{n_1-1}^1, x_{n_1}^2, \ldots, x_{n_2}^2, x_{n_2+1}^1, \ldots, x_n^1\right),$$
$$V_2'' = \left(x_1^2, x_2^2, \ldots, x_{n_1-1}^2, x_{n_1}^1, \ldots, x_{n_2}^1, x_{n_2+1}^2, \ldots, x_n^2\right).$$

The children will replace their parents in the next generation. The objective function can be used to evaluate the fitness of each chromosome of the new population. This process continues for a predefined number of simulation runs, or until a desired solution is obtained.

2.7.1.3 Mutation

Mutation introduces random changes at the gene level of chromosomes. The mutation rate, P_m, is usually very small, and depends on the length of the chromosomes in the population. The process starts selecting a mutation point, and replacing the gene at that location in the chromosome with another randomly generated value. The purpose of including mutation is to ensure that the GA escapes a local extrema in the solution space to continue the search in another location. The inclusion of mutation therefore increases the chances of finding the global extrema.

2.7.1.4 Fitness

The fitness of a candidate solution is determined by the objective function, which, in turn, depends on the type of the problem. For instance, in the case of a minimisation problem candidate solutions are assigned a higher fitness for lower objective values. As mentioned earlier, the ability of a chromosome to reproduce or survive diminishes with a poor fitness. It is up to the decision maker to determine an appropriate goal function to determine chromosome fitnesses.

© University of Pretoria

2.7.1.5 Selection

The selection scheme describes how chromosomes are selected for reproduction (i.e. crossover and mutation). The general rule is that chromosomes with higher fitness are selected more frequently. As a result, weaker solutions are slowly, but surely, ruled out as possible solutions. Popular selection schemes include proportional selection, rank-based selection and tournament selection.

2.7.1.6 Elitism

Elitism is a mechanism included in GAs to ensure that the best solution(s) of a generation is not lost after reproduction. In most applications, this is accomplished by simply copying some of the best solutions of one generation to the next.

2.7.2 Hybrid Genetic Algorithms

Many approaches employ a hybrid GA to solve problems [23, 26, 43, 44]. In these hybrids, a GA incorporates one or more methods to improve the performance of the search. The advantages that hybrid algorithms exhibit include enhanced search capabilities, increasing the efficiency of the GA, improving search speed, etc. For instance, an NN can be integrated with a GA to approximate the objective values for a candidate solution and is typically used to generate an initial population. Such an approach would speed up the process of generating a population and guarantee feasible solutions are produced.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 3

THE OMNET++ SIMULATION ENVIRONMENT

3.1 CHAPTER OVERVIEW

In order to adequately evaluate the behaviour of an AMDF, a simulation model of the environment of clients interacting with the access network and the AMDF topology was developed and presented in this chapter. This chapter also touches on a number of available network simulators. A comparison between the simulators is provided and reasons for selecting the OMNeT++ environment are stipulated. A brief and compact overview of OMNeT++ is presented.

3.2 NETWORK SIMULATORS

3.2.1 Available Simulation Environments

A wide range of both commercial and free-license simulators are available, each with its own level of complexity, flexibility and modularity. A short description of the key features of the popular simulation environments are summarised in Table 3.1 (list based on [45, 46]).

3.2.2 Network Simulators Selection Process

All the packages mentioned in Table 3.1 provide support for the development of both wireless and wireline networks; however, GloMoSim currently places emphasis on wireless networks and it was not considered. Freely available simulators were given preference, and since suitable simulators were found, commercial ones were not considered further. The main functionality of Physim would not benefit the model for the purpose of the research and was therefore not considered. The remainder of the simulators are freely available for research purposes.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.1: List of available simulation environments.

Simulation environment	**Description**
OMNeT++	OMNeT++ is a C++-based Discrete Event Simulator (DES) for modelling communication networks, multiprocessors and other distributed or parallel systems introduced in 1997 [47–49]. The motivation for developing OMNeT++ was to produce a powerful open-source discrete event simulation tool that can be used by academic, educational and research-oriented commercial institutions for the simulation of computer networks and distributed or parallel systems [47]. OMNeT++ was designed from the onset to support large-scale network simulations by constructing hierarchical models built from reusable components. Strong GUI support aids in the visualisation of communication networks. More importantly, OMNeT++ also permits the simulation of circuit-based simulations (black box functionality) along with packet-based simulations.
NS-2	The NS-2 simulation environment is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of Transmission Control Protocol (TCP), routing, and multicast protocols over wired and wireless (local and satellite) networks [50]. The complete source code of NS-2 is available under the GNU public license and can be compiled on multiple platforms, including most Unix distributions and Windows.
NS-3	NS-3 is a revision of NS-2 focussing on an improved core architecture, software integration and educational components of NS-2, incorporating C++ to aid in complex simulations [51].
OPNET	The OPNET modeller [52] is a leading commercial network simulator that provides predictive modelling, which offers customers to design, deploy, manage and optimise network applications and infrastructures.
NetSim	NetSim is an educational stochastic DES development environment with source code editor and compiler. Various model libraries exist and are available for use modification [53]. NetSim provides animated basics and pictorial presentations of networks. NetSim also contains modules for real-time data packet capture.

Continued on next page

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.1 – continued from previous page

Simulation environment	**Description**
GloMoSim	GloMoSim is a scalable simulation environment for wireless and wired network systems; however, emphasis is currently placed on wireless networks [54]. It employs the parallel discrete-event simulation capability provided by Parsec (a C-based parallel simulation language). It is freely available for academic and research purposes. Commercial users must use QualNet, the commercial version of GLoMoSIm [55].
Physim	Physim is a network simulator focussing on Radio Frequency (RF) channel modelling, with built-in oscilloscope, spectrum analyser and polar plot functionalities [56]. This package is useful when investigating the effect of various frequency modulation schemes.

The main candidates were NS-2/3 and OMNeT++. Table 3.2 shows a comparison between NS-2/3 and OMNeT++. Important features that influenced the final decision included:

- language support,
- flexibility,
- documentation, support and community involvement,
- visualisation,
- programming model,
- scalability,
- event logging, and
- parallel simulation.

Table 3.2: Comparison between NS-2/3 and OMNeT++.

NS2/3	**OMNeT++**
Language support	
Dual language simulator. Relies on C++ to implement simulation models. Also supports oTcl scripts to construct network topologies (NS-2 only). NS-3 integrates Python to realise a network.	Network Description (NED) files are used to describe network topologies. Module behaviours are described in C++.

Continued on next page

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.2 – continued from previous page

NS2/3	**OMNeT++**
Flexibility	
Originally designed to be a TCP simulator. NS-2/3 models are very detailed and it is very difficult to manipulate or alter their behaviour.	Very flexible general framework. Atomic behaviour of modules can be implemented via *simple modules* which operates independently. Simple modules communicate by means of message passing. Modules can be grouped into larger *compound modules* to build hierarchical models. This modularity allows users to create virtually any type of circuit simulation.
Documentation	
Plentiful documentation and online support.	Excellent, thorough and well-structured documentation with plenty of tutorials and examples. Very active community.
Visualisation	
Provides an interactive GUI with packet-level animations and data inspection tools.	Provides an interactive GUI (TkEnv) which allows users to track messages between modules using vibrant animations. Command line representation of the simulation run is also available.
Portability	
Windows (Cygwin) and Linux platforms supported.	Windows and Linux platforms supported.
Programming model	
Mixed mode Object-oriented (OO) Tcl (OTcl) with underlying C++ classes. Network descriptions are generated by OTcl. Care has to be taken to prevent memory leaks.	OO, event-driven model written in C++. A GUI aids in the creation of network descriptions and configuration files.
Scalability	

Continued on next page

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.2 – continued from previous page

NS2/3	**OMNeT++**
Limited scalability in terms of memory usage and simulation run-time. NS-3 exhibits better scalability characteristics.	Provides excellent scalability of models and allows distribution over multiple hosts during run-time.
Result logging	
Results are logged in trace files that store timestamps and the corresponding events. The visualisation tool (nam) replays events from a trace file. Xgraph is used to draw vector plots.	Output statistics can be captured as series data or single statistics in vector and scalar files, respectively. The EventLog and Sequence Chart features allow users to view the flow of events, which greatly aid in debugging.
Parallel simulation	
Parallel simulation extension developed by the Georgia Institute of Technology, but is not in wide use.	Parallel simulation supported via Message Passing Interface (MPI).

A performance comparison conducted in [46] showed that NS-2, NS-3 and OMNeT++ are equally capable of carrying out large-scale simulation runs in an efficient way. OMNeT++ is not a network simulator by definition, but is a general DES framework. However, NS-2 is thoroughly outperformed by both OMNeT++ and NS-3 in terms of simulation run-time and memory usage, with NS-3 having the best overall performance. Nevertheless, NS-3 is still in the early stages and has not gained widespread adoption, and the rich collection of NS-2 models still needs to be ported to NS-3. OMNeT++ has a less steep learning curve and appears to be more user-friendly. In addition, OMNeT++ provides an excellent GUI and abstract modelling language, which makes it a much more attractive option.

3.3 OMNET++ INTRODUCTION

OMNeT++ is a C++-based OO modular DES of which the primary application area involves modelling communication networks, multiprocessors and other distributed systems [47]. However, it can be used to simulate almost any discrete event system. From the onset, OMNeT++ was designed to [47, 48]:

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- enable large-scale simulation, which requires simulation models to be hierarchical and to be built from reusable components,
- facilitate debugging and visualisation of simulation models to reduce design-debug cycles,
- be modular, customisable and allow embedding simulations into larger applications,
- allow data interfaces to be accessible with other commonly available software tools, and
- provide an Integrated Development Environment (IDE) that facilitates model development and the analysis of results.

The motivation for developing OMNeT++ was to produce a powerful open-source DES to be used under the Academic Public License (APL) for academic and research purposes. A license is also available for a commercial version of OMNeT++. OMNeT++ attempts to fill the gap between research-oriented simulators (such as NS-2) and expensive commercial products (such as OPNET) and is available on most platforms including Linux, Windows and Mac OS, using the GCC toolchain.

OMNeT++ presents a component-based simulation framework approach. Instead of providing hard-wired simulation components, it provides the basic mechanisms and tools to write such components, making OMNeT++ an ideal candidate for circuit-based simulations as well (along with packet-based simulations). A high-level language exclusive to OMNeT++, called NED, allows developers to define hierarchical components of a network as well as the topology of the network. A component is referred to as a *module* in the OMNeT++ environment. The behaviour of a module is written in C++ and special functions are provided to pass instructions to the network description layer. Modules communicate via message passing, which can contain complex, user-defined data structures. Messages can be sent directly to other modules or via a predetermined path or channel. Using modularity, the developer is able to build reusable modules that are defined by a set of *parameters* specific to the module. One module can inherit the properties of another module with subtle changes. All the parameters of the simulation run is configurable from a single configuration file, which is passed to the modules during initialisation of the simulation run.

3.3.1 Discrete Event Simulation

In discrete event simulation, events of the operation of a system are scheduled as a chronological sequence of events. These events induce a change of state in the system, usually at random instances

© University of Pretoria

in time. Multiple events can be scheduled at the same point in time; however, this may lead to logical complexities about the order of which the events are processed. Events are processed in strict timestamp order to maintain causality, i.e. to ensure that no current event may have an effect on earlier events.

Unlike real-time simulation, transitions in time between events are instantaneous. The simulation *clock* keeps track of simulation time, which is incremented to the timestamp where the next event is scheduled once all the events at the current timestamp have been processed. The execution cycle of a DES run can be encapsulated in a two-phase loop [57]:

1. Carry out all possible actions at the current simulated time.
2. Advance the simulation clock.

As mentioned, entities migrate between states after events are processed. In DES, an entity can assume one of five states [57]:

- *Active*: The active state is the state a moving entity presides in while migrating to another state. Only one entity can be active at any time.
- *Ready*: An entity waiting to become active, but being delayed by another active entity, is in the ready state.
- *Time-delayed*: The state of an entity waiting for a *known* simulated time stamp, before entering the ready state.
- *Condition-delayed*: The state an entity assumes when delayed by a condition until an *unknown* future time stamp.
- *Dormant*: The state that an entity cannot automatically escape from by changes in model conditions.

Entities in the ready state are placed in the Current Events List (CEL), otherwise future events are kept in the Future Events List (FEL). A transition from the FEL to the CEL is referred to as a *move*. All condition-delayed entities are placed in a delay list. By using polled waiting or related waiting, the event is transferred to the CEL if the conditions are right. Entities in the dormant state must be managed by user-managed lists. An event is removed from all lists once it exits the active state.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

3.3.2 OMNeT++ Model Structure

Refer to Figure 3.1 for a graphical depiction of the hierarchical model structure used in OMNeT++. The system module, also known as the *network*, is the top-level module of the design. The system module contains all the submodules of the network, which in turn could contain other submodules, termed *compound modules*. The network is instantiated as a compound module without gates to the outside world. How the submodules are connected within the compound module is defined by using the NED language. On the other hand, the modules at the lowest level are termed *simple modules*. Simple modules contain the C++ code to be executed when messages arrive at the *gates* of the module. The gates of simple and compound modules can be connected via a *connection*; however, connections across hierarchy levels are not permitted. Since only simple modules can react to messages, the gates of compound modules transparently relay messages to the gates of destination simple modules.

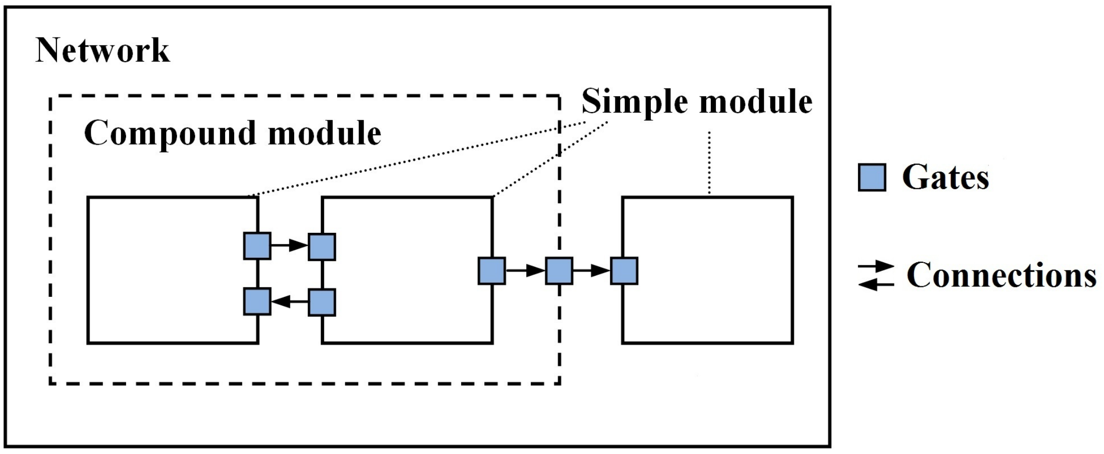


Figure 3.1: Model structure in OMNeT++.

Simple and compound models are both instances of *module types*. Multiple instances of a predefined module can be instantiated in the network. This feature is especially useful when an array of functionally identical components needs to be simulated. By means of inheritance, more complex modules can be defined from *base modules*.

3.3.3 The Simulation Library

A rich object library is implemented for simulation. Following an OO approach results in a compact and slim simulation kernel; components can be added to the kernel as they are required. Aggressive memory optimisation is implemented in the simulation kernel based on shared objects and copy-on-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

write semantics.

OMNeT++ is able to generate *random numbers* for common statistical distributions from multiple independent streams. Provision is also made for the developer to create his/her own custom distributions, which should be defined by histograms.

Provision is also made for container classes such as *queues*. These are mainly implemented to account for multiple messages arriving at module gates. Queues can be used optionally as priority queues as well.

Several statistical classes are included to collect statistical data on simulation runs. These include simple ones that collect means and standard deviations of data to a number of distribution estimation classes.

3.4 THE NED LANGUAGE

OMNeT++'s topology description language, NED, describes the physical structure of the model, modules and their interconnections. Typical ingredients are component and parameter definitions. Different component descriptions are used for the main components: simple modules, compound modules and channels. Once defined, these components can be reused in other network descriptions.

The NED language is designed to promote ease of scalability. Iterative loops (for-loops) along with conditional assignments (if-statements) aid in effortlessly defining large-scale or complex networks. NED supports partitioning of large files into smaller-sized files and reconstruction via file inclusion. The OMNeT++ IDE provides a graphical editor for generating NED files. These files can also be tweaked or hand-written from scratch via a standard text editor. The NED language has a one-to-one mapping to Extensible Markup Language (XML), that is, NED files can be converted to XML without loss of data.

3.4.1 Features of NED

The NED language exhibits the following features, which let it scale well to large projects [58]:

1. *Hierarchical*: OMNeT++ uses this traditional method to elevate the task to deal with complex-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ity. Large and complex modules are broken up into smaller entities, which greatly aids in the debugging phase.

2. *Component-based*: Simple and compound modules are reusable, which allows component libraries to be used.

3. *Interfaces*: Module and channel interfaces are placeholders for actual concrete modules. This feature is useful when the actual concrete module or channel type is determined during the network setup. That is, the actual code to be used for the module is not hardwired.

4. *Inheritance*: If desired, modules and channels can be subclassed. Inherited components derive their properties from base classes and may add extra parameters, gates, and in the case of compound modules, connections and submodules. A derived class may also overwrite the values of existing parameters.

5. *Packages*: A packet structure is included to reduce the risk of name clashes.

6. *Inner types*: Modules and connection types are used locally within a compound module when defined within the compound module.

7. *Metadata annotations*: Extra information (metadata) can be annotated in the code without being used by the simulation kernel. Annotations can be added to parameters, gates, channels and submodules by adding properties. For instance, the measurement unit (Watts, etc.) can be specified as metadata annotations.

3.4.2 Network Definition

A network is the top-level entity in the OMNeT++ simulation environment. The network, a compound module itself, defines all sub-top-level simple and compound modules and its interconnections. The compound modules, in turn, hold the definitions of any low-level entities. The NED description of a standard network is:

© University of Pretoria

```
// A network example
//
network Network{
parameters: // Example parameters
    param1 = 10;
    param2 = true;
    param3 = normal(100,10);
    ...
submodules: // Example submodules
    node1: type1;
    node2: type2;
    node3: type3;
    ...
connections: // Example connections
    node1.gate1 <--> node2.gate1;
    node2.gate2 --> node3.gate1;
    node3.gate3 <-- node1.gate3;
    ...}
```

As mentioned earlier, parameters can take on a variety of data types. Submodules are named and defined by the type of module (the name provided for the simple or compound modules). Unidirectional connections are denoted by `-->` and `<--`, whereas a bidirectional connection is depicted by `<-->`. Only one connection is allowed per gate.

3.4.3 Gates

Gates are the connection points of modules. Three types of gates exist, namely *input*, *output* and *inout* gates. Unidirectional connections are connected to input and output gates and bidirectional connection are used for inout gates. Nevertheless, unidirectional connections are also allowed on inout gates. A gate can be defined as a single gate or a gate vector. The definition of a vector is akin to the definition of an array in C++. Square brackets are added, with either a predetermined or an open size. One is able to dynamically add gates to a vector by using the increment operator (e.g. gate++) if no size is defined. The current size of a gate vector is queried by `sizeof()` operator in C++. Messages are sent/received at gates via the `sendDirect()` method; however, if it is desired to send messages directly to a specific gate, a gate should be annotated with `@directIn`.

3.4.4 Simple Module Definition

Simple modules are basic atomic components of a network or compound module. A simple module is defined with the `simple` keyword as follows:

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```
// A simple module example
//
simple moduleName{
parameters: // Example parameters
    param1 = 10 ms;
    ...
gates: // Example gates
    input in;
    output out;
    inout both;
    ...}
```

3.4.5 Compound Module Definition

Compound modules contain one or more submodules to group them into a larger unit. Akin to simple modules, a compound module has gates and parameters, which can be passed to submodules. However, compound modules do not exhibit active behaviour. A compound module is defined by the `module` keyword as follows:

```
// A compound module example
//
module moduleName{
types:
parameters: // Example parameters
    param1 = 10 ms;
    ...
gates: // Example gates
    input in;
    output out;
    inout both;
    ...
connections: // Example connections
    node1.gate1 <--> node2.gate1;
    node2.gate2 --> node3.gate1;
    node3.gate3 <-- node1.gate3;
    ...
submodules: // Example gates
    node1 : type1;
    node2 : type2;
    ...}
```

All the modules contained in the compound module are listed in the `submodules` section, similar to network definitions. The internal interconnections are also listed under connections. Any modules and channel to be used locally are defined in the `types` section as inner types. Via subclassing, compound modules can be extended using the `extend` keyword. The extended module inherits all the properties of the base module, as well as additional submodules, parameters, or gate definitions

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

in the extended module, for example:

```
// An extended compound module example
//
module extendedModuleName extends moduleName{
parameters: // Example additional parameters
     param2 = 30 Mbps;
     ...
gates: // Example additional gates
     inout both2;
     ...
connections:
// Example additional connections
     node3.gate1 --> both2;
     node3.gate2 <-- node1.gate2;
     ...
submodules: // Example additional gates
     node3 : type3;
     ...}
```

3.4.6 Channel Definition

Channels are introduced to create a reusable connection with predefined properties. In essence, channels are simple modules, with C++-classes behind them. A channel can be defined within a network to reduce namespace pollution. A channel is defined in the *type* section of the network definition:

```
// A network with a channel type
//
network Network{
types:
     channel C {
     delay = 100 ms;
     datarate = 100 Mbps;
     };
     ...
connections:
     node1.gate1 --> C --> node2.gate2;
     ...}
```

Alternatively, a custom channel can be extended from any existing type:

```
// An Extended channel
//
channel C extends ned.DataRateChannel{
     distance = 200 km;
}
```

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Three predetermined types of channels are available to the user. These are listed below:

- `IdealChannel`: A channel with no parameters. An `IdealChannel` simply relays messages without any effect. Any connections without a specified channel is assumed to be an `IdealChannel`.
- `DelayChannel`: This channel introduces a propagation delay. Parameters:
  - `delay`: A double value to indicate the delay of a channel. The value must be followed by a time unit (`s`, `ms`, etc.).
  - `disabled`: A boolean value. If true, all messages sent through this channel are dropped.
- `DataRateChannel`: This channel introduces the datarate of the channel.
  - `datarate`: A double value indicating the datarate of the channel, needs to be specified in bits per second or its multiples as a unit (`Kbps`, `Mbps`, etc.).
  - `ber`: The bit error rate (ber) is a double value parameter in [0, 1]. The channel randomly decides on when an error occurs and sets an error flag in the packet object.
  - `per`: The packet error rate (per) is a double value parameter in [0, 1].

3.4.7 Configuration File

A single configuration file allows the user to define all the required parameters in the simulation model. During the initialisation process, the assigned values are copied to the parameters in all the defined modules and channels. The simulation environmental variables are also defined in this file. Some examples include maximum simulation time, maximum CPU time, time scale exponent, GUI or command-line interface, allowable stack usage, and partitioning information.

3.5 SIMULATION CONCEPTS

3.5.1 The Event Loop

The DES maintains the set of future events in the FEL or Future Events Set (FES). The following pseudocode describes the event loop [58]:

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```
initialise: includes building the model and initialising events in FEL
while (FEL not empty)
{
    retrieve first event from FEL, place in CEL
    t:= timestamp of this event (simulation time)
    process event: new events in CEL or delete processed ones
}
finish simulation (write statistical results, etc.)
```

The initialisation phase entails the data structures representing the model being built for simulation. Initial events are scheduled and the initialisation code for each module is executed. The loop extracts events from the CEL and processes them in strict timestamp order to maintain causality. The simulation is terminated once all the scheduled events have been processed, the CPU time has reached a given limit or until statistics collected during the simulation run have reached a desired accuracy.

3.5.2 Events

Events are represented by messages in the OMNeT++ environment. Each event is represented by the `cMessage` class or one of its subclasses. A message is sent from one module (the source module) to another module (destination module). Once a message arrives at the destination module, an event occurs at the arrival time at the module. The event of timeouts, or timers, are generated by the module by sending a message to itself. A scheduling priority can be assigned to messages in order to account for messages arriving simultaneously; the one with the smaller integer value is executed first. Otherwise, the message with the earliest arrival time is executed first.

3.5.3 Simulation Time

The simulated time is represented by a 64-bit integer, using a decimal fixed-point representation type, named `simtime_t`. The resolution is determined by the user by setting the appropriate scale exponent. The exponent can vary between zero and -18 in multiples of three. Table 3.3 shows the possible resolutions.

The `SimTime` class performs mathematical operations as 64-bit operations on `simtime_t` variables. Provision is made to account for integer overflow and any other error will be reported and

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.3: Simulation time resolutions

Exponent	Resolution	Time range
-18	1 as	±9.22 s
-15	1 fs	±153.72 minutes
-12	1 ps	±108.57 days
-9	1 ns	±292.27 years
-6	1 µs	±292271 years
-3	1 ms	±2.9227e8 years
0	1 s	±2.9227e11 years

halt the simulation. The `SimTime` class provides methods to convert the 64-bit value to `double` and `float` types. Various other methods exist that return the simulation time as a string, the current global scale and resolution, maximum possible simulation time and a number of other conversions.

3.6 RANDOM NUMBER GENERATION

Ideally, the random numbers generated during simulation runs should be unknown and unpredictable. These numbers are generated via algorithms that repeat the numbers after a certain period. To account for randomness, these algorithms take a *seed* value and commence with deterministic calculations to produce a random number. The next seed is also calculated. These algorithms are known as Random Number Generators (RNGs), or Pseudo RNGs (PRNGs).

If provided with the same seed, RNGs always produce the same sequence of numbers. This feature is of particular importance, since it makes simulation runs repeatable, or on the the other hand truly random. RNGs produce uniformly distributed integers in a specified range, usually [1, 2^{32}]. Mathematical transformations are employed to produce variates that conform to some distribution. Table 3.4 summarises supported distributions. It is also possible to specify custom distributions by providing the corresponding histograms.

The following RNGs can be selected in the configuration file:

- *Mersenne Twister RNG*: The default RNG used by OMNeT++ is Mersenne Twister (MT). MT has a period of 2^{19937} - 1 and is very fast; at least as fast as ANSI C's `rand()`.
- *Minimal standard RNG*: This RNG uses linear congruential generator (LCG) with a cycle

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.4: Summary of available distributions in OMNeT++.

Function	Description
Continuous distributions	
`uniform(a, b)`	Uniform distribution in the range [a, b)
`exponential(mean)`	Exponential distribution with the given mean
`normal(mean, stddev)`	Normal distribution with the given mean and standard deviation
`truncnormal(mean, stddev)`	Normal distribution truncated to non-negative values
`gamma_d(alpha, beta)`	Gamma distribution with parameters alpha > 0, beta > 0
`beta(alpha1, alpha2)`	Beta distribution with parameters alpha1 > 0, alpha2 > 0
`erlang_k(k, mean)`	Erlang distribution with k > 0 phases and the given mean
`chi_square(k)`	Chi-square distribution with k > 0 degrees of freedom
`student_t(i)`	Student-t distribution with i > 0 degrees of freedom
`cauchy(a, b)`	Cauchy distribution with parameters a, b where b > 0
`triang(a, b, c)`	Triangular distribution with parameters a $\leq$ b $\leq$ c, a $\neq$ c
`lognormal(m, s)`	Lognormal distribution with mean m and variance s > 0
`weibull(a, b)`	Weibull distribution with parameters a > 0, b > 0
`pareto_shifted(a, b, c)`	Generalised Pareto distribution with parameters a, b and shift c
Discrete distributions	
`intuniform(a, b)`	Uniform integer in [a, b]
`bernoulli(p)`	Result of a Bernoulli trial with probability 0 $\leq$ p $\leq$ 1
`binomial(n, p)`	Binomial distribution with parameters n $\geq$ 0 and 0 <= p $\leq$ 1
`geometric(p)`	Geometric distribution with parameter 0 $\leq$ p $\leq$ 1
`negbinomial(n, p)`	Negative binomial distribution with parameters n > 0 and 0 $\leq$ p $\leq$ 1
`poisson(lambda)`	Poisson distribution with parameter lambda

length of 2^{31} - 2. This RNG is only suitable for small-scale simulation models owing to the small cycle length, especially on fast computers.

- *Akaroa RNG*: An external RNG available from the Akaroa library.
- *Others*: OMNeT++ allows the plugging in of custom RNGs.

© University of Pretoria

3.7 MESSAGES

Messages represent events in OMNeT++ simulation environment. Messages are defined by the `cMessage` class. The `cMessage` class contains several fields in its data structure, for use by the simulation kernel and for convenience for the user. The contents of these fields are typically extracted when processing a message. Some of the important fields are listed below:

- `name`: This is a string data field, containing the optional name of a message. It is especially useful to name messages for debugging purposes.
- `kind`: An integer field used to categorise a message.
- `scheduling priority`: This integer field determines the delivery order of the current message, should it arrive simultaneously with other messages.
- `send time`: The time at which the message was sent by the source module.
- `arrival time`: The time at which the message arrives at the destination module.

OMNeT++ provides functionality to add additional fields to the `cMessage` class easily. A new message class is defined, which includes the new defined data fields. The new message name is used to define a message variable instead of `cMessage`. A subclass of `cMessage`, named `cPacket`, adds extra data fields to the message, which is useful for computer network or any other packet-based simulation. Since the primary concern of this research is to simulate circuits rather than packets, the `cPacket` class variation will not be discussed. A message is defined by:

```
// OMNeT++ custom message
//
message messageName{
    int newField1;
    long newField2 = 123456;
    string newField3;}
```

The above definition generates a class for `messageName` with the standard `cMessage` methods, as well as getter and setter methods for the new data fields. Therefore, the programmer can easily access the new data fields when processing the message. Data fields are also not limited to scalar types, but fixed-sized and dynamic arrays are permitted. More complex data structures (using ANSI C's `struct`) can be defined if complex data structures are required. The convention when using

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

OMNeT++ is to define a new message in the scope of the simulation kernel and to send a reference (pointer) of the message between modules.

3.8 SIMPLE MODULES

Simple modules are the active atomic elements in the OMNeT++ environment. The active behaviour is programmed in C++, using the OMNeT++ simulation class library. The functionality is implemented by subclassing the `cSimpleModule` class with virtual members to allow the programmer to redefine functions to implement the required behaviour. By default, OMNeT++ searches for a C++ class of the same name as the simple module type in the NED language. If this is not the case, the programmer is able to explicitly identify an alternative class to account for the behaviour via the `@class` annotation.

The `cSimpleModule` class has four virtual member functions that interact with the simulation kernel. These are intended to be redefined by the actual module class of instantiated modules. These members are listed below:

- `initialize()`: This method is invoked after the layout of the network has been completed. Variables are usually initialised at this stage by retrieving parameter values from the NED descriptions.
- `handleMessage(cMessage *msg)`: This method accepts the pointer to the message that caused the event and performs the processing of the event. The method returns control to the simulation kernel immediately after it completes. Simulation time never elapses during the processing of `handleMessage()`.
- `activity()`: This method is launched as a coroutine during network setup along with the simulation kernel. Blocking functions, such as `receive()` or `wait()`, are used to suspend the routine until a message arrives. Simulation time elapses with the blocking functions. The routine lasts as long as the entire simulation run.
- `finalise()`: This method is called upon successful completion or user termination of the simulation run. The recommended use is to record statistics of the simulation run (collected during the simulation run).

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

3.8.1 Message Handling

Functionality is implemented via one of two programming models: *coroutine-based* and *event-processing functions*. In coroutine-based programming, the module code runs in its own non-preemptively scheduled thread. Control is received from the simulation kernel once an event is issued. The `activity()` method is used when following a coroutine-based approach. The function typically never returns. When using event-processing programming, the kernel simply invokes the required module object function and returns immediately after processing. The `handleMessage()` method is used in this approach.

The main disadvantage of using a coroutine-based approach, is that each module object to which functionality is added via `activity()` requires its own CPU stack. This results in hefty memory requirements for large models and scalability is therefore limited. Switching overhead is also introduced, increasing the total simulation time. The pros of using a coroutine-based approach is that no initialisation is required (no call to `initialize()`, and it is a natural programming model for most experienced programmers. On the other hand, event-processing allows faster switching between simulation kernel and function calls, and consumes much less memory. However, all local variables requires to be initialised in `initialize()`. Generally, `handleMessage` is favoured over `activity`.

3.8.2 Message Passing

The most frequent task of simple modules is message passing and reception. Messages can be sent via gates to a predetermined path, or directly to another gate. A message can also be delayed or scheduled at a certain timestamp for delayed sending of a message. A scheduled message can be cancelled at any time. Table 3.5 summarises the functions that can be used with each programming model.

3.8.3 Parameters

Parameters defined in the NED descriptions are represented by the `cPar` class. The `par(..)` family methods are used to access the value of a parameter in C++ code. It can be used to store the value in local variables or used directly in mathematical operations. The values of the parameters can also be changed during runtime, although this is rarely needed. Overloaded assignments operations exist for various types, including `long`, `double`, `boolean`, `int`, etc.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 3.5: Message handling and passing functions

`handleMessage(..)`	**`activity()`**
• `send(..)`: A family of functions used to send messages to other modules via one of the gates of the source module. • `sendDirect(..)`: Used to send messages directly to one of the gates of the destination module. • `scheduleAt(..)`: Schedule an event at a certain timestamp. • `cancelEvent(..)`: Delete an event scheduled by `scheduleAt()`. • `sendDelayed(..)`: Delay the sending of a message.	• `receive(..)`: Block module execution until a message arrives. • `wait(..)`: Suspend the modules for some time (virtual time) • `send(..)`: A family of functions used to send messages to other modules via one of the gates of the source module. • `sendDirect(..)`: Used to send messages directly to one of the gates of the destination module. • `scheduleAt(..)`: Schedule an event at a certain timestamp. • `cancelEvent(..)`: Delete events scheduled by `scheduleAt()`. • `end()`: Used to finish execution of this module. No parameters. • `sendDelayed(..)`: Delay the sending of a message.

Modules may be required to respond to certain parameter changes in the simulation model. To account for this, the `handleParameterChange()` method can be redefined to handle parameter changes. This method is called by the simulation kernel whenever any parameter is altered, and control is returned immediately.

3.9 COMPOUND MODULES

As discussed earlier, compound modules are not active elements in the simulation model. The intention of defining compound modules is purely to act as a hierarchical interface for underlying simple modules. Compound modules are discussed in more detail in subsection 3.4.5.

3.10 PARALLEL SIMULATION

As cluster computing is becoming a norm in high-intensity computing, the need for the ability to distribute the workload over multiple processors arises. Because of the nature of large-scale simulation models, OMNeT++ has taken great care to incorporate the option of parallel distributed simulations.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

OMNeT++ permits the simulation model to be distributed over multiple CPU cores using the MPI standard. Communication can also be implemented as a file system based mechanism, using text files in a shared directory. This is especially useful when multi-node clusters are considered and network communication and synchronisation are required between local processes.

For parallel simulation, the OMNeT++ model is partitioned into several local processes. Each local process is executed independently on a different host or processor, maintaining its own simulation time and FELs and CELs. The main issue is to synchronise the local processes in order to maintain causality of events, i.e. to ensure that a message arrives at another local process on the timestamp it was intended. OMNeT++ employs synchronisation using the Null Message Algorithm (NMA) with link delays as lookahead [59]. One great advantage of this approach is that the serial model can easily be converted to a parallel model with very little or no modification. The main disadvantage is that message overhead is created as inter-process messages are transmitted. The constraints of a parallel model are the following [59]:

- Modules communicate via message passing only (no direct method calls).
- Global variables are not permitted (standard for distributed memory parallel computation).
- There are some limitations on direct message sending.
- Lookahead must be incorporated in the form of link delays.

This functionality may significantly speed up the simulation execution time if the correct partitioning is employed, especially if most modules are implemented as coroutines. Unfortunately, the ideal partitioning usually has to be determined through trial and error. The ideal partitioning of the presented simulation model was investigated and is presented in the following chapter.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 4

SIMULATING AMDF BEHAVIOUR

4.1 CHAPTER OVERVIEW

This chapter presents the model of an AMDF constructed in the OMNeT++ environment. The purpose of this model is to evaluate the dynamic behaviour of an AMDF in an exchange environment and to explore possible equipment card redundancy scenarios. The architecture and the modules are described in detail. The hierarchical structure of the model is shown in Figure 4.1.

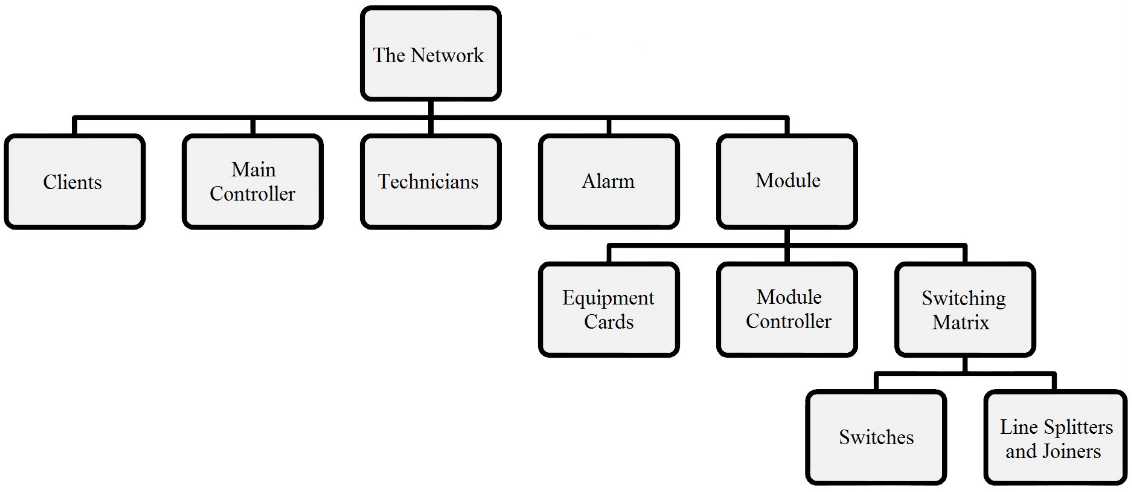


Figure 4.1: Hierarchical structure of the model.

4.2 THE NETWORK

The AMDF model consists of several million modules (including clients, controllers and individual switches) when tens of thousands of clients are simulated. A hierarchical model is used to employ modularity, which allows the user to reflect the logical structure of the actual system in the model structure. Figure 4.2 depicts an example of the logical structure in an abstract diagram.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

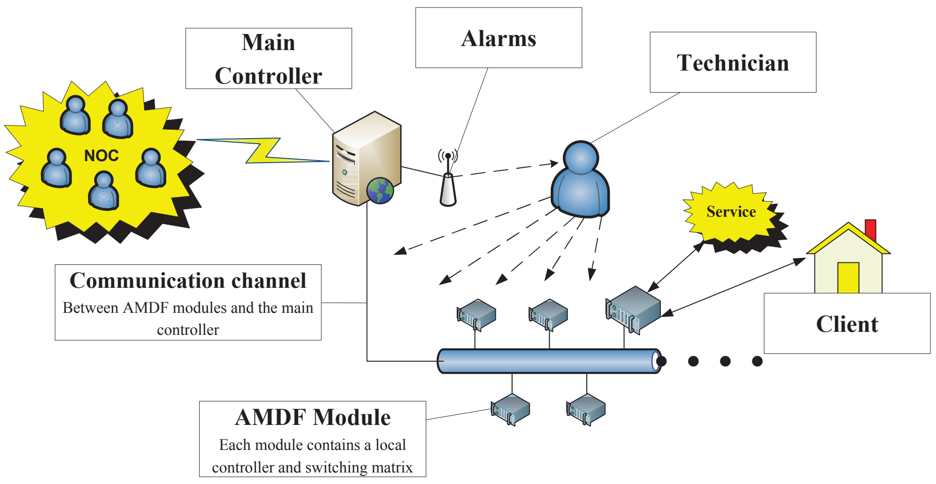

Main Controller
Alarms
Technician
NOC
Service
Communication channel
Between AMDF modules and the main controller
Client
AMDF Module
Each module contains a local controller and switching matrix

Figure 4.2: Abstract diagram of an AMDF [60].

The top-level entity is a translation of the diagram in Figure 4.2. It represents the environment of clients requesting service activations, discontinuations or service changes. The user specifies via the configuration file a number of parameters used in the model, such as the total number of clients, clients per AMDF module, equipment card lifetimes, etc. Table 4.1 lists all the parameters that can be altered for a simulation run. The model automatically builds a modular AMDF based on the provided input parameters.

For demonstration purposes, a small modular AMDF with 15 clients, constructed using OMNeT++, is depicted in Figure 4.3 to show the logical structure. The modules and its underlying submodules are discussed in detail in the succeeding sections.

4.3 SWITCHES AND SWITCHING MATRIX

The switches are modelled as simple entities that merely pass messages received at their input gates when they assume the *CLOSED* state; otherwise, the messages are dropped. For visualisation purposes, the switches appear red for open switches, and green for closed switches. Figure 4.4 displays an example of a switching matrix containing an arbitrary switching configuration. The line splitters and joiners (grey boxes) are implemented purely for simulation purposes, since multiple connections are not permitted in OMNeT++. Splitters and joiners are programmed to relay messages arriving at their gates to the correct switch immediately. The information regarding the intended destination switch is encoded in the message itself. The actual matrix is implemented as an any-to-any non-blocking

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 4.1: List of simulation parameters.

Parameter/Feature	Description
Clients	
`totalClients`	Total number of clients to be simulated / capacity of the module AMDF.
`numClientsPerModule`	Number of clients per AMDF module. Determines number of AMDF modules.
`equipmentPerClient`	Number of possible services allowed per client, e.g. ADSL, POTS, etc.
CO equipment	
`cardLifetime`	Configurable random expected lifetime of equipment cards, sampled from any statistical distribution.
`redundancyPercentage`	Redundancy percentage of equipment cards in the CO.
Technician	
`visit`	Configurable random expected time technician arrives after an alarm was issued, sampled from any statistical distribution.
`averageTimePerFix`	Configurable random expected time to replace faulty hardware by technician, sampled from any statistical distribution.
Parallel partitioning	
`sDelay`	Link delays for inter-partition communication.
`Partition-id`	Each module must be assigned to one, and only one partition. Partitions are specified by a certain partition ID, which is assigned to each module in the model.

cross-connect matrix. Any input (client) line can be connected to any output (equipment) at any instance. Each switch is independently controlled by the AMDF module controller (local controller), which in turn receives instructions from the master controller.

As far as redundancy is concerned, Figure 4.4 shows a configuration of an AMDF module for five clients. Note that there are six equipment cards for each component available. This is due to the 20 % redudancy percentage that was defined for this model; 20 % redudancy for five clients translates to one redundant equipment card available for each service to each AMDF module serving five clients.

4.4 MAIN CONTROLLER

The main controller receives requests from the call centre once the Network Operations Centre (NOC) has received client requests. The main controller contains a record of all client-service connections and is updated accordingly during client requests. The appropriate instructions are passed to the local AMDF controller of the corresponding AMDF to which the client in question is connected. The

© University of Pretoria

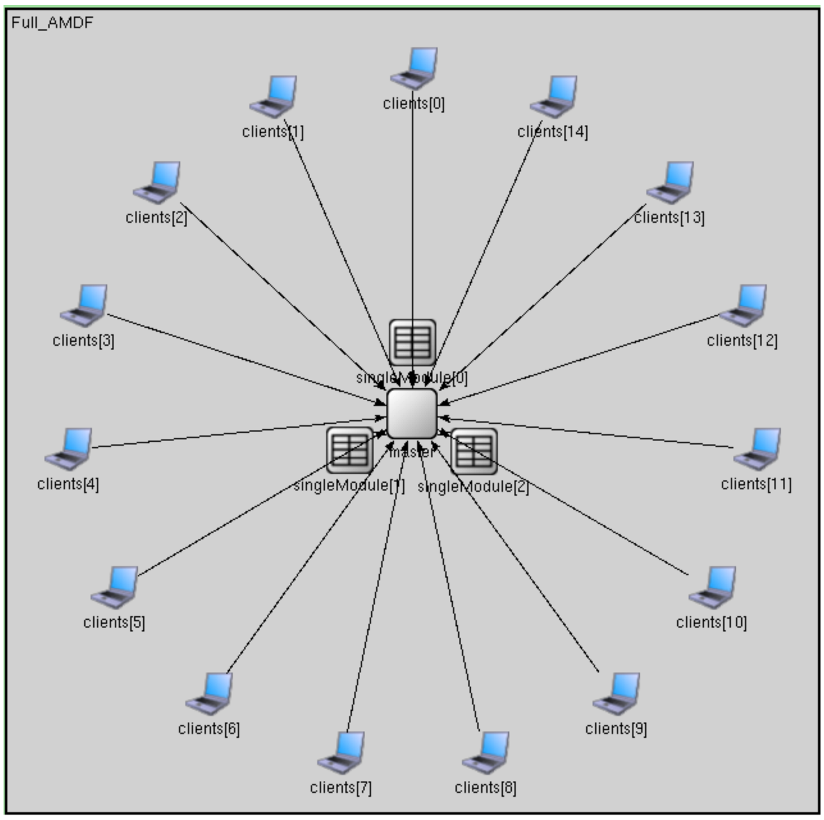


Figure 4.3: A full modular AMDF constructed in OMNeT++. A total number of 15 clients are simulated, with five clients per AMDF module.

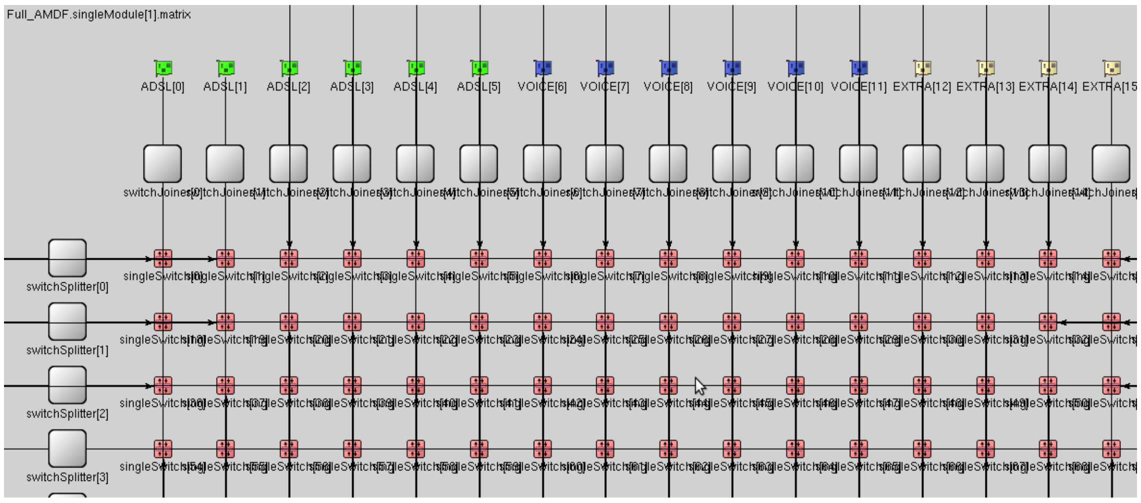


Figure 4.4: An extract of a switching matrix with an arbitrary switching configuration as generated by the OMNET++ model.

main controller does not have the ability to control the switches in the switching matrix directly. The main controller is employed as an event-processing function. No parameters are defined for the main controller.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

4.5 MODULE CONTROLLER

The local module controller is in direct control of the switching matrix. Once all the details of the client and his/her request are retrieved, the corresponding and affected switch number is obtained from the local database and the appropriate action is performed. This controller is also responsible for all fault detection and correction. If one of the CO equipment cards fails (i.e. exceeds its expected lifetime), the controller takes all the necessary steps required to switch the affected client to one of the redundant equipment cards, given that the total capacity of the AMDF module is not exceeded. The connection to the faulty equipment is severed until a technician arrives to attend to the problem or replace the card. Statistics are collected during the simulation run, which includes total and average equipment failures per time interval, the total average duration of technician visits and repair times, the total and average downtime experienced by clients, and the number of times (and total time) the system exceeded its maximum switching capacity for given redundancies. The actual number of AMDF modules is determined by dividing the `totalClients` parameter by the `numClientsPerModule` parameter. The user can readily add new statistics to be recorded for whatever purpose. The module controller is also implemented as an event-processing function.

4.6 CLIENTS

The total number of clients is defined in the configuration file in `totalClients`. Clients are instantiated as coroutines for ease of implementation. The clients are used to feed the model with requests for service activations, discontinuations or changes. Utilising the built-in RNGs and statistical distribution functions of OMNeT++, a client randomly issues requests with some probability. These probabilities can be configured beforehand. The simulation routinely runs through the entire array of clients, and depending on the current state of the client, the client enters another state with some probability. As a result, the array of clients takes up most of the processing time. A client can assume one of five states:

- *CONNECTED*: The state a client assumes when connected to one of the equipment cards.
- *DISCONNECTED*: The state a client assumes when not connected to any equipment card.
- *REQUESTNEWSERVICE*: A client enters this state when requesting a new service while in the *DISCONNECTED* state.

© University of Pretoria

- *DISCONTINUESERVICE*: A client enters this state when cancelling the current service(s) while in the *CONNECTED* state.
- *CHANGESERVICE*: A client enters this state when requesting a new service after cancelling a service while in the *CONNECTED* state.

The statistical frequencies of client requests are configurable to allow client profiles to be created in order to explore typical scenarios. For example, one might be interested in the behaviour of an AMDF during special events when a temporary increase in communication traffic is observed. Such an event could include a World Cup tournament or the Olympic Games.

4.7 EQUIPMENT CARDS

During initialisation, each equipment card is assigned a random expected lifetime based on the statistical distribution selected for the `cardLifetime` parameter. The actual number of equipment cards is determined by the number of clients per AMDF module, `numClients`, and the redundancy percentage, `redundancyPercentage`, and the number of services, `equipmentPerClient`. Thus, the total number of equipment cards is determined by `numClients` $\times$ `equipmentPerClient` $\times(1+$`redundancyPercentage`$)$.

An equipment card can assume one of two states:

- *OPERATIONAL*: The state the equipment card assumes when operational. For visualisation purposes, an operational card's colour is unique to the service.
- *FAULTY*: The state the equipment card assumes when faulty. For visualisation purposes, a faulty card's colour temporarily turns to red and returns to the service colour when operational.

4.8 ALARMS

Equipment cards are assigned random lifetimes with the built-in statistical distribution functions. When an equipment card exceeds its assigned expected lifetime, the equipment card assumes the *FAULTY* state. Subsequently, an event is scheduled when the alarm alerts the main controller (containing a record of all client-service connections) of the failure. The main controller, in turn, informs the corresponding module controller of the fault, and the module controller performs the necessary

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

steps to rectify the problem. At the same instance, the main controller schedules an event for a technician call-out based on the `visit` parameter. The alarm has two possible states: *ON* and *OFF*. Once the technician arrives, the alarm is deactivated.

4.9 TECHNICIAN

Once a call-out is made, a technician is dispatched to the AMDF. Upon arrival, a message is immediately sent to the alarm to be deactivated (*OFF* state). Depending on the number of faulty pieces of equipment, an event is scheduled at a time proportional to the `averageTimePerFix` parameter to inform the main controller when the replacement of all the faulty hardware is complete. At this stage, the module controller switches all the affected clients back to the replaced cards.

4.10 CUSTOM MESSAGES

Three custom messages with custom data fields were defined for use in the AMDF model. These data fields contain additional information regarding the operation of the model. The three messages are: `clientRequest`, `fault`, `switchControlMessage`. Table 4.2 lists and discusses all the messages used in the model.

4.11 PARALLEL DISTRIBUTED EVENT SIMULATION EVALUATION

In order to evaluate the effectiveness of the implementation of the simulation model, the features programmed into the model adhering to the requirements in section 2.2 had to be assessed. It was hypothesised that a potential problem with the current model would be scalability and Parallel Distributed Event Simulation (PDES). A run of the serial model confirmed that a large number of clients (over 10,000) did not scale well with a single CPU core, resulting in very large simulation execution times. The optimal partitioning of the model and process distribution had to be investigated. In view of the number of clients for even a small-sized AMDF, a logical approach would be to divide the clients evenly over the available processes. To aid in the assessment of the effectiveness of a proposed parallel model, a criterion was developed in [61], which was used to evaluate the potential parallelism of the serial model.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 4.2: Summary of custom messages used in AMDF model.

Source modules	Destination modules	Fields
	`clientRequest`	

`clients`
`main controller`
	`main controller`
	`module controller`
		• `name`: Name of the client for debugging purposes.
		• `surname`: Surname of the client for debugging purposes.
		• `ID`: ID of the client module.
		• `service`: An enumerated type signifying the requested service to be activated or discontinued.
		• `oldService`: Current service of the client to be discontinued during change request.
		• `request`: An enumerated type used to determine the action to be performed based on the client's request.

	`fault`	
`card`	`main controller`	• `cardIndex`: Index of the card in the equipment card array. • `status`: Operational state of the card. • `oldSwitch`: Index of the old switch during a change (when client requests service change). • `cardName`: Name of the card for debugging purposes.
	`switchControlMessage`	

`main controller`
`local controller`
	`local controller`
	`switch`
		• `command`: An enumerated command sent to open or close a switch.

As a benchmark, an unpartitioned model was run for 3200 clients, 100 clients per module, 20 % redundancy in express mode[1]. The goal was to determine the optimal set of inter-partition link delays and number of CPU cores to maximise the execution speedup.

4.11.1 Feasibility Criterion

The events/sec (*P*) and events/simsec (*E*) values were obtained with P = 437,305 and E = 320. Note that *P* and *E* are fixed properties of the serial model, and cannot be changed during runtime. To

[1]Fastest possible execution, non-debug mode with no visualisations and minimal status updates about the progress

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

increase the effectiveness of the PDES implementation, the link delays between partitions should be maximised, since the null message protocol in MPI uses link delays as lookahead (L). L should be extremely large compared to $1/E$ ($L \gg 1/E$) and MPI end-to-end latency (τ). A cluster typically exhibits higher values for τ, since the actual speed of the switch limits its performance. However, high-speed interconnections used in most clusters exhibit values ranging from 5 µs to 30 µs, typically 20 µs [61]. The number of partitions is denoted by n. The coupling factor, λ, is considered:

$$\lambda = \frac{L \times E}{\tau \times P \times n} \tag{4.1}$$

It follows from the definition that if $\lambda < 1$, frequent blocking is guaranteed and good performance in terms of execution times from the simulation cannot be expected. Ideally, λ should be greater than 10.0. A typical plot for λ as a function of the number of cores and partitions is shown in Figure 4.5. It is evident that favourable values of λ are for a lower number of cores and larger link delays. The effect of the NMA overhead is apparent for a larger number of cores. The limitation, however, is apparent when selecting the appropriate link delays. If the value of L is too high, the frequency of the events to be generated between local processes will drop, resulting in a lower value for E. In addition, the simulation model may not be a true reflection of the actual model if link delays are unrealistically high between various modules. For this reason, the values for L was unrealistically varied (up to 1,000 s) to illustrate the effect.

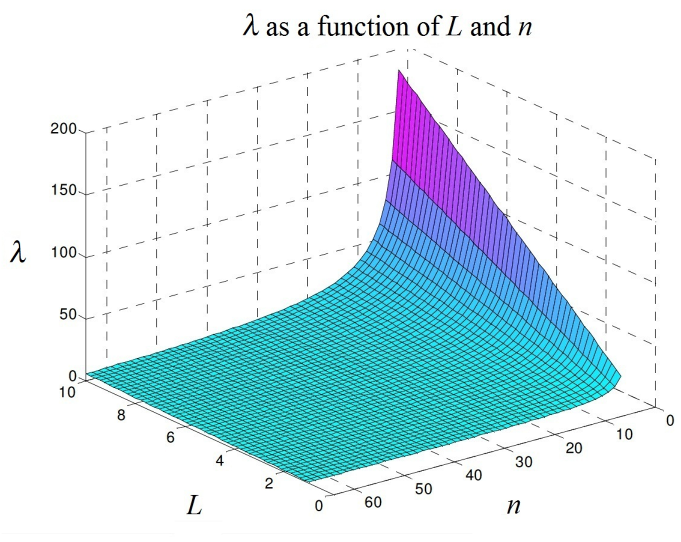


Figure 4.5: Lambda plot as function of link delays and number of cores [60].

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

4.11.2 Results and Discussion

The simulation was run with a simulation time limit of 100,000 simulation seconds (27,78 hours virtual time) and the elapsed time was found to be around 75.9 seconds on a single core for all values of L. The hardware environment was a Linux-based cluster of 2.66GHz 2x Quad-Core, 8GB RAM PCs, interconnected via Gigabit Ethernet. The tasks performed during the run included various client requests, technician visitations, equipment failures at random times, replacement of faulty equipment and automatic fault correction. The execution time for various configurations was noted and is presented in Table 4.3.

Table 4.3: Summary of simulation runs of the OMNeT++ modular AMDF model.

		Number of cores (n)					
		1	**2**	**4**	**8**	**16**	**32**
Link Delay (L)	**100 ms**	75.9	error	error	error	error	error
	1 s	76.4	68.8	21.9	10.4	110.9	476.3
	5 s	75.9	67.2	21.4	12.2	30.2	100.7
	10 s	75.5	67.2	20.8	10.2	19.3	88.3
	100 s	75.6	67.1	19.9	9.6	16.7	10.0
	1000 s	75.8	67.1	19.7	9.4	12.6	5.6

Some configurations yielded errors, since the link delays are not classified as sufficiently large according to the NMA class. This means that in order to maintain causality, the resulting speedup is severely penalised owing to synchronisation. The corresponding speedup for each simulation run was calculated and the results are presented in Figure 4.6. Client request intervals and equipment card expected lifetimes were unrealistically shortened in order to stress the model in terms of discrete events per second. It is therefore expected that longer simulation runs (in excess of tens of millions of virtual seconds) will follow the tendency of the graph in Figure 4.6 for lower client request frequencies and longer lifetimes as well.

Note that no significant speedup is observed between one and two cores, since all client coroutines are still executed on a single core. Otherwise, a significant overall speedup is observed for $n > 2$. This is due to the fact that the client coroutines are evenly divided between cores. It is clear that the presence of MPI communication overhead heavily affects the performance of the PDES model between eight and 16 cores, especially for smaller link delays. The performance penalty may be attributed to extra overhead added by the switch connecting the hosts in the cluster. The effect is especially evident when inter-process messages and NMA synchronisation are more frequent (i.e. when more hosts are involved). Furthermore, since the load is distributed, E and P are significantly reduced for all

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

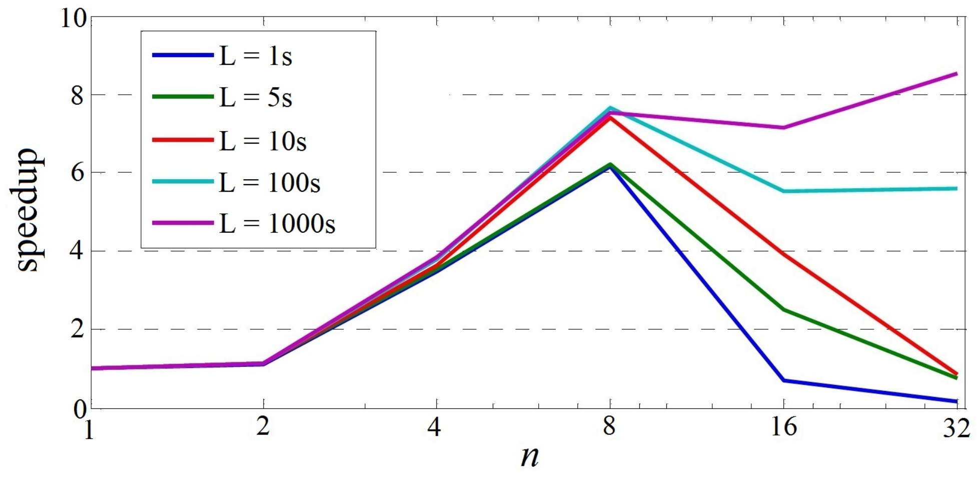


Figure 4.6: Speedup results for various *L* and *n* values.

local processes. The positive effect of using multi-node clusters becomes clear for extremely large link delays. Unfortunately, such large link delays (to model the actual system accurately) may be unacceptable for most designs. It is therefore concluded that the most efficient possible speedup can be obtained using MPI within a single node in the cluster for the current model.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 5

REDUNDANCY OPTIMISATION MODEL

5.1 CHAPTER OVERVIEW

This chapter presents and discusses a mathematical model used to optimise the redundancy allocation of a concept design. The purpose of this model is to supplement the AMDF model constructed in OMNeT++ in terms of evaluating and maximising the expected lifetime of a concept design subject to certain optimisation criteria such as cost, power and size. As mentioned in section 2.5, there are two general approaches to improving system reliability: increasing component reliability and/or adding redundancy. In order to increase the overall system reliability beyond that of the inherent reliability of standard grade components, the RAP was explored to determine the optimal redundant unit allocation for each component under various design constraints. Moreover, the extended specifications and cost of military grade components are not justified for this research.

5.2 NOTATIONS

This section lists the notations and symbols used to model the RAP (following the convention used in [23]):

n	number of components in the system
i	component i
$\mathbf{x}$	$x_1, x_2, \ldots, x_n$; decision vector
x_i	number of type-i elements selected for component i
$T_i(\mathbf{x}, \boldsymbol{\xi})$	lifetime of subsystem i
$T(\mathbf{x}, \boldsymbol{\xi})$	lifetime of system
$\mathbf{y}$	$y_1, y_2, \ldots, y_n$; state vector

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

$\mathbf{y}(t)$	$y_1(t), y_2(t), \ldots, y_n(t)$; state vector at time t
y_i	state of component i
$y_i(t)$	state of component i at time t
$\Psi(\mathbf{y})$	state of the system
$\xi_{i,j}$	random lifetime of redundant unit in component i, $j = 1, 2, \ldots, x_i$
$\boldsymbol{\xi}$	$(\xi_{1,1}, \xi_{1,2}, \ldots, \xi_{1,x_1}, \ldots \xi_{2,1}, \xi_{2,2}, \ldots, \xi_{2,x_2}, \ldots, \xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,x_n})$
N	number of cycles used for stochastic simulation
M	number of training pairs
C	Cost constraint
P	Power constraint
W	Weight/size constraint

5.3 ASSUMPTIONS

Generally, the following assumptions are made in redundancy optimisation, and was used when implementing concept designs in the model:

- All components in the system can assume one of two states: *functional* (denoted by 1) and *non-functional* (denoted by 0).
- Models can include a mixture of component redundancies with and without component mixing.
- Component failures are independent, following the convention used in stochastic optimisation publications [23, 62–64].
- A mixture of standby and active redundancy schemes can be employed.
- The failover switching time is perfect, following the convention used in [21–23, 29].
- All lifetimes are random variables (stochastic simulation [23, 29, 62–64]).

5.4 EXTERNAL LIBRARIES

This section lists the external packages and libraries included in the model. Refer to Table 5.1 for a brief summary of the libraries.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 5.1: Summary of external libraries.

Library/package	**Description**
Fast Artificial Neural Network (FANN)[1]	A free, open-source and widely used multilayer NN library. The library is easy to use, user-friendly and very fast in execution. The NN is implemented using this library.
The Boost Library[2]	This free set of C++ library extends the functionality of C++. The libraries range from simple pointer libraries to OS abstraction. The RNGs were of particular interest in this project. All number-generating functions are seeded by a number generated by the */dev/urandom* pseudorandom number generator.

5.5 THE MODEL

A mathematical model was realised in order to maximise the mean system lifetime, α-system lifetime, or system reliability of a concept design. Stochastic programming models concerning the maximisation of the expected system lifetime are presented and described in [23] for active and standby redundancy optimisation models. The models were particularly attractive for solving the RAP, especially for complex system configurations, since the approach avoids the derivation of analytical formulations of systems. The advantages are ease of implementation and less room for error. To solve these models, stochastic simulation and hybrid intelligent algorithms are combined to estimate the approximate optimal solutions. Following the guidelines presented in [23], the model is developed using C++. A modular OO approach was followed.

5.5.1 System Performance Metrics

Typically, there are three types of system performance measures used with system structure functions [23]:

1. *Expected system lifetime,* $E[T(\mathbf{x},\boldsymbol{\xi})]$: The goal is to maximise $E[T(\mathbf{x},\boldsymbol{\xi})]$ as it is desirable that the system remain functional for as long as possible.

2. α*-System lifetime*: The largest value of $\bar{T}$ satisfying $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq \alpha$. The system reaches an expected lifetime of $\bar{T}$ with probability α.

3. *System reliability,* $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq T^0\}$: The probability that the system lifetime will exceed or equal a given T^0.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Generally speaking, it is difficult to obtain exact solutions using an algorithm based on analytic solutions to calculate system performance metrics. This is mainly due to the extreme complexity of system structures. Using the proposed hybrid intelligent algorithm in [23], one way to alleviate the intense computation commonly involved with optimisation algorithms, is to incorporate stochastic simulation to estimate performance parameters. The relation between a system's structure function and the system's expected lifetime must be clarified.

It is assumed that $\Psi(\mathbf{y})$ is a monotonic decreasing function of time. That is, if $\Psi(\mathbf{y}(t_1)) = 1$, then $\Psi(\mathbf{y}(t_2)) = 1$ given that $t_2 \leq t_1$. The relation can formally be proposed as:

$$\textit{For any redundant system, } T(\mathbf{x}, \boldsymbol{\xi}) \geq t \textit{ iff. } \Psi(\mathbf{y}(t)) = 1.$$

Since $\Psi(\mathbf{y}(t))$ is a monotone function, a bisection search algorithm can be used to determine the crossover point where $\Psi(\mathbf{y}(t)) = 0$.

5.5.1.1 Stochastic Simulation of $E[T(\mathbf{x}, \boldsymbol{\xi})]$

The algorithm used to estimate the lifetime of the entire system can be summarised as follows:

1. Generate observational vectors, $\boldsymbol{\xi}_i$, from the lifetimes of each redundant unit ξ, $i = 1, 2, \ldots, N$.
2. Provide bounds, a_i and b_i, such that $\Psi(\mathbf{y}(a_i)) = 1$ and $\Psi(\mathbf{y}(b_i)) = 0$ for $\boldsymbol{\xi}_i$, $i = 1, 2, \ldots, N$.
3. Set $t_i = 0.5(a_i + b_i)$. Evaluate $\Psi(\mathbf{y}(t_i))$; if 1 then set new $a_i = t_i$, otherwise $b_i = t_i$, $i = 1, 2, \ldots, N$.
4. Repeat step 3. until $|a_i - b_i| < \varepsilon$, ε is a predefined error margin, $i = 1, 2, \ldots, N$.
5. Set $T_i(\mathbf{x}, \boldsymbol{\xi}_i) = 0.5(a_i + b_i)$, $i = 1, 2, \ldots, N$.
6. Calculate $E[T(\mathbf{x}, \boldsymbol{\xi})] = (1/N) \sum\limits_{i=1}^{N} T_i(\mathbf{x}, \boldsymbol{\xi}_i)$.

5.5.1.2 Stochastic Simulation of $\bar{T}$

Firstly, generate N random observational vectors, $\boldsymbol{\xi}_i$, from the lifetimes of redundant unit $\boldsymbol{\xi}$. Use the bisection search method to estimate $E[T(\mathbf{x}, \boldsymbol{\xi}_i)]$ for each $\boldsymbol{\xi}_i$, $i = 1, 2, \ldots, N$. Thus, a vector of lifetimes, $\{T(\mathbf{x}, \boldsymbol{\xi}_1), T(\mathbf{x}, \boldsymbol{\xi}_2), \ldots, T(\mathbf{x}, \boldsymbol{\xi}_N)\}$ is obtained. Next, take N' as the integer part of αN. The N'th largest value in the vector of lifetimes can be regarded as $\bar{T}$. The algorithm used to estimate $\bar{T}$

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

for the entire system can be summarised as follows:

1. Generate observational vectors, ξ_i, from the lifetimes of each redundant unit ξ, $i = 1,2,\ldots,N$.
2. Provide bounds, a_i and b_i, such that $\Psi(\mathbf{y}(a_i)) = 1$ and $\Psi(\mathbf{y}(b_i)) = 0$ for ξ_i, $i = 1,2,\ldots,N$.
3. Set $t_i = 0.5(a_i + b_i)$. Evaluate $\Psi(\mathbf{y}(t_i))$; if 1 then set new $a_i = t_i$, otherwise $b_i = t_i$, $i = 1,2,\ldots,N$.
4. Repeat step 3. until $|a_i - b_i| < \varepsilon$, ε is a predefined error margin, $i = 1,2,\ldots,N$.
5. Set $T(\mathbf{x}, \boldsymbol{\xi}_i) = 0.5(a_i + b_i)$, $i = 1,2,\ldots,N$.
6. Set N' as integer part of αN.
7. The N'th largest element in $\{T(\mathbf{x}, \boldsymbol{\xi}_1), T(\mathbf{x}, \boldsymbol{\xi}_2), \ldots, T(\mathbf{x}, \boldsymbol{\xi}_N)\}$ is the estimation of $\bar{T}$.

5.5.1.3 Stochastic Simulation of $Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq T^0\}$

This process is similar to the procedure for estimating $\bar{T}$. Let N' be the number of instances that $T(\mathbf{x}, \boldsymbol{\xi}_i) \geq T^0$, $i = 1,2,\ldots,N$. The overall system reliability can be estimated by N'/N. The algorithm is summarised as follows:

1. Set $N' = 0$.
2. Generate an observational vector, $\boldsymbol{\xi}^0$, from the lifetimes of each redundant unit ξ, $i = 1,2,\ldots,N$.
3. Provide bounds, a_i and b_i, such that $\Psi(\mathbf{y}(a_i)) = 1$ and $\Psi(\mathbf{y}(b_i)) = 0$ for ξ_i, $i = 1,2,\ldots,N$.
4. Set $t_i = 0.5(a + b)$. Evaluate $\Psi(\mathbf{y}(t))$; if 1 then set new $a_i = t$, otherwise $b = t$, $i = 1,2,\ldots,N$.
5. Repeat step 4. until $|a - b| < \varepsilon$, ε is a predefined error margin, $i = 1,2,\ldots,N$.
6. If $0.5(a + b) \geq T^0$, increment N'.
7. Repeat steps 2. and 6. N times.
8. Set $Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq T^0\} = N'/N$.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.5.2 Redundancy Optimisation Models

The models presented in [23] of particular interest to the goal of this research are the Redundancy Expected Value Model (REVM) and Redundancy Expected Value Multi-Objective Programming (REVMOP) models. The REVM, which deals with maximising the expected lifetime under various constraints, is given by:

$$\begin{cases} \quad\quad \max \quad E[T(\mathbf{x},\boldsymbol{\xi})] \\ \text{subject to :} \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[c_i]\cdot x_i \leq C \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[p_i]\cdot x_i \leq P \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[w_i]\cdot x_i \leq W \\ \quad\quad\quad\quad x_i \geq 1, \quad\quad \mathbf{x} \text{ is an integer vector} \end{cases} \tag{5.1}$$

The model presented in Equation 5.1 can be used with CM or without CM. A large system however typically consists of several local subsystems. Thus, a programming model with multiple goals must be considered. The REVMOP attempts to maximise the expected lifetime of all subsystems simultaneously under concurrent constraints. The REVMOP for m subsystems is:

$$\begin{cases} \quad\quad \max \quad E[T_1(\mathbf{x},\boldsymbol{\xi}),T_2(\mathbf{x},\boldsymbol{\xi}),\ldots,T_m(\mathbf{x},\boldsymbol{\xi})] \\ \text{subject to :} \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[c_i]\cdot x_i \leq C \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[p_i]\cdot x_i \leq P \\ \quad\quad\quad\quad \sum_{i=1}^{N} E[w_i]\cdot x_i \leq W \\ \quad\quad\quad\quad x_i \geq 1, \quad\quad \mathbf{x} \text{ is an integer vector} \end{cases} \tag{5.2}$$

The actual procedures for solving both models are similar. Stochastic simulation and GA operations (selection, crossover and mutation) are performed identically for both models. The only difference is how fitnesses are assigned to possible solutions (see subsubsection 2.6.2.3). Moreover, the REVMOP model in Equation 5.2 produces a set of Pareto-optimal solutions, which are presented to the decision-maker who selects the best solution according to the decision-maker's preference.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.6 HYBRID ALGORITHM

This section describes the hybrid intelligent algorithm that is implemented in the model. As mentioned earlier, stochastic simulation is used to estimate system performance metrics. However, this approach inherently requires significant CPU time owing to the large number of computations. In order to counter this effect, stochastic simulation is used to produce a training data set to train in an NN using a standard back-propagation algorithm.

5.6.1 Inclusion of the NN

The purpose of the NN is to approximate the system performance metrics during the optimisation process instead of calculating exact values, which significantly speeds up the optimisation process. The NN is trained from a specific set of component properties in order to approximate the system performance metrics of a possible solution. If any of the component properties is altered, such as the statistical distribution parameters, the NN is retrained to approximate the new system performance metrics. The approximated functions are used by the objective function to determine the best and worse solutions. An example of a set of uncertain functions is:

$$\begin{cases} U_1 : \mathbf{x} \to E[T(\mathbf{x}, \boldsymbol{\xi})] \\ U_2 : \mathbf{x} \to Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq \bar{T}\} \geq \alpha \\ U_3 : \mathbf{x} \to Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq T^0\} \end{cases} \tag{5.3}$$

or collectively as an uncertain function set:

$$\begin{cases} \mathbf{U} : \mathbf{x} \to (U_1(\mathbf{x}), U_2(\mathbf{x}), U_3(\mathbf{x})) \\ U_1(\mathbf{x}) \equiv E[T(\mathbf{x}, \boldsymbol{\xi})], \\ U_2(\mathbf{x}) \equiv Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq \bar{T}\} \geq \alpha, \\ U_3(\mathbf{x}) \equiv Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq T^0\}, \\ \mathbf{x} \equiv (x_1, x_2, \ldots, x_n), \\ x_i \in 1, 2, \ldots \end{cases} \tag{5.4}$$

Finally, the NN is embedded in a GA to form the hybrid intelligent algorithm for solving general stochastic programming models. This method (with a trained NN) significantly reduces simulation execution times.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.6.2 Data Generation

Data generation entails the collection of samples from an uncertain function via stochastic simulation:

$$\mathbf{U} = (U_1(\mathbf{x}), U_2(\mathbf{x}), \ldots, U_m(\mathbf{x}))$$

The process commences by randomly generating a single integer vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The integer vector is used as the input to stochastic simulation to estimate the uncertain function $\mathbf{U}(\mathbf{x})$. This process is repeated M times, producing M input-output training samples. The exact choice of the parameter M is not critical provided it is sufficiently large to ensure good generalisation of the NN. The same procedure can be used to generate a test data set used for training the NN.

5.6.3 Training Process

A standard back-propagation algorithm is used to train a feed-forward NN. For most applications, a single hidden layer suffices for the NN's ability to generalise. The NN is defined by n input neurons, m neurons in a hidden layer and p output layers. The number of neurons in the hidden layer is determined by $(M/(n \log M))^{1/2}$ as approximated in [65] when $M/n >> 30$. In this case, the number of input neurons is equal to the size of the input vector, i.e. the number of components in the system. During training, the error between the output of the NN and a test data set is continuously monitored; once a persistent increase in the error is noticed over multiple training cycles, training is stopped.

Using the hybrid intelligent algorithm, a possible solution vector, $\mathbf{x}$, is represented as a chromosome in the GA as $V = (x_1, x_2, \ldots, x_n)$. The corresponding output vector produced by the NN is evaluated by the GA.

5.6.4 Initialisation Process

A set of initial chromosomes (the initial population) is generated. If a randomly generated chromosome, V, is proven to be a feasible solution by meeting all the constraints of the REVM or REVMOP, it is accepted as an initial chromosome. This process is repeated until the initial population is filled, based on the arbitrary, preselected population size (`popsize`). The resulting initial population consists of a chromosome set $\{V_1, V_2, \ldots, V_{\texttt{popsize}}\}$.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.6.5 Evaluation Function

The objective functions of all the chromosomes in the initial population are estimated by the trained NN. Based on these values, the chromosomes are arranged in an ordered relationship. Once the order has been determined, the population is rearranged from good to bad. The arranged population is still denoted by $\{V_1, V_2, \dots, V_{\texttt{popsize}}\}$ for convenience. Finally, a rank-based evaluation function is defined as follows:

$$eval(V_i) = a \cdot (1-a)^{i-1} \qquad i = 1, 2, \dots, \texttt{popsize} \tag{5.5}$$

where $a \in (0,1)$ is a parameter in the GA.

5.6.6 Selection Process

Selection is employed by spinning the roulette wheel `popsize` times [23]. First, the cumulative probability for each chromosome, $V_i,\ i = 1, 2, \dots, n$ is calculated based on its rank, i.e.:

$$q_i = \sum_{j=1}^{i} Eval(j), \qquad i = 1, 2, \dots, \texttt{popsize} \tag{5.6}$$

Next, a random real number , $r \sim U(0, q_{\texttt{popsize}})$ is generated, which is used to select a parent. Consequently, chromosome i is selected for the new population in the next generation if $q_{i-1} < r < q_i$ with $q_0 = 0$, $i = 1, 2, \dots, \texttt{popsize}$.

5.6.7 Crossover

The crossover operation requires that two parents are selected using the selection process described above. A random real value, $r \sim U(0,1)$ is generated for each $i = 1$ to `popsize`. If $r < P_c$ (crossover rate), chromosome V_i is selected as one of the parents. Next, the two crossover points, n_1 and n_2, are determined randomly. The crossover is performed as described in subsection 2.7.1. The purpose of crossover is to create more feasible chromosomes in context of the problem, which is attempted by merging the genes of the best chromosomes in a population to hopefully form a better offspring.

© University of Pretoria

5.6.8 Mutation

Similar to the crossover operation, a random real value, $r \sim U[0,1]$, is generated for each chromosome. Chromosome V_i is only selected for mutation provided that $r < P_m$ (mutation rate). A random integer, $n' \sim U[1,n]$, is generated as the mutation position. Another random integer is generated to replace the gene at n'. In other words, a random solution at another location in the search space is considered. After mutation and crossover operations are completed on a candidate chromosome, the same feasibility check is performed to ensure the chromosome does not exceed any design constraints.

5.6.9 Hybrid Intelligent Algorithm Procedure

5.6.9.1 Single Objective

After selection, crossover and the mutation operation are completed, a new generation is available for the next evaluation. The hybrid intelligent algorithm is terminated after a user-defined number of cyclic repetitions have been completed. The execution cycle is summarised below:

1. Define component parameters, system structure function and stochastic simulation parameters.
2. Generate training and test data sets for uncertain function(s) via stochastic simulation.
3. Train a feed-forward NN to approximate the uncertain function(s) using a standard back-propagation algorithm according to the training data set.
4. Enter parameters for GA.
5. Initialise the initial population by accepting chromosomes representing feasible solutions that can be evaluated by the trained NN.
6. Generate new chromosomes by means of crossover and mutation and accepting the chromosomes representing feasible solutions.
7. Calculate the objective values via the trained NN for each chromosome in the population.
8. Sort the population according to chromosome fitnesses and determine the ranks.
9. Select chromosome parents by spinning the roulette wheel to perform crossover and mutation

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

operations for the next generation.

10. Repeat steps 6. – 9. for a given number of cycles.

11. Report the best chromosome as the optimal solution.

5.6.9.2 Multiple Objectives

The MOGA algorithm approach presented in [44, 66] uses Pareto-based ranking techniques to encourage the search towards the true Pareto front while maintaining diversity in the population. Furthermore, by assigning random weights to the objective vector during fitness assignments, multiple random search directions towards the Pareto front are imposed during the simulation run (see [66] for a detailed explanation). This approach was adopted to implement the redundancy optimisation model. The entire procedure is summarised as follows:

1. Define component parameters, system structure function, stochastic simulation parameters and objectives.

2. Generate training and test data sets for uncertain function(s) via stochastic simulation for multiple objectives.

3. Train a feed-forward NN to approximate the uncertain function(s) using a standard back-propagation algorithm according to the training data set.

4. Enter parameters for GA.

5. MOGA:

   (a) Initialise the initial population that can be evaluated by the trained NN.

   (b) Calculate objective values to form the objective vector.

   (c) Update a tentative set of Pareto optimal solutions (elitist strategy).

   (d) Determine the fitness vector of each chromosome using random weights $\in [0,1]$.

   (e) Select parents from the current population to perform crossover and mutation operations for next generation.

   (f) Add all non-dominated solutions in the current population to the tentative Pareto-optimal

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

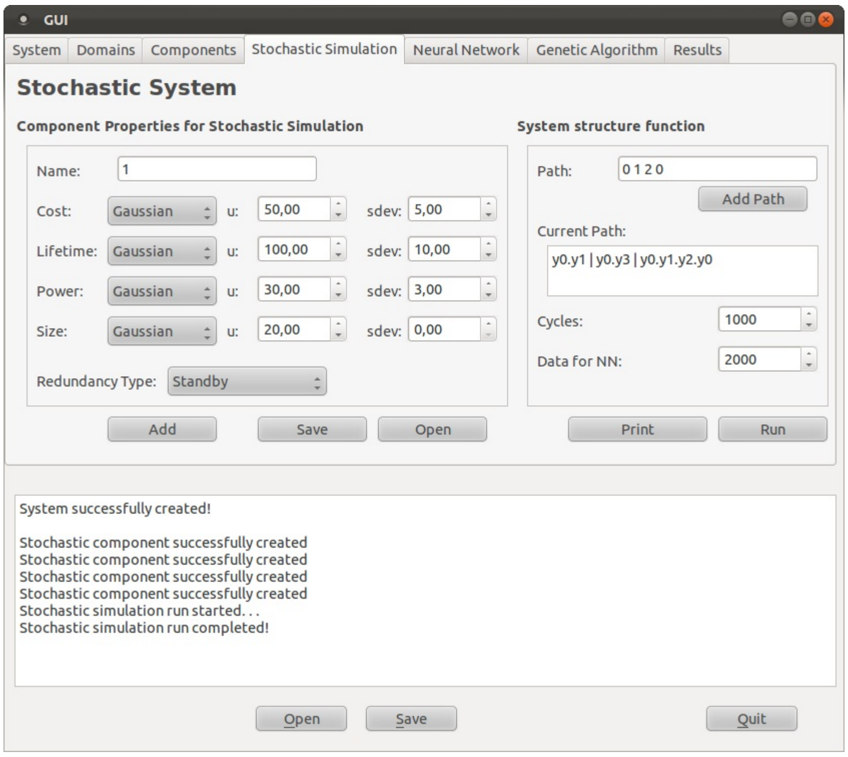
set and remove solutions dominated by new entries.

6. The MOGA shows the final set of Pareto-optimal solutions to the human decision-maker. The best solution is then selected according to higher level information by the decision-maker.

Recall that the solutions in a Pareto-optimal set are non-dominated with respect to each other. All the solutions are *acceptable or compromise solutions* and requires additional high-level information to select a single solution.

5.7 USAGE

User interaction is established through a GUI. The GUI was developed using the QT library from Nokia[3]. The GUI provides message outputs to inform the user of its current state. A snapshot of the GUI is shown in Figure 5.1. A complete flowchart describing the flow of the application is provided in Figure 5.2.



Stochastic System

Component Properties for Stochastic Simulation

Name: 1

Cost: Gaussian u: 50,00 sdev: 5,00

Lifetime: Gaussian u: 100,00 sdev: 10,00

Power: Gaussian u: 30,00 sdev: 3,00

Size: Gaussian u: 20,00 sdev: 0,00

Redundancy Type: Standby

System structure function

Path: 0 1 2 0

Current Path: y0.y1 | y0.y3 | y0.y1.y2.y0

Cycles: 1000

Data for NN: 2000

```
System successfully created!

Stochastic component successfully created
Stochastic component successfully created
Stochastic component successfully created
Stochastic component successfully created
Stochastic simulation run started. . .
Stochastic simulation run completed!
```

Figure 5.1: Snapshot of the GUI.

[3]Available at: http://qt.nokia.com/

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

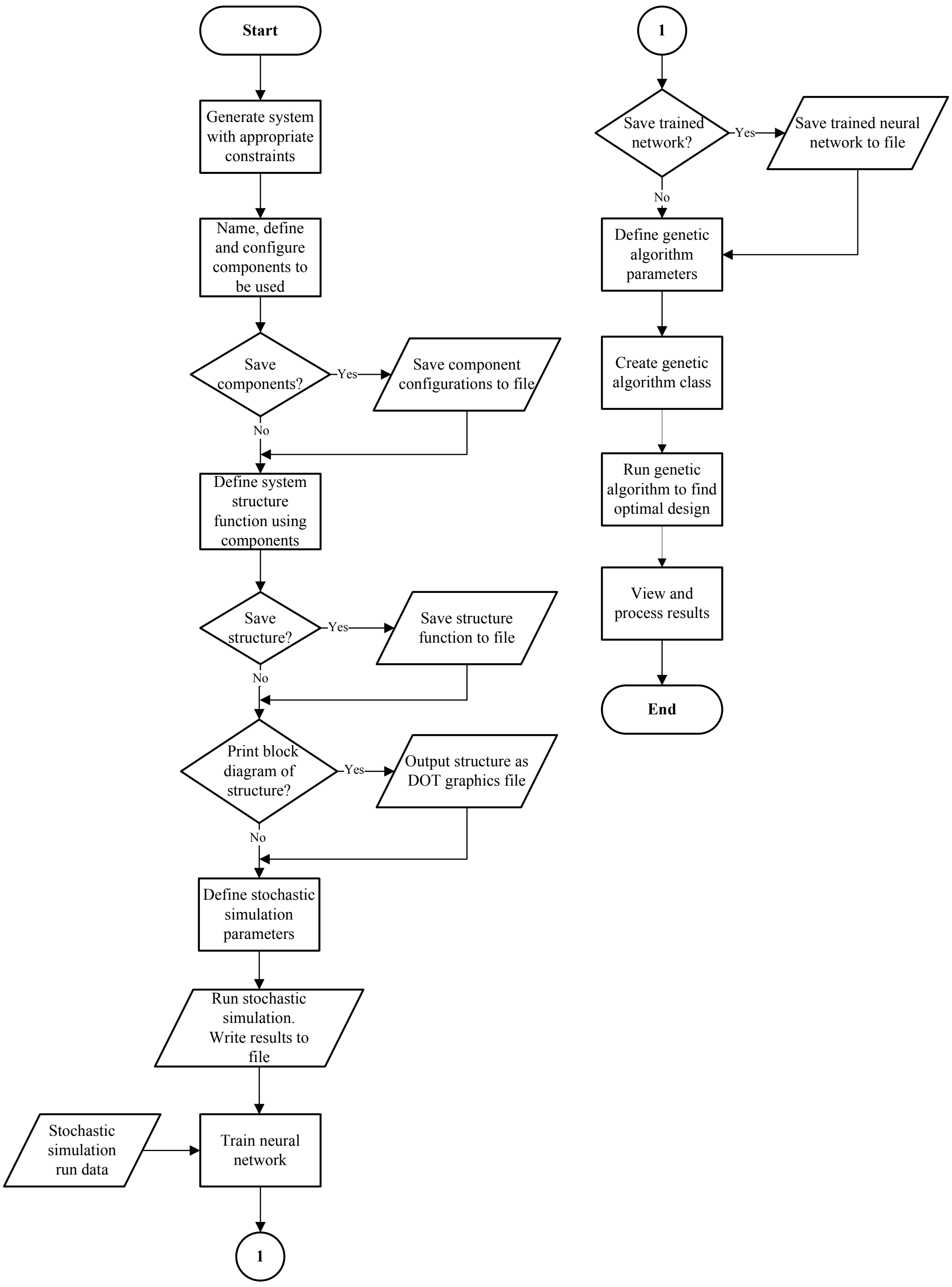

Start
Generate system with appropriate constraints
Name, define and configure components to be used
Save components?
Yes
Save component configurations to file
No
Define system structure function using components
Save structure?
Yes
Save structure function to file
No
Print block diagram of structure?
Yes
Output structure as DOT graphics file
No
Define stochastic simulation parameters
Run stochastic simulation. Write results to file
Stochastic simulation run data
Train neural network
1
1
Save trained network?
Yes
Save trained neural network to file
No
Define genetic algorithm parameters
Create genetic algorithm class
Run genetic algorithm to find optimal design
View and process results
End

Figure 5.2: Flow diagram of the redundancy optimisation model.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.8 EVALUATION

This section evaluates the implementation of the redundancy optimisation model by implementing well known problems. Specifically, the computation of the performance parameters are scrutinised and compared to the results of other publications. The section concludes with a few simple examples of redundancy optimisation problems. Component lifetimes may assume any unit of time, such as hours, days, etc. Likewise, cost may be defined with any unit, such as \$, ¥, R, etc. For evaluation purposes, no units are assigned for the parameters. In some instances, some parameters may not be applicable for certain components. For example, a power supply does not contribute to the total power consumption of the system. As a result, that property is denoted by "N/A" and is ignored when checking whether the system properties exceed the constraints.

5.8.1 Series System Evaluation

5.8.1.1 Evaluation

The behaviour of a series system was evaluated for various parameters. Consider the series configuration depicted in Figure 5.3. The corresponding system structure function is straightforward: $\Psi(\mathbf{y}) = y_0 \cdot y_1 \cdot y_2 \cdot y_3 \cdot y_4$.

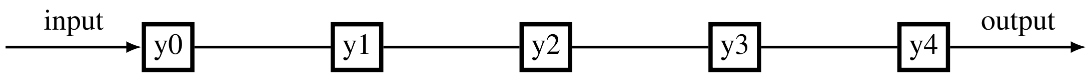


Figure 5.3: A series system.

The first task was to perform stochastic simulation of the three performance criteria and the effect some parameter changes can impose on the results. All five components have been assigned random normal distributed lifetimes. The goal was to observe $E[T(\mathbf{x},\boldsymbol{\xi})]$, $\bar{T}$, and $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq T^0\}$ for the arbitrarily chosen decision vector, $\mathbf{x} = (4,4,4,4,4)$, i.e. each component has four redundant units, with no component mixing. For simplicity, the following normal distributions have been assigned to the respective components:

- y_0- Lifetime: $N(100, 10^2)$; Standby,
- y_1- Lifetime: $N(150, 10^2)$; Standby,
- y_2- Lifetime: $N(220, 10^2)$; Standby,

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- y_3- Lifetime: $N(180, 10^2)$; Standby,
- y_4- Lifetime: $N(200, 10^2)$; Standby.

Relatively small variances were assigned to the lifetimes in order to focus on lifetimes close to the mean provided. Standby redundancy results in a component lifetime equal to the sum of all the lifetimes of the redundant units. Stochastic simulation with 10^4 cycles was performed to estimate $E[T(\mathbf{x}, \boldsymbol{\xi})]$ accurately. Figure 5.4 shows the result, in which the straight line represents the expected lifetime 400.028, and the curve represents the expected system lifetimes obtained by various simulation cycles. It is noticed that the relative error of the results obtained is less than 0.08 % after 2000 cycles of simulation and converges to the estimated value of 400.028.

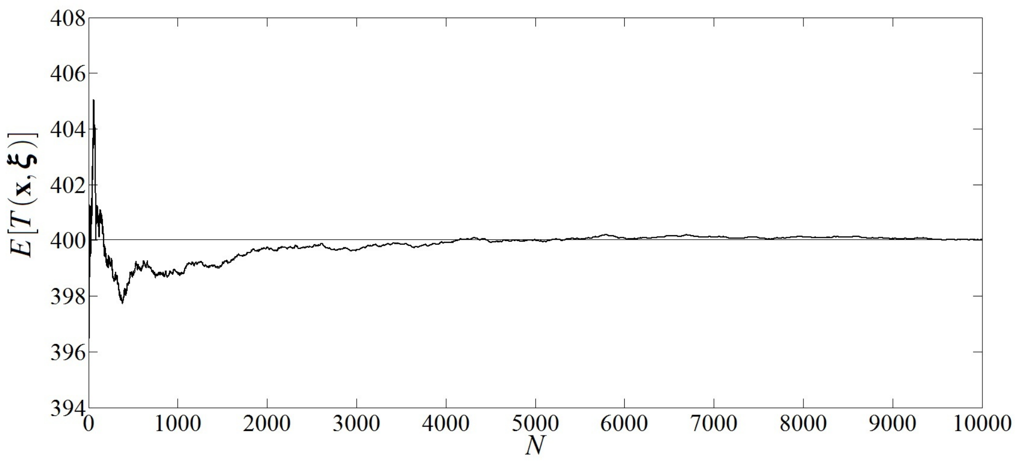


Figure 5.4: Estimated lifetime of normal distributed lifetimes of standby redundancy as a function of the number of simulation cycles for a series system.

Next, stochastic simulation was used to estimate $\bar{T}$ such that $Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90$, i.e. the 0.9-system lifetime, also an arbitrary value. The run that also consisted of 10^4 cycles returned a maximal value of 425.705. This means that the system lifetime approaches 425.705 with a probability of 90 %.

Finally, the reliability of the system is evaluated in a run of 10^4 cycles for 350, i.e. $Pr\{T(\mathbf{x}, \boldsymbol{\xi}) \geq 350\}$. The resulting probability is returned as 0.9946. This means the system exhibits a reliability of 99.46 % for a required lifetime of 350.

Once all three performance criteria had been estimated for the provided lifetimes, the lifetimes of the components were changed to equivalent exponential distributions, i.e. exponential distributions with averages equal to the original uniform distributions. The lifetimes of the components were assigned

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

as follows:

- y_0- Lifetime: $Exp(100)$; Standby,
- y_1- Lifetime: $Exp(150)$; Standby,
- y_2- Lifetime: $Exp(220)$; Standby,
- y_3- Lifetime: $Exp(180)$; Standby,
- y_4- Lifetime: $Exp(200)$; Standby.

The process described above was repeated and the following results were produced: $E[T(\mathbf{x},\boldsymbol{\xi})]$ = 395.397 , $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90$ = 612.709, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 350\}$ = 0.574. The following section provides an explanation of why some of the values differ a great deal. Figure 5.5 shows $E[T(\mathbf{x},\boldsymbol{\xi})]$ for the exponential distributed lifetimes. In this case, more simulation cycles are required for convergence to the estimated lifetime value.

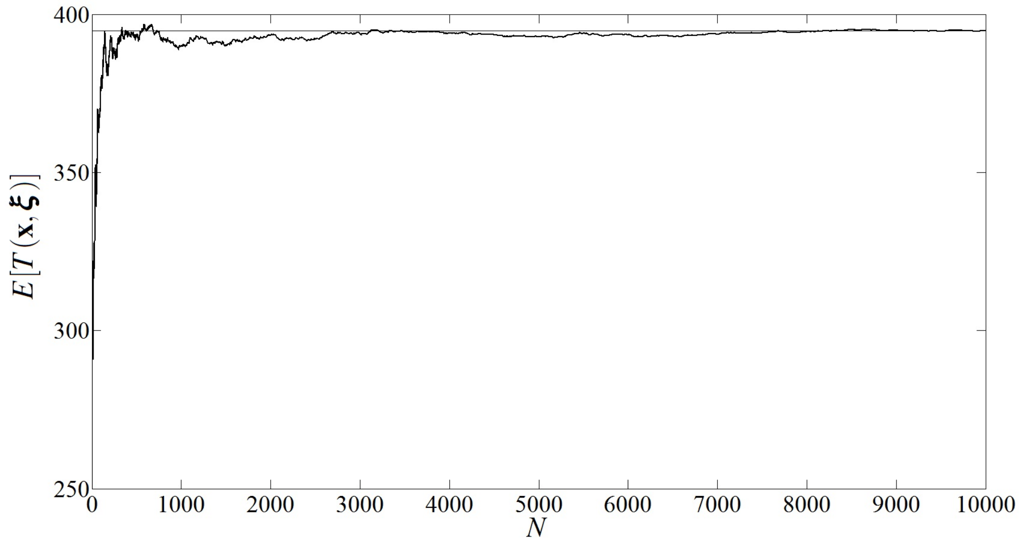


Figure 5.5: Estimated lifetime of exponential distributed lifetimes of standby redundancy as a function of the number of simulation cycles for a series system.

Next, the standby redundant units were replaced with active redundant units. In this configuration the total component lifetime is determined by the longest lifetime of the components in parallel. The same decision vector, $\mathbf{x} = (4,4,4,4,4)$, was used. The same normal distributions as described above were assigned to component lifetimes. Figure 5.6 shows the estimated lifetime simulation run.

The figure shows that the the estimated lifetime remains within a very small margin from the actual estimated lifetime as the simulation run approaches 10^4 cycles. Similar to standby redundancy, the

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

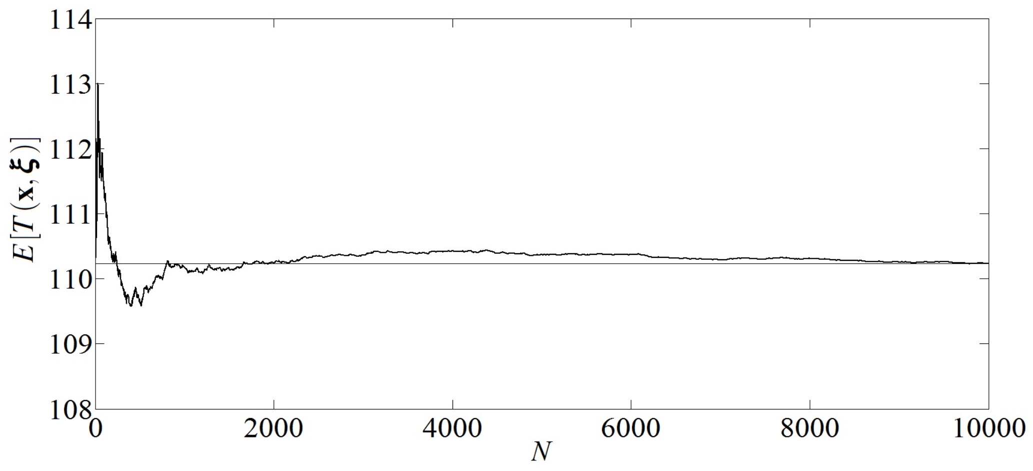


Figure 5.6: Estimated lifetime of normal distributed lifetimes of active redundancy as a function of the number of simulation cycles for a series system.

overall system lifetime is dictated by the shortest lifetime average of all the components in series, i.e. 100 in this case. The estimated lifetime after 10^4 cycles is 110.232 because of the longest lifetime of the redundant units was around 110. The estimated performance parameters are: $E[T(\mathbf{x},\boldsymbol{\xi})]$ = 395.397 , $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90$ = 119.242, and the reliability for a lower lifetime (100) is $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 100\} = 0.941$.

5.8.1.2 Discussion

In a series configuration the system is only operational if all the components are functional. It is often said that a chain is as strong as its weakest link; by implication the weakest component in a series system is the most important. Therefore, it is expected that the total system lifetime will be equal to the shortest lifetime of all the components in series. Standby redundancy results in a component lifetime that is equal to the sum of the lifetimes of the redundant units. It therefore comes as no surprise that the shortest lifetime of all the redundant units, averaging 100 with four redundant units, is a average estimated lifetime of 400.028 for the entire system.

When the distributions are changed to exponential distributions, it was expected that $E[T(\mathbf{x},\boldsymbol{\xi})]$ would remain constant, since the new distribution had the same mean. However, since the variance of an exponential distribution is much greater than that of the the equivalent normal distribution, it was expected that the the α-system lifetimes would be much greater and the estimated reliability much lower than the previous run. The reason for this is that since $\bar{T}$ is estimated as the N'th largest element

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

in the array of lifetimes (see subsubsection 5.5.1.1) and because of the larger variance, $\bar{T}$ at the N'th position would be larger. Similarly, the estimation of system reliability (subsubsection 5.5.1.1) entails adding the number of instances that the expected lifetime exceeds a given lifetime. The result is that a greater portion of the estimated lifetimes fall below the given lifetime, diminishing the ratio as well. Indeed, this was the case. It is expected that a similar observation will be made with normal distributions with larger variances.

In conclusion, it is recommended to avoid components with lifetimes with great variances in a series system because of the system structure function. It is seen, as expected, that active redundancy results have much lower estimated lifetimes than standby redundancy. However, the controlling mechanism typically involved with implementing standby redundancy has a huge negative impact on the cost. The true effect of active redundancy may nevertheless be observed for components with large variances. Furthermore, it was noted that large variances in component lifetimes do not affect the estimated lifetime, but instead drastically raise the α-system lifetime and decreases system reliability. Moreover, it appears that active redundancy only has a significant impact on the lifetime of the system when components with large variances are considered, such as an exponential distribution.

5.8.2 Parallel System Evaluation

5.8.2.1 Evaluation

Consider the parallel configuration depicted in Figure 5.7 with the following system structure function: $\Psi(\mathbf{y}) = y_0|y_1|y_2|y_3|y_4$. Note that the parallelism is unrelated to the parallelism introduced for redundancy purposes, but is inherent to the operation of the system. Again, the $\mathbf{x} = (4,4,4,4,4)$ decision vector was chosen in order to compare results with the series system. The same normal distributions used in the series system were assigned to the lifetimes of each component, i.e.:

- y_0- Lifetime: $N(100, 10^2)$; Standby,
- y_1- Lifetime: $N(150, 10^2)$; Standby,
- y_2- Lifetime: $N(220, 10^2)$; Standby,
- y_3- Lifetime: $N(180, 10^2)$; Standby,
- y_4- Lifetime: $N(200, 10^2)$; Standby.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

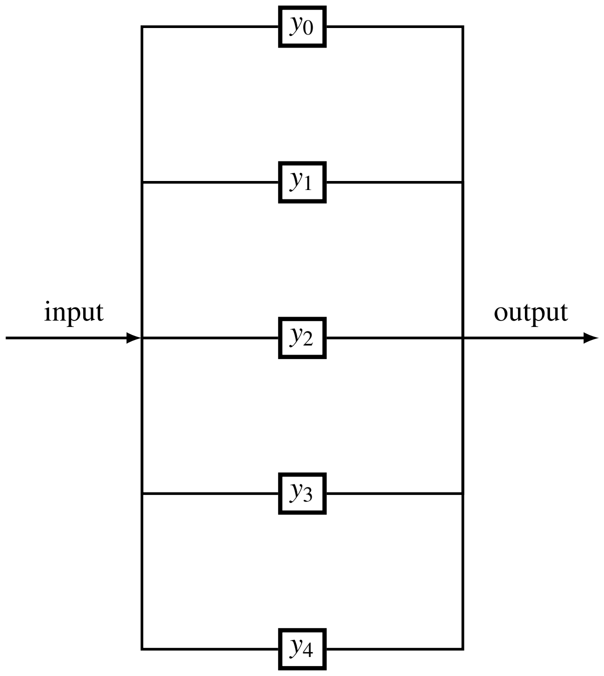


Figure 5.7: A parallel system.

The following results have been observed: $E[T(\mathbf{x},\boldsymbol{\xi})] = 880.103$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90 = 905.505$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 850\} = 0.9376$ or 93.76 % reliability. The estimation is shown in Figure 5.8. It is evident that the estimated lifetime is very large with regard to the assigned lifetimes. It is approximately equal to four times the largest average of the lifetime normal distributions, i.e. $N(220,10)$, owing to the chosen decision vector with four redundant units for each component.

On the other hand, when the redundant units were defined with an active redundancy scheme on the system structure function above, the following results were obtained: $E[T(\mathbf{x},\boldsymbol{\xi})] = 230.46$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90 = 239.56$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 220\} = 0.9418$.

5.8.2.2 Discussion

In a serial configuration the system is only operational if all the components are functional. A parallel configuration of components inherently extends the total expected lifetime of a system, since multiple paths exist between the input and the output. As a result, the overall system will be functional until the last operational component in a parallel system/subsystem fails. By adding redundancy to a parallel system, the total expected lifetime increases drastically, especially when a standby redundancy scheme is employed. Active redundancy results in a system lifetime that is equal to the longest life-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

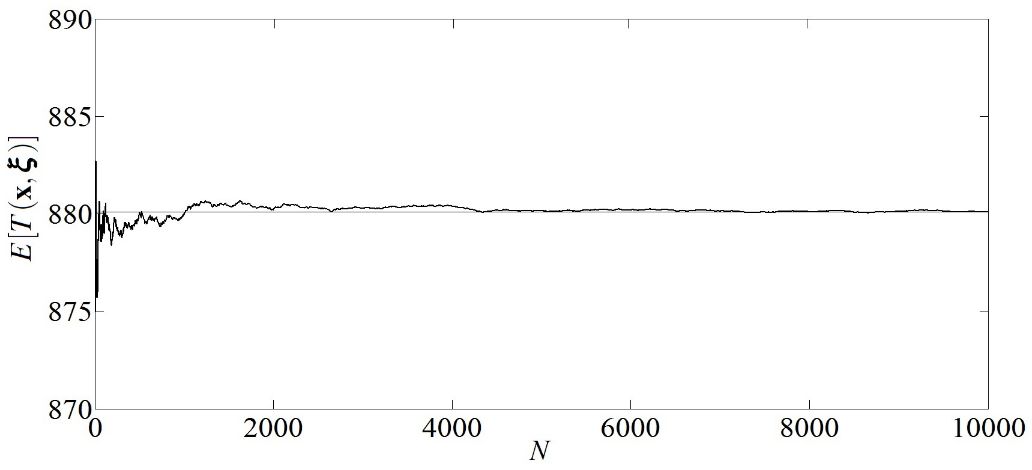


Figure 5.8: Estimated lifetime of normal distributed lifetimes of active redundancy as a function of the number of simulation cycles for a parallel system.

time of all the components. The parallel configuration may be used to replace other smaller subsets of standby redundancies, alleviating the complexity and cost involved in such a redundancy arrangement.

It is clear from the parallel system evaluation that the component with the longest expected lifetime dictates the expected lifetime of the overall parallel system/subsystem, regardless of the redundancy scheme implemented, and can replace an entire branch of the parallel system, unless the components are not functionally identical. If the components are not functionally identical, any redundancy scheme can be used to extend the lifetime of the system.

5.8.3 Complex System

5.8.3.1 Evaluation

A complex network is defined as one that cannot be modelled as a series, parallel, or series-parallel system. Examples of complex systems include a common PC, communication, transportation, electrical and manufacturing networks. A practical example of a complex system configuration of a life-support system in a space capsule can be found in [67]. Another example of a complex system is the bridge system depicted in Figure 2.6. For convenience, this figure is repeated in Figure 5.9. By means of inspection, the system structure function can be deduced as $\Psi(\mathbf{y}) = \{y_0 \cdot y_3 \mid y_0 \cdot y_2 \cdot y_4 \mid y_1 \cdot y_4 \mid y_1 \cdot y_2 \cdot y_3\}$. The system reliability can be derived by using

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Equation 2.3 for each path: assuming that event R_i indicates the reliability of component i, then the system reliability is [68]:

$$\begin{aligned} R_S = {} & R_0R_3 + R_1R_4 + R_0R_2R_4 + R_1R_2R_3 - R_0R_1R_2R_3 \\ & - R_0R_1R_2R_4 - R_0R_2R_3R_4 - R_1R_2R_3R_4 \\ & - R_0R_3R_1R_4 + 2R_0R_3R_1R_4R_2. \end{aligned}$$

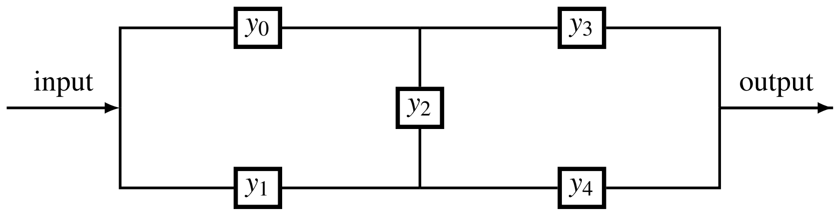


Figure 5.9: A bridge configuration.

It is evident that even for a very simple example of a complex system, the mathematics describing the system are quite broad and extensive. Such a description leaves much room for errors and may prove difficult to implement in the model. Consequently, it may be infeasible to derive analytic formulations for non-trivial systems, which was the main motivation for incorporating the bisection search algorithm to estimate the system performance metrics stochastically for any type of system.

For comparison purposes, the same normal distributed lifetimes are assigned to the components with the series and parallel systems. The results are as follows: $E[T(\mathbf{x},\boldsymbol{\xi})] = 599.671$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90 = 625.822$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 550\} = 0.9937$. In essence, this result means that the longest estimated lifetime out of all the possible paths is equal to $E[T(\mathbf{x},\boldsymbol{\xi})]$. The estimation is shown in Figure 5.10. It is interesting to note that for a complex system that the α-system lifetime is actually slightly higher than the estimated lifetime. This may be due to multiple paths with varying lifetimes with different means.

The experiment was repeated with an active redundancy scheme and the same component lifetime parameters. The simulation run yielded: $E[T(\mathbf{x},\boldsymbol{\xi})] = 160.332$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90 = 169.328$ and $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 550\} = 0.9366$. Figure 5.11 shows the estimation. Little deviation is noticed around the mean of 160.332.

According to [69], the bridge configuration in Figure 5.9 can be transformed to a series-parallel

© University of Pretoria

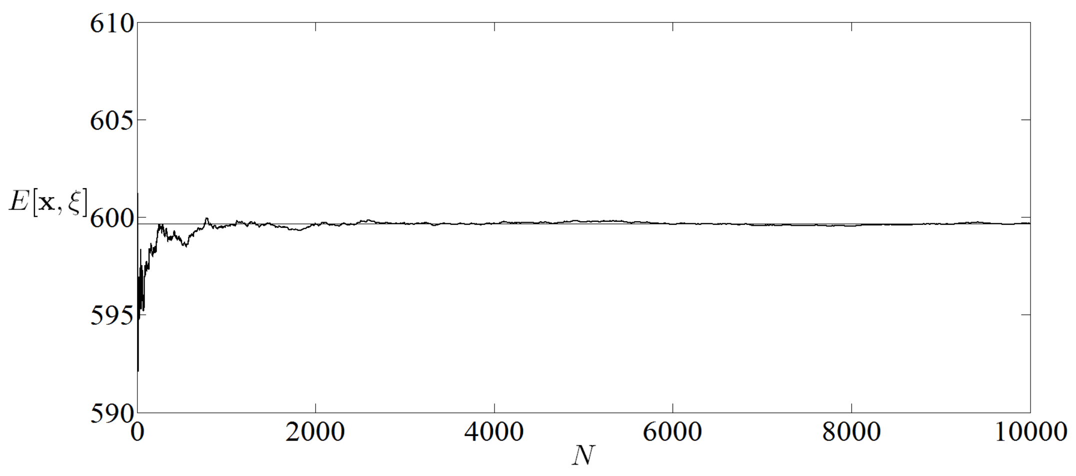


Figure 5.10: Estimated lifetime of normal distributed lifetimes of standby redundancy as a function of the number of simulation cycles for a complex system.

system with four subsystems. The transformed network is shown in Figure 5.12 and the corresponding system structure function by inspection is: $\Psi(\mathbf{y}) = (y_0 \mid y_1) \cdot (y_3 \mid y_4) \cdot (y_0 \mid y_2 \mid y_4) \cdot (y_1 \mid y_2 \mid y_3)$. One should notice that the new configuration goes against the assumption of independent component failures, since components are repeated in different subsystems. The intent, however, was to verify the validity of the implemented model. By constructing the structure depicted in Figure 5.12 in the model, similar results should be obtained by stochastic simulation as with the structure in Figure 5.9. The same random lifetimes were assigned to each component, and a stochastic simulation was run for 10^4 cycles. The results were: $E[T(\mathbf{x},\boldsymbol{\xi})] = 599.703$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90 = 626.209$, $Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 550\} = 0.9366$. The estimation is shown in Figure 5.13 and a comparison of the results is shown in Table 5.2.

Table 5.2: Comparison of complex system results.

Performance	Bridge system	Transformed system
$E[T(\mathbf{x},\boldsymbol{\xi})]$	599.671	599.703
$Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq \bar{T}\} \geq 0.90$	625.822	626.209
$Pr\{T(\mathbf{x},\boldsymbol{\xi}) \geq 550\}$	0.9937	0.9936

5.8.3.2 Discussion

It is evident that for a complex system the estimated system lifetime cannot be easily predicted by visual inspection. It is difficult to predict which component will have the greatest effect on the system lifetime without systematically running through each path, regardless of the redundancy scheme

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

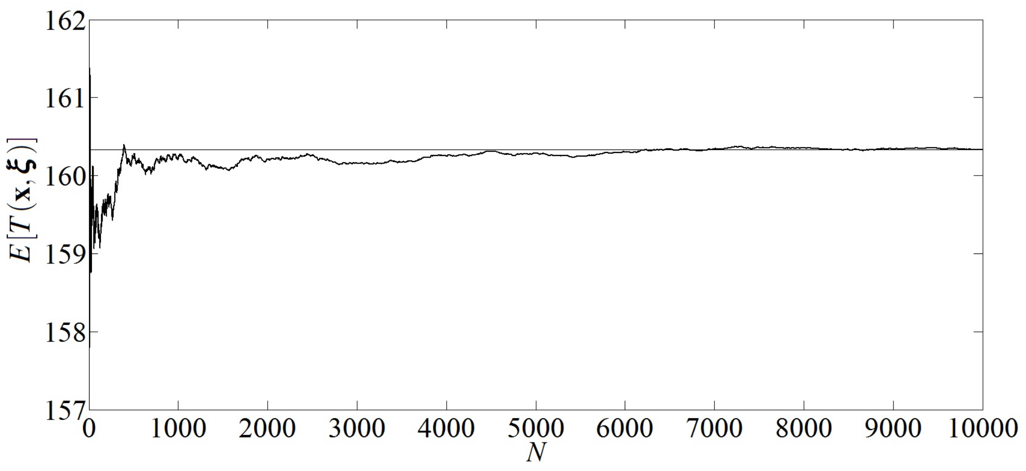


Figure 5.11: Estimated lifetime of normal distributed lifetimes of active redundancy as a function of the number of simulation cycles for a complex system.

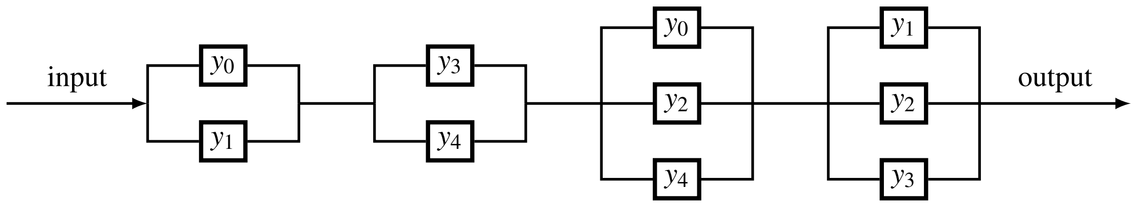


Figure 5.12: A series-parallel transformation of the bridge network.

used.

The validity and accuracy of the implemented models were confirmed. Two different system structure functions, which were shown to be a transformation of the other, yielded almost identical results during stochastic simulation runs.

5.8.4 Single-objective Optimisation Problems

This section explores a few numerical examples, assessing the effectiveness of the complete RAP simulation model. All experiments were performed with the following configuration parameters (typical values in [23, 64]):

- Stochastic simulation:
  - 3000 data for NN: Sufficient to train the NN successfully, in order to cover most of the combinations of the decision vector.
  - 10000 cycles: Previous section showed satisfactory convergence to the estimated lifetime

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

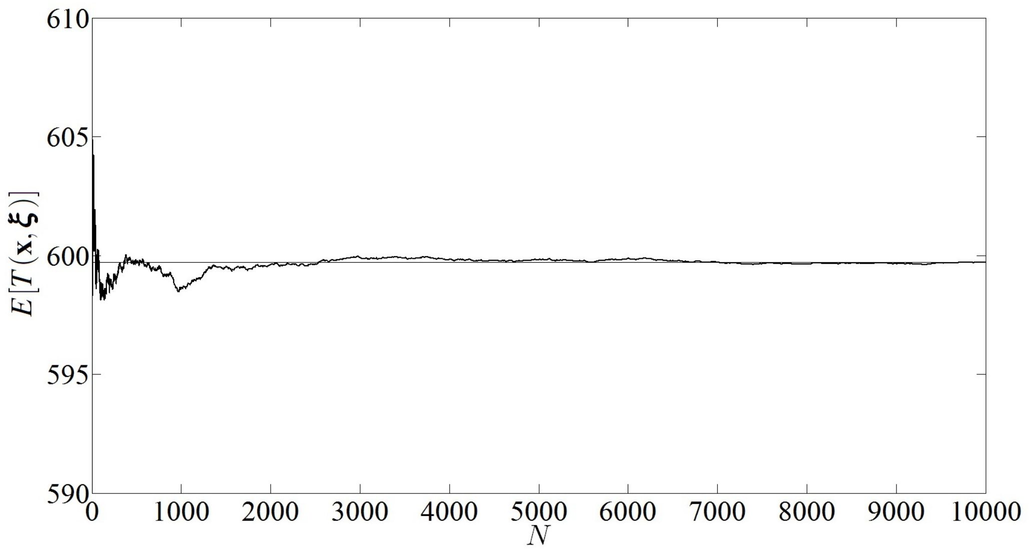


Figure 5.13: Estimated lifetime of normal distributed lifetimes of standby redundancy as a function of the number of simulation cycles for the transformed bridge system.

of the system.

- NN:
  - Number of input neurons are equal to the number of components in the system.
  - Neurons in hidden layer determined by $(M/(n\log M))^{1/2}$ as approximated in [65] when $M/n >> 30$.
  - One output neuron for each of the system performance metrics.
- GA:
  - Population size = 20,
  - P_c = 30 %,
  - P_m = 20 %,
  - a = 0.05,
  - Maximum generations = 300.

5.8.4.1 Series System with Constraint

Refer to the series system depicted in Figure 5.3. The goal was to maximise the expected lifetime of the serial system under arbitrary constraints. The following is a list of all the parameters specific to this model that were used:

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- Constraints:
  - Cost: 350, an arbitrary constraint.
  - Power: N/A
  - Size: N/A
  - Weight: N/A
- Components without CM.
- Components (lifetimes used in previous section with arbitrary costs):
  - y_0- Cost: 80; Lifetime: $N(100,10^2)$; Standby,
  - y_1- Cost: 20; Lifetime: $N(150,10^2)$; Standby,
  - y_2- Cost: 30; Lifetime: $N(220,10^2)$; Standby,
  - y_3- Cost: 50; Lifetime: $N(180,10^2)$; Standby,
  - y_4- Cost: 10; Lifetime: $N(200,10^2)$; Standby.

A run of the hybrid intelligent algorithm returned the following optimal solution:

$$\mathbf{x}^* = (2,2,2,1,2),$$

with a corresponding lifetime of

$$E[T(\mathbf{x}^*,\boldsymbol{\xi})] = 523.54,$$

and a total corresponding cost of

$$C(\mathbf{x}^*) = 334.76.$$

To verify the consistency of the simulation model, the experiment was repeated for a hundred runs. The resulting expected lifetimes are incorporated in the histogram displayed in Figure 5.14. It is clear that the resulting histogram takes the form of an undefined statistical distribution, which can be best approximated as a normal distribution with a mean of 524.

5.8.4.2 A practical complex system

The life-support system in a space capsule shown in [67] is analysed. Another analysis of the same model is presented in [23]. The experiment is repeated with the same parameters in this section. A block diagram of the system with rearranged component indices are shown in Figure 5.15.

Let the seven types of components have normal distributed lifetimes: $N(290,21^2)$, $N(533,23^2)$,

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

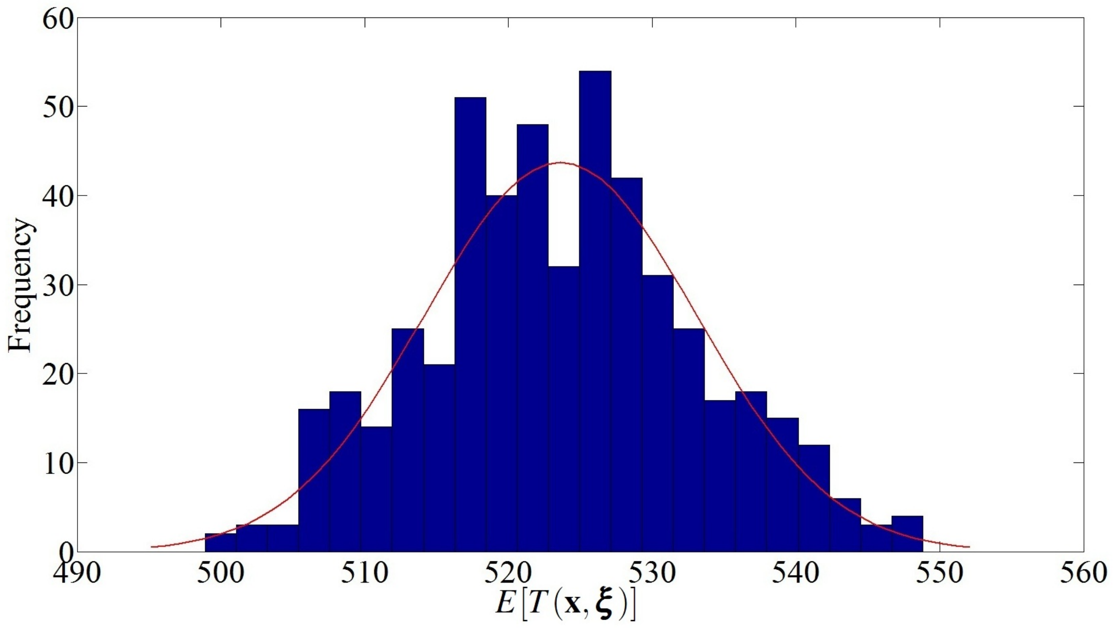


Figure 5.14: Series system histogram

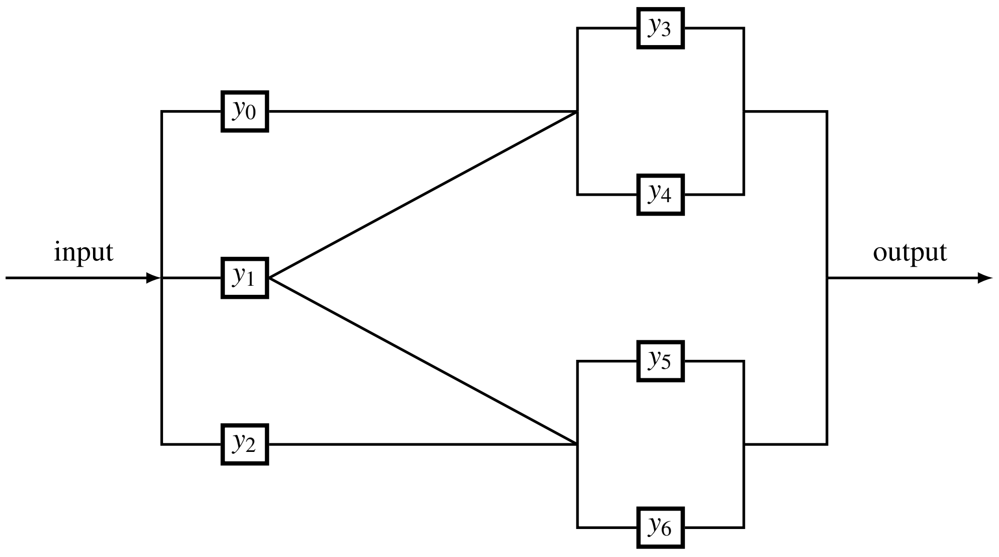


Figure 5.15: Life-support system in a space capsule.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

$N(312,25^2)$, $N(276,23^2)$, $N(350,26^2)$, $N(291,21^2)$, $N(271,24^2)$. All the redundant units are in parallel. The cost of each unit is: 56, 50, 64, 60, 79, 45, 28. The resulting cost function is: $C(\mathbf{x}) = 56x_0 + 50x_1 + 64x_2 + 60x_3 + 79x_4 + 45x_5 + 28x_6$. If the total capital available should not exceed 600, then the constraint becomes $C(\mathbf{x}) \leq 600$. The system structure function is expressed as: $\Psi(\mathbf{y}) = \{y_0 \cdot y_3,\ y_0 \cdot y_4, y_1 \cdot y_2, y_1 \cdot y_4,\ y_1 \cdot y_5, y_1 \cdot y_6,\ y_2 \cdot y_5,\ y_2 \cdot y_6\}$. The corresponding REVM is formulated as:

$$\begin{cases} \quad\quad \max \quad E[T(\mathbf{x},\boldsymbol{\xi})] \\ \text{subject to :} \\ \quad\quad\quad\quad C(\mathbf{x}) \leq 600 \\ \quad\quad\quad\quad x_i \geq 1, \quad \mathbf{x} \text{ is an integer vector} \end{cases}$$

The model is solved by first generating a training data-set for the following uncertain function

$$U : \mathbf{x} \to E[T(\mathbf{x},\boldsymbol{\xi})]$$

by stochastic simulation. Next, the NN (seven input neurons, 13 hidden neurons, one output neuron) is trained, to approximate $U(\mathbf{x})$. Finally, the NN is embedded in the GA. A run of the hybrid intelligent algorithm with the parameters specified in subsection 5.8.4 shows the optimal solution is:

$$\mathbf{x}^* = (2,1,1,1,3,1,1),$$

with an expected mean system lifetime corresponding to $\mathbf{x}^*$ of

$$E[T(\mathbf{x}^*,\boldsymbol{\xi})] = 379.204,$$

and a total corresponding cost of

$$C(\mathbf{x}^*) = 596.$$

The results of the solved model are compared to the results presented by [23] in Table 5.3 depicting satisfactory similarities.

Table 5.3: Comparison of results of the life-support system.

Output	Results obtained	Results obtained in [23]
Optimal decision vector	(2, 1, 1, 1, 3, 1, 1)	(2, 1, 1, 1, 3, 1, 1)
$E[T(\mathbf{x}^*,\boldsymbol{\xi})]$	379.204	371.95
$C(\mathbf{x}^*)$	596	596

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5.8.5 Multi-objective Optimisation Problem

A small communications network is presented in [70] that can be represented as the block diagram in Figure 5.16. By inspection, the system structure functions of interest are:

$$\Psi_1(\mathbf{y}) = \{y_0 \cdot y_3 \cdot y_5,\ y_1 \cdot\ y_4 \cdot y_5,\ y_2 \cdot y_6\}$$

for subsystem 1, and

$$\Psi_2(\mathbf{y}) = \{y_0 \cdot y_4 \cdot y_6\}$$

for subsystem 2. The lifetimes of these components are: $N(334, 24^2)$, $N(300, 26^2)$, $N(310, 24^2)$, $N(350, 24^2)$, $N(307, 23^2)$, $N(325, 24^2)$, $N(323, 24^2)$. The redundant units are in parallel. The costs of the components are: $N(105, 23^2)$, $N(123, 24^2)$, $N(100, 22^2)$, $N(125, 24^2)$, $N(98, 22^2)$, $N(102, 24^2)$, $N(85, 23^2)$. If the total capital available is 900, the goal is to maximise the expected system lifetime of both subsystems under the cost constraint.

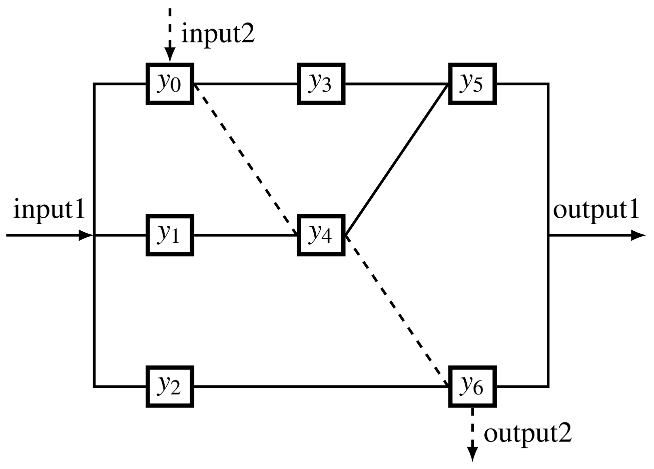


Figure 5.16: Small communication network.

The final Pareto-optimal set is plotted in Figure 5.17. The black dots represent the Pareto-optimal set of solutions. It is clear that the figure follows the trend for a maximise-maximise problem as depicted in Figure 2.8. The result in [23], however, only mentions one solution for the model. This solution was included in the plot in Figure 5.17 as the $'+'$, and it is evident that it lies on the Pareto front. In fact, it may be possible that the solution at (325.34, 297.45) returned by the simulation run corresponds to the solution presented in [23], which is (325.30, 297.95). Given that stochastic simulation is used, the estimated values would not be exact. The deviation can be attributed to different estimated lifetimes and different approximations by the NN. Furthermore, the decision vector producing the

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

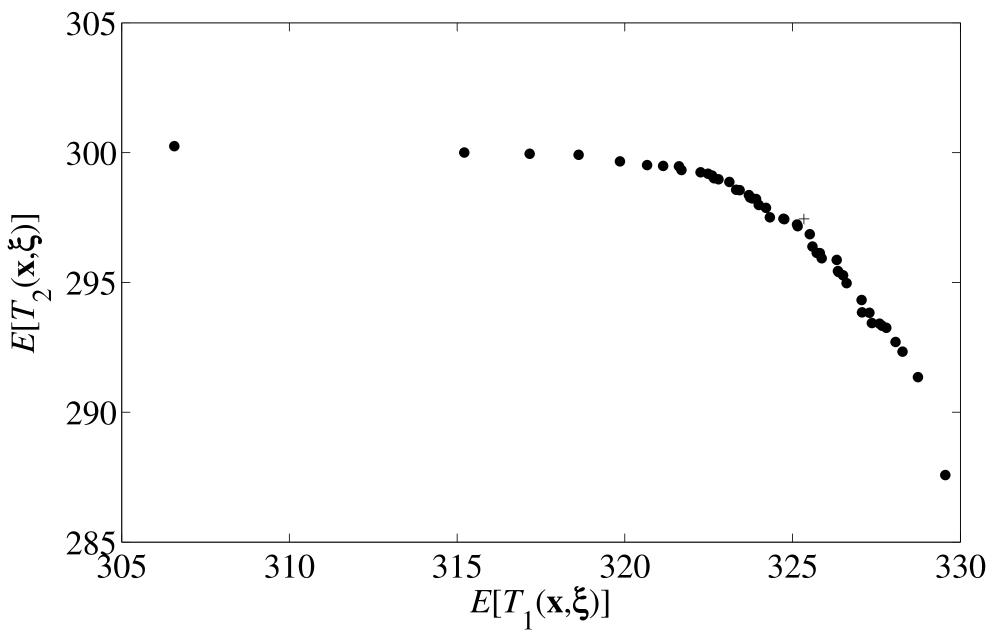


Figure 5.17: Pareto-optimal set of the communication system.

(325.34, 297.45) solution was $(1,1,1,1,1,2,1)$, which correlates to the decision vector produced in [23].

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 6

FINAL DESIGN EVALUATION

6.1 CHAPTER OVERVIEW

This chapter presents some examples of final designs of AMDF conceptual designs to illustrate that the goals of the research have been achieved. To this end, the two simulation models discussed in the previous chapters were applied to perform an evaluation of candidate AMDF designs. The abstract flowchart shown in Figure 6.1 describes the methodology. The OMNeT++ model determines the required equipment card redundancy to achieve a certain service availability (external redundancy). The redundancy optimisation model is used to allocate the optimal component redundancies in order to increase the expected system lifetime or reliability (internal redundancy). Both models provide a unified evaluation of an AMDF concept design. Once all the design specifications/requirements are defined (such as the capacity of the exchange, number of clients per module, expected service availability, cost, power and size constraints and designing with or without CM) the user proceeds by entering the parameters for both models. Next, the REVM or REVMOP are formulated using the same approach used in the preceding chapter.

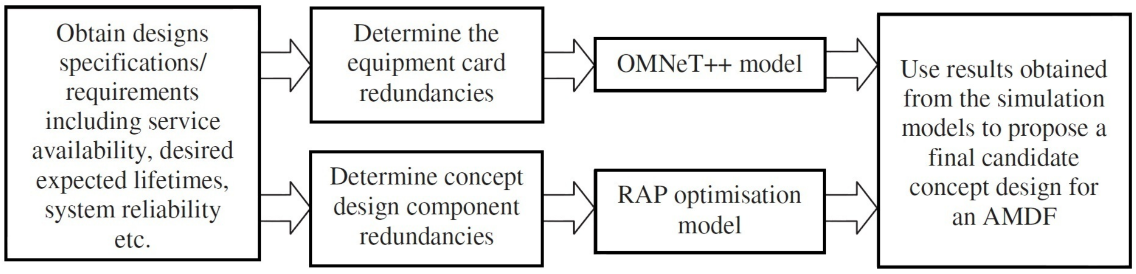

Obtain designs specifications/ requirements including service availability, desired expected lifetimes, system reliability etc.
Determine the equipment card redundancies
Determine concept design component redundancies
OMNeT++ model
RAP optimisation model
Use results obtained from the simulation models to propose a final candidate concept design for an AMDF

Figure 6.1: Abstract flow diagram depicting the methodology.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Finally, both simulations were run; the first model returned the number of additional ports the AMDF should make provision for in order to accommodate redundant equipment cards, and the second returned the optimal redundancy configuration in order to maximise the expected system lifetime or system reliability. The result is that multiple candidate designs can be compared and enables the user to make an informed choice for the best candidate design in terms of cost, power, size and the system performance metrics. No units are assigned to the parameters.

6.2 CANDIDATE CONCEPT DESIGN ONE

The abstract diagram in Figure 4.2 shows the elements of an AMDF. The block diagram depicted in Figure 6.2 was translated from the abstract diagram, without CM. The system consists of six elements. The power supply (y_0) may be modelled as a component in series with the system; if it fails, the whole system fails. The main controller (y_1) is in direct control of all AMDF modules, which contains the local module controller (y_3), the switch actuation control (y_4) and the actual switching matrix (y_5). Alternatively, y_3, y_4 and y_5 can collectively be modelled as y_7 to compare component level and system level redundancy allocations. The communication bus (y_2) is also modelled as a component in series, as a failure would result in an overall system failure. Redundancy is therefore allocated to the channel.

6.2.1 Assumptions

Table 6.1 list parameters/assumptions were made prior to simulation runtime.

Table 6.1: Assumptions and parameters.

Parameter / assumption	**Motivation / reason**
OMNeT++ model	
10000 total clients	Typical exchange size.
Clients expect 99.99 % availability	Study conducted in [20].
100 clients per AMDF module	Block sizes in employed MDFs.
Three service types per client	The services mostly used in South Africa: POTS, ADSL and ISDN.

Continued on next page

© University of Pretoria

Table 6.1 – continued from previous page

Parameter / assumption	**Motivation / reason**
Equipment card lifetimes: Weibull(12 months, 1, 2)	Weibull typically used for component failures. Assuming one year Mean Time to Failure (MTTF) and increasing hazard over time ($\beta > 1$).
Card replacing time: N(2 days, 1 day)	Arbitrary value.
Simulated time: 24 months (two years)	Arbitrary value.
Redundancy optimisation model	
Stochastic simulation	
5000 training data sets for NN	Used for six or more components in [23].
10000 cycles in simulation	Number of cycles used during evaluation in Chapter 5.
Genetic Algorithm	
Population size = 30	As convention in [23].
Crossover rate, P_c = 30 %	As convention in [23].
Mutation rate, P_m = 20 %	As convention in [23].
a = 0.05	As convention in [23].
Maximum generations = 300	As convention in [23].
Constraints	
Cost = 2000	Arbitrary value.
Power = 500	Arbitrary value.
Size = N/A	Not applicable.
Weight = N/A	Not applicable.
Maximum generations = 300	As convention in [23]
Component properties	
y_0- Cost: $N(80,10^2)$; Lifetime: $N(360,150^2)$; Power: N/A, Standby redundancy	
y_1- Cost: $N(500,50^2)$; Lifetime: $N(730,200^2)$; Power: $N(300,30^2)$, Standby redundancy	
y_2- Cost: $N(20,4^2)$; Lifetime: $N(700,100^2)$; Power: $N(10,10^2)$, Standby redundancy	
y_3- Cost: $N(150,20^2)$; Lifetime: $N(1000,200^2)$; Power: $N(30,10^2)$, Standby redundancy	
y_4- Cost: $N(100,10^2)$; Lifetime: $N(720,100^2)$; Power: $N(25,10^2)$, Standby redundancy	
y_5- Cost: $N(500,20^2)$; Lifetime: $N(800,130^2)$; Power: $N(1,0.1^2)$, Standby redundancy	

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

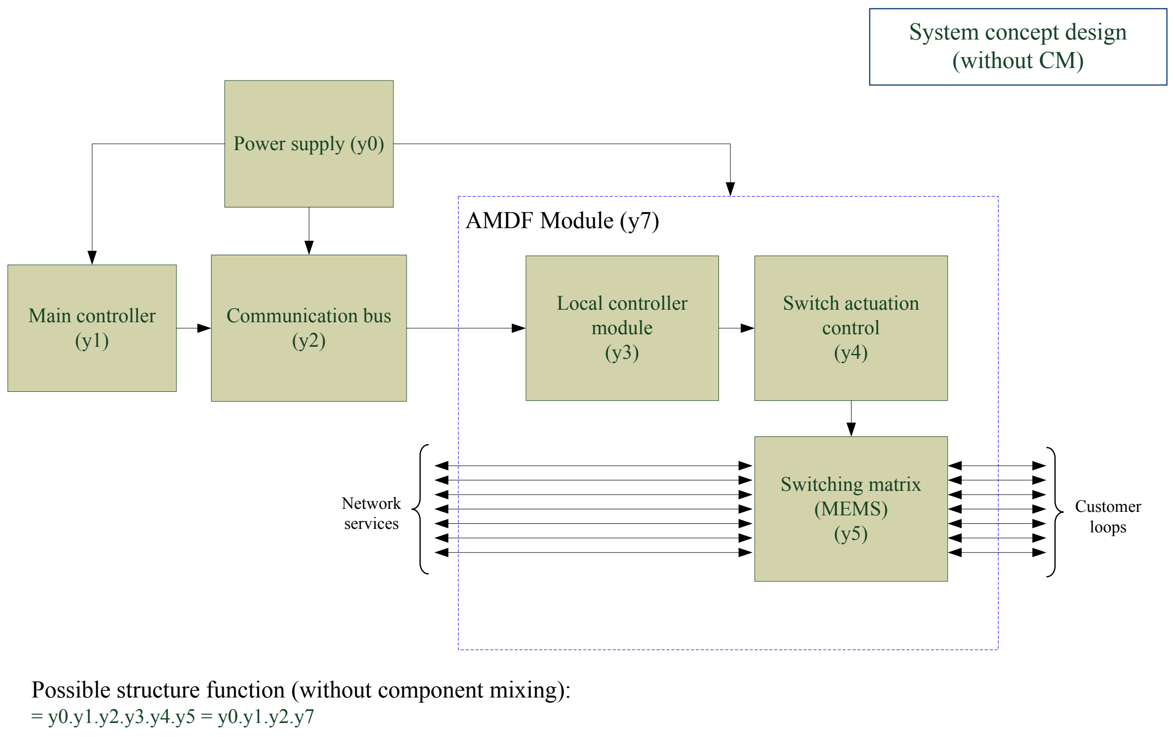

System concept design (without CM)
Power supply (y0)
AMDF Module (y7)
Main controller (y1)
Communication bus (y2)
Local controller module (y3)
Switch actuation control (y4)
Switching matrix (MEMS) (y5)
Network services
Customer loops
Possible structure function (without component mixing):
= y0.y1.y2.y3.y4.y5 = y0.y1.y2.y7

Figure 6.2: Example of a concept design without CM block diagram.

6.2.2 Questions

As an example, the following questions were considered to illustrate the capability of incorporating both models to converge to a final design:

- OMNeT++ model:
  1. What equipment card redundancy percentage yields at least 99.99 % service availability per client per year (52.5 minutes downtime per year)?
- Redundancy optimisation model:
  1. What is the optimal redundancy allocation to maximise expected lifetime under the given constraints?
  2. What is the corresponding expected lifetime?
  3. What is the corresponding expected cost?
  4. What is the corresponding power consumption?

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

For redundancy optimisation, the following REVM can be formulated as the model to be solved (system structure function is shown in Figure 6.2):

$$\begin{cases} \max \quad E[T(\mathbf{x}, \boldsymbol{\xi})] \\ \text{subject to :} \\ \qquad \sum_{i=0}^{n} E[c_i] \cdot x_i \leq 2000 \\ \qquad \sum_{i=0}^{n} E[p_i] \cdot x_i \leq 500 \\ \qquad x_i \geq 1, \qquad \mathbf{x} \text{ is an integer vector} \end{cases}$$

6.2.3 Findings

The answers were documented and tabulated in Table 6.2.

Table 6.2: Findings for the first candidate design.

Question	Result
OMNeT++	
What equipment card redundancy percentage yields at least 99.99 % service availability?	Equipment card redundancy percentage between 33 % and 34 % yields 99.99 % service availability. The AMDF should make provision for at least 34 % redudancy for equipment cards.
Redundancy optimisation model	
Optimal redundancy allocation?	$\mathbf{x}^* = (4,1,5,2,2,1)$
Corresponding expected lifetime?	940
Corresponding cost?	1966.51
Corresponding power consumption?	427.29

6.2.4 Discussion

According to the OMNeT++ model, at least 34 % equipment card redundancy will result in a service availability greater than 99.99 %. Since it was one of the key assumptions that clients expect at least 99.99 % service availability, the redundant equipment cards are mandatory to avoid customer dissatisfaction. For the 10,000 clients simulated, an additional 3,400 equipment cards must be procured. This is quite a large number of redundant equipment cards and may prove problematic when accommodating larger exchanges in metropolitan areas.

The redundancy optimisation model yielded a configuration that results in an expected lifetime of 940.058 (just over two and a half years if lifetimes were provided in days). With four redundant

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

power supplies, it would be highly unlikely that the system would fail as a result of malfunctioned power supplies. With only one main controller, five communication buses seems to be overkill for the sole purpose of establishing communication between components. Nevertheless, since the system was still well under the constraints, an excess of communication buses were added due to their relative low cost and power consumption. Inside the AMDF module, it seems logical to include two redundant elements for the local controller and switch actuation control components. If one fails, the other should immediately be able to take over.

In conclusion, the results suggest that the majority of resources will be consumed in fulfilling the 34 % redundancy percentage for equipment cards rather than internal redundancies.

6.3 CANDIDATE CONCEPT DESIGN TWO

The experiment is repeated with all the parameters in the previous section, with the standby redundancy replaced by active redundancy for components. However, it is evident from inspection that it would be infeasible to choose an active redundancy scheme for the power supply. Because of the relatively low expected lifetime in comparison with the other components, no number of redundant elements would result in an expected lifetime that would exceed the expected lifetimes of the other components. Although standby redundancies for power supplies are atypical, standby power supplies are purposely included to compare the effect of redundancy schemes on the components. In this case, the redundancy optimisation model would strive to maximise the expected lifetime of the element with the shortest lifetime, consuming resources and prematurely exceeding system constraints. To account for this, the standby redundancy scheme is still assigned to the power supply. The remainder of the components are assigned an active redundancy scheme. Since no changes had been made to the OMNeT++ model, it was expected that the results would remain unchanged. However, the change in component properties required that the REVM be re-evaluated. The REVM of this candidate design is identical to the REVM used in the previous section:

$$\begin{cases} \quad \max \quad E[T(\mathbf{x},\boldsymbol{\xi})] \\ \text{subject to :} \\ \qquad \sum_{i=0}^{n} E[c_i]\cdot x_i \leq 2000 \\ \qquad \sum_{i=0}^{n} E[p_i]\cdot x_i \leq 500 \\ \qquad x_i \geq 1, \qquad \mathbf{x} \text{ is an integer vector} \end{cases}$$

© University of Pretoria

Table 6.3: Findings for the second candidate design.

Question	Result
Redundancy optimisation model	
Optimal redundancy allocation?	$\mathbf{x}^* = (3,1,5,3,2,1)$
Corresponding expected lifetime?	689.152
Corresponding cost?	1990.51
Corresponding power consumption?	491.57

Table 6.3 shows the results for the second candidate design. The optimal solution produced is similar to that in the previous section. In essence, one power supply is sacrificed to add another local controller. The overall effect of replacing the standby redundant elements with parallel redundant ones is seen in the expected lifetime, which, not surprisingly, is much lower than the expected lifetime obtained in the previous section.

6.4 CANDIDATE CONCEPT DESIGN THREE

Another possibility, one with CM, is depicted in Figure 6.3. The case where CM is used, results in more extensive system structure functions. In this section it is assumed that two component types of the local controller are available to illustrate the effect on the system structure function with CM. Fortunately, by using the bisection search algorithm, the task of handling these structure functions is alleviated. Since an additional component type is added, the system structure function changes to (see Figure 6.3 for other examples):

$$\Psi(\mathbf{y}) = \{y_0 \cdot y_1 \cdot y_2 \cdot y_3 \cdot\ y_4 \cdot y_5,\ y_0 \cdot y_1 \cdot y_2 \cdot y_3' \cdot\ y_4 \cdot y_5\}$$

Similar properties were assigned to y_3' than to y_3, with slight differences: Cost: $N(130, 20^2)$; Lifetime: $N(900, 200^2)$; Power: $N(25, 10^2)$ Standby. If a standby redundancy scheme is assumed for all components and the same REVM applies, then Table 6.4 shows the results.

Table 6.4: Findings for the third candidate design.

Question	Result
Redundancy optimisation model	
Optimal redundancy allocation?	$\mathbf{x}^* = (3,2,4,1,1,2,1)$
Corresponding expected lifetime?	1161.03
Corresponding cost?	1987.52
Corresponding power consumption?	456.52

A similar result is obtained than in Section 6.2. In this case, one power supply and one switch

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

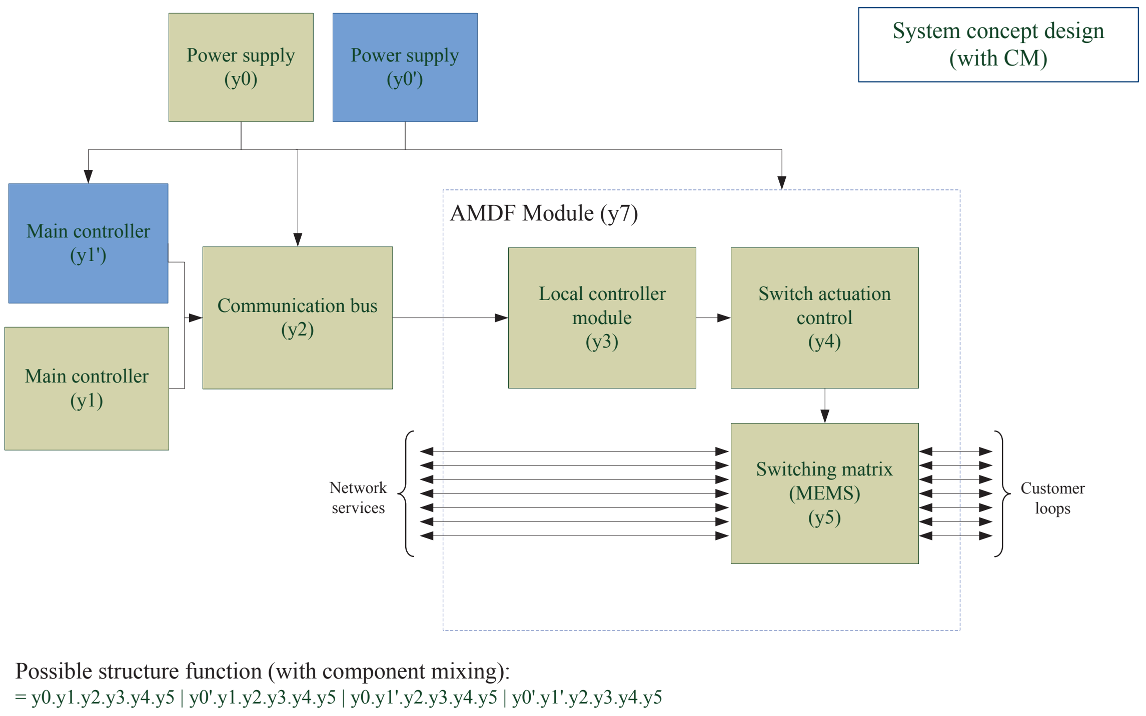


Figure 6.3: Example of a concept design with CM block diagram.

actuation controller is sacrificed to add another main controller. Furthermore, it is noticed that there are still two local controllers present, one of each type.

6.5 CANDIDATE CONCEPT DESIGN FOUR

The final candidate design focuses on optimising certain parts, or domains, of the block diagram. To this end, the user is required to specify the subsystems that are of particular interest, to be evaluated concurrently. In context of the proposed AMDF design shown in Figure 6.3 and Figure 6.2, a possible specification may be to group components y_3 (and subsequently $y_3^{'}$), y_4 and y_5 into one subsystem (denoted by y_7). The resulting model to be solved is a multi-objective optimisation problem concerned with maximising the expected lifetimes of all defined subsystems. Assuming the same parameters as

© University of Pretoria

presented in Section 6.2, the model is used to solve the following REVMOP:

$$\begin{cases} \quad \max \quad E[T_1(\mathbf{x},\boldsymbol{\xi}),T_2(\mathbf{x},\boldsymbol{\xi})] \\ \text{subject to :} \\ \qquad \sum_{i=0}^{n} E[c_i]\cdot x_i \leq 2000 \\ \qquad \sum_{i=0}^{n} E[p_i]\cdot x_i \leq 500 \\ \qquad \mathbf{x} \geq 1, \qquad \text{integer vector} \end{cases}$$

where the system structure function of subsystem 1 (domain 1) is given by

$$\Psi_1(\mathbf{y}) = \{y_0 \cdot y_1 \cdot y_2\}$$

and the system structure function of subsystem 2 (domain 2) is

$$\Psi_2(\mathbf{y}) = \{y_3 \cdot y_4 \cdot y_5\}.$$

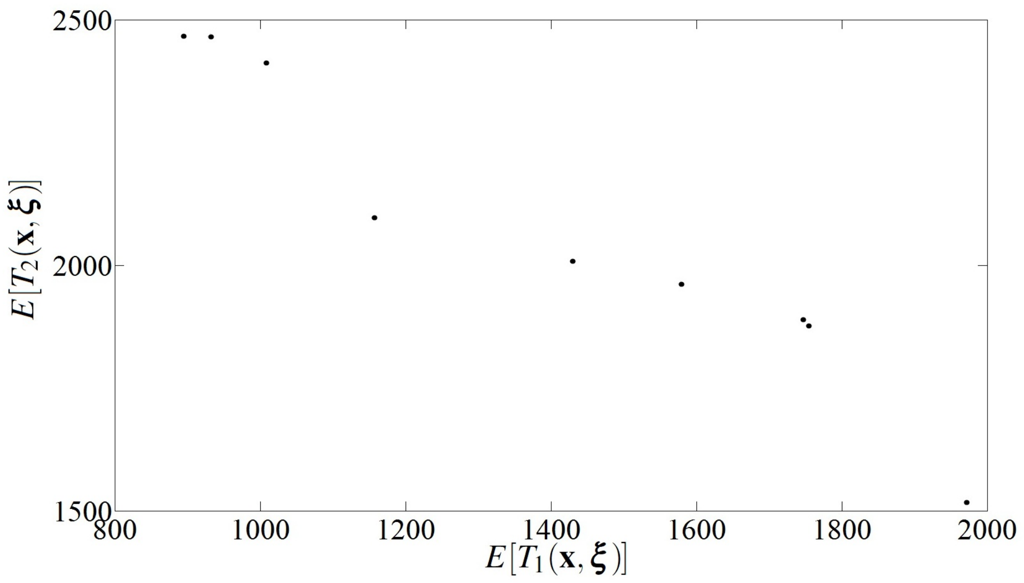


Figure 6.4: Final Pareto set of the fourth AMDF candidate concept design.

Figure 6.4 shows the Pareto-optimal set. It is now up to the decision-maker to select one of the solutions in the Pareto set as the optimal solution. For instance, if the second subsystem (the AMDF module in this case) assumes a higher priority than the rest of the system (first subsystem), then the solutions at the top left corner in Figure 6.4, i.e. a solution yielding a higher value for $E[T_2(\mathbf{x},\boldsymbol{\xi})]$,

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

should be considered. The decision vector of the left-most solution is: $\mathbf{x}^* = (3,1,2,3,3,1)$. It is evident that the components in subsystem 2 are assigned a higher number of redundant elements than in the other models in the preceding sections. This is expected, since the higher priority was assigned to maximising the lifetime of the second subsystem. Moreover, note that a more realistic number of redundant elements for the communication bus are returned.

6.6 CANDIDATE CONCEPT DESIGN COMPARISON

Table 6.5 shows a comparison of the optimal decision vectors obtained. The similarities between the candidate concept designs are apparent. The actual solution, however, depends on the constraints, the employed redundancy scheme, whether the design is implemented with or without CM, and the priority structure of subsystems. It is up to the decision-maker to decide which model is best for an application.

Table 6.5: Comparison of AMDF candidate concept design results.

	Design one	**Design two**	**Design three**	**Design four**
$E[T(\mathbf{x},\boldsymbol{\xi})]$	940	689.152	1161.03	943.674
$\mathbf{x}^*$	$(4,1,5,2,2,1)$	$(3,1,5,3,2,1)$	$(3,2,4,1,1,2,1)$	$(3,1,2,3,3,1)$
Cost	1966.51	1990.51	1987.52	1996.12
Power	427.29	491.57	456.52	471.34

Other factors that could have been incorporated include the size/weight constraints, which are important in terms of scalability and physical layout. In addition, as more objectives are added to the optimisation problem, the nature of the Pareto is radically changed. If the optimisation problem was defined with three objectives, the Pareto set would become a 3D surface.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

CHAPTER 7

CONCLUSION

7.1 OVERVIEW

In Chapter 1, the specific objectives of this dissertation were listed. These objectives were addressed during the course of this dissertation, and the outcomes are concluded, referring to the relevant sections or chapters. Lastly, possible improvements to the model and recommendations for future work are discussed, which is followed by a closing remark.

7.2 CONCLUSIONS ON THE OBJECTIVES

The literature regarding the design and implementation of AMDFs is very limited. In this dissertation, a simulation framework was presented to aid designers in evaluating *i*) the dynamic client-AMDF interaction, and *ii*) system performance metrics of AMDF candidate concept designs, as stated in Section 1.4. The presented work partially fills the void in the literature.

The purpose of the OMNeT++ dynamic behaviour model was to address the cost-effective scalability issue of modular AMDFs. In addition, current AMDFs do not incorporate equipment card redundancies. The model was used to determine equipment card redundancies in order to achieve a desired service availability. However, the results obtained from the OMNeT++ model could not be compared to the results of other models, since no other models performing the same simulation function exist. The work presented in [23] was the inspiration for incorporating stochastic simulation to solve the RAP for both series, parallel and complex systems. The purpose of the redundancy optimisation simulation model was to incorporate optimisation techniques to solve the RAP effectively in order to address the reliability issue regarding AMDFs. Numerical experiments were performed to assess the validity of each model, which was compared to results of other publications.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

7.3 FRAMEWORK IMPLEMENTATION DISCUSSION

For the first model, a fully functional model of a modular AMDF was constructed using OMNeT++. A brief overview of OMNeT++ was provided in Chapter 3 and a comparison was made with a variety of other network simulators. The model presented in Chapter 4 allows the user to investigate the dynamic behaviour of a modular AMDF and the effect of certain requirements such as client division between modules, automatic cross-connection jumpering, automatic fault detection and correction, equipment redundancy, high availability and downtimes. The model is highly configurable in order to allow the effect of various configurations to be investigated in different scenarios. Parallel simulation is implemented via MPI using the NMA as the synchronisation scheme. The optimal number of CPU cores and inter-partition link delays was investigated to maximise speedup. The speedup obtained with the parallelised model allows various configurations to be evaluated within significantly reduced time frames. The model was used to investigate and compare various candidate designs prior to prototype development. The model presented is supplemented by the redundancy optimisation model.

For the second model, stochastic simulation is incorporated with a hybrid intelligent algorithm to solve RAPs in Chapter 5. A spectrum of examples was used to verify its functionality, which includes estimating expected lifetimes, α-system lifetimes, and system reliabilities for series, parallel and complex configurations. Ultimately, the model was capable of handling any type of system structure function, and was able to train NNs from relatively large training data sets that was generated by stochastic simulation. The model also exhibits the ability to solve both single-objective and multi-objective optimisation problems successfully. Repeatability is paramount for any simulation run, and the model exhibits excellent repeatability of all the simulation results. A number of practical examples of complex systems have also been evaluated and the results were compared to the results obtained in other publications. The correct procedure for using the model is presented in a well-structured flowchart in Chapter 5.

Finally, both models were encapsulated in a unified simulation framework in Chapter 6, which was used to perform a complete evaluation of proposed AMDF candidate designs. Observations regarding the effect of various properties, such as designs with CM and without CM and the actual redundancy strategies employed, were investigated and discussed in Chapter 6.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

7.4 IMPROVEMENTS TO THE MODELS

For the first model, no real improvement to aid in the analysis of the dynamic behaviour of modular AMDFs can be suggested. However, the current model can be supplemented in terms of adding other statistics to be collected during the simulation run, depending on designer preference. Moreover, new entities can be implemented, along with he new statistics to be collected, to perform additional tasks during the simulation run.

For the second model, multi-state components can be incorporated, which may include components with various modes of operation and failure, which in turn may impact allocated redundancy configurations. Such a modification would radically change the method by which the system structure function is evaluated. Furthermore, components can also be classified as repairable, or non-repairable. In such a case, maintenance is introduced in the model, having either preventive or corrective maintenance. Preventative maintenance, as the name suggests, is used to attend to a problems to prevent faults from occurring. Conversely, corrective maintenance is performed after a failure. The inclusion of maintenance would have additional cost implications on the design and could have an effect on the service availability. Furthermore, the constraints can be extended to include, for example, space requirements and thermal aspects.

7.5 RECOMMENDATIONS FOR FUTURE WORK

In this work, the two models presented were developed using OMNeT++ for the first, and stochastic simulation integrated with a hybrid intelligent algorithm for the second. As an alternative, the first model can be implemented with other network simulators, such as NS-2 or OPNET. The produced results using the other network simulators can then be compared with those produced by the OMNeT++ model, as well as other aspects, such as simulation execution time and possible speedups. The simulation model is very specific to the design of an AMDF. Therefore, such a model must also be developed from the ground up.

For the second model, this research focused on maximising the expected lifetimes of concept designs. The model, however, can already compute the remaining two system performance metrics. Future work regarding system performance metrics may be to incorporate objective functions for system reliability, or α-system lifetime computations. Subsequently, multiple objectives regarding different system performance metrics may be considered, e.g. simultaneously maximising expected lifetime

© University of Pretoria

and system reliability.

7.6 CLOSING REMARK

In practice, it is difficult to assess the efficiency and feasibility of a concept design without field data or performing extensive experiments. Simulation modelling is a popular method of predicting the performance of a concept design prior to development. The presented framework will greatly aid service provider businesses in effectively managing current subscriber loop networks and assist with future network planning. Some may argue that emerging wireless technologies are destined to replace wireline communication networks; however, upcoming fibre optics technologies may provide affordable options for Fibre-to-the-Building (FTTB) and Fibre-to-the-Home (FTTH) to replace copper lines. Recently, it was announced that a multi billion-rand project has been launched to connect over 2.5 million homes in six major cities in South Africa with a high-speed fibre network [71]. The current framework can also be used to manage fibre optic lines and planning fibre optic networks; it is required of the designer to provide the appropriate parameters for the models. The only significant impact fibre optics may have on the design is replacing the copper switching media with the appropriate switches for fibre optics.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

REFERENCES

[1] Web ProForum Tutorials. (2007, Jun.) The International Engineering Consortium. [Online]. Available: http://www.iec.org/online/tutorials/man_copper/

[2] Telkom S. A. Limited. (2010, Nov.) DSL Conditions. [Online]. Available: http://www.telkom.co.za/products_services/dsl/conditions.html

[3] T. Kanai, S. Umemura, S. Inagaki, and S. Hosokawa, "High Density Pin Board Matrix Switches for Automated MDF Systems," *IEEE Trans. Compon., Hybrids, and Manuf. Technol.*, vol. 15, no. 5, pp. 893–903, Nov. 1992.

[4] K. Yoshida, T. Hirono, S. Hosokawa, and A. Nagayama, "A New Automated Main Distributing Frame System using Robot," in *1991 IEEE Int. Conf. on Commun.*, vol. 2, 1991, pp. 977–982.

[5] E. Kovac and W. Mitchell, "The DCS as a Universal Digital Cross-Connect System," *IEEE J. Sel. Areas Commun.*, vol. 5, no. 1, pp. 53–58, Jan. 1987.

[6] S. Umemura, T. Kanai, S. Inagaki, and Y. Kumakura, "Design of High Density Pin Board Matrix Switches for Automated Main Distributing Frame Systems," *IEEE Trans. Compon., Hybrids, and Manuf. Technol.*, vol. 15, no. 2, pp. 266–277, Apr. 1992.

[7] J. Teixeira, "Flow-Through Using An Automated Distribution Frame," U.S. Patent 10,429,861, Jul. 15, 2004.

[8] R. L. Freeman, *Telecommunication System Engineering*, 4th ed. New York, NY, USA: John Wiley & Sons, Inc., 2004.

[9] S. Inagaki, T. Kanai, and S. Hosokawa, "High-density pin-board matrix switch and its crosstalk characteristics," *Electron. and Commun. in Japan (Part I: Commun.)*, vol. 78, no. 7, pp. 42–45,

© University of Pretoria

Jul. 1995.

[10] H. Fukuda, "Automatic MDF Apparatus," U.S. Patent 5,870,528, Feb. 9, 1999.

[11] S. Hosokawa, "Automated Optical MDF System," Europe Patent 0,494,768, Feb. 9, 1992.

[12] T. Kanai, A. Nagayama, S. Inagaki, and K. Sasakura, "Automated Optical Main-Distributing-Frame System," *J. of Lightwave Technology*, vol. 12, no. 11, pp. 1986–1992, Nov. 1994.

[13] S. Braun, J. Oberhammer, and G. Stemme, "MEMS Single-Chip 5×5 and 20×20 Double-switch Arrays for Telecommunication Networks," in *2007 IEEE 20th Int. Conf. Microelectromech. Syst.*, 2007, pp. 811–814.

[14] S. Braun, J. Oberhammer and G. Stemme, "MEMS Single-Chip Microswitch Array for Reconfiguration of Telecommunication Networks," in *2006 Proc. of the 36th European Microwave Conf.*, 2006, pp. 811–814.

[15] D. Czaplewski, G. Patrizi, G. Kraus, J. Wendt, C. Nordquist, S. Wolfley, M. Baker, and M. de Boer, "A Nanomechanical Switch for Integration with CMOS Logic," *Journal of Micromechanics and Microengineering*, vol. 19, no. 8, p. 085003, Jul. 2009.

[16] "Micro Magnetic Latching Switches For Automated Crossconnect Systems," White paper, Telepath Networks, Inc. [Online]. Available: http://www.telepathnetworks.com/s.nl/sc.5/category.36/.f

[17] D. Cuda, P. Giaccone, and M. Montalto, "Design and Control of Next Generation Distribution Frames," in *2011 IEEE Int. Conf. High Performance Switching and Routing*, Nashville, TN, Jul. 2011, pp. 115–120.

[18] C. Simons, T. S. Pearson, C. R. Noel, and J. D. Kidder, "Network Device for Supporting Multiple Redundancy Schemes," U.S. Patent 6,332,198, Dec. 18, 2001.

[19] J. Gray and D. P. Siewiorek, "High-availability computer systems," *Computer*, vol. 24, no. 9, pp. 39–48, Nov. 1991.

[20] M. Huynh, S. Goose, , and P. Mohapatra, "Resilience Technologies in Ethernet," *Computer Networks*, vol. 54, no. 1, pp. 57–58, Jan. 2010.

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

[21] D. W. Coit, "Cold-Standby Redundancy Optimization for Nonrepairable Systems," *IIE Transactions*, vol. 33, no. 6, pp. 471–478, 2001.

[22] L. R. Goel and R. Gupta, "Reliability Analysis of Multi-Unit Cold Standby System with Two Operating Modes," *Microelectronic Reliability*, vol. 23, no. 6, pp. 1045–1050, 1983.

[23] R. Zhao and B. Liu, "Stochastic Programming Models for General Redundancy-Optimization Problems," *IEEE Trans. Rel.*, vol. 52, no. 2, pp. 181–191, Jun. 2003.

[24] S. W. Ormon, C. R. Cassady, and A. G. Greenwood, "Reliability Prediction Models to Support Conceptual Design," *IEEE Trans. Rel.*, vol. 51, no. 2, pp. 151–157, Jun. 2002.

[25] J. Safari and R. Tavakkoli-Moghaddam, "A Redundancy Allocation Problem with the Choice of Redundancy Strategies by a Memetic Algorithm," *J. Ind. Eng. Int.*, vol. 6, no. 11, pp. 6–16, 2010.

[26] C. Y. Lee, M. Gen, and W. Kuo, "Reliability Optimization Design using a Hybridized Genetic Algorithm with a Neural-Network Technique," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E84-A, no. 2, pp. 627–637, Jul. 2001.

[27] S. Cheng, "Ant Colony Algorithm to Reliability Optimization in Complex System," in *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications*, Sep. 2010, pp. 318–322.

[28] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probabilities for Engineers*, 4th ed. NJ, USA: John Wiley & Sons, Inc., 2007.

[29] D. Coit and A. Smith, "Optimization Approaches to the Redundancy Allocation to the Redundancy Allocation Problem for Series-parallel Systems," in *1995 Proceedings of the Fourth Industrial Engineering Research Conference*, Nashville, TN, 1995, pp. 342–349.

[30] M. S. Chern, "On The Computational Complexity of Reliability Redundancy Allocation in a Series System," *Operations Research Letters*, vol. 11, no. 5, pp. 309–315, 1992.

[31] B. Liu, *Theory and Practice of Uncertain Programming*, 2nd ed. Heidelberg: Physica-Verlag, 2002.

© University of Pretoria

[32] Y. Liang and A. E. Smith, "An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP)," *IEEE Trans. Rel.*, vol. 53, no. 3, pp. 417–423, Sep. 2004.

[33] W. Yeh and T. Hsieh, "Solving reliability redundancy allocation problems using an artificial bee colony algorithm," *Computers & Operations Research*, vol. 38, no. 11, pp. 1465–1473, Nov. 2010.

[34] D. Karaboga and B. Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–71, 2007.

[35] I. F. Sbalzarini, S. Müller, and P. Koumoutsakos, "Multiobjective Optimization using Evolutionary Algorithms," in *Proceedings of the Center for Turbulence Research*, 2000, pp. 63–74.

[36] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.

[37] C. A. Coello Coello, "Evolutionary Multi-Objective Optimization: A Historical View of the Field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, Feb. 2006.

[38] E. Zitzler, M. Laumanns, and S. Bleuler, "A Tutorial on Evolutionary Multiobjective Optimization," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, Eds. Springer Berlin Heidelberg, 2004, vol. 535, pp. 3–37.

[39] N. Srinivas and K. Deb, "Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[40] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *1994 Proc. IEEE Conf. Evol. Comput.*, Jun. 1994, pp. 82–87 vol.1.

[41] C. Fonseca and P. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *1993 Proceedings of the Fifth International Conference on Genetic Algorithms*, May 1993, pp. 416–423.

[42] J. H. Holland, "Adaptation in Natural and Artificial Systems," *Ann Arbor: University of*

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Michigan Press, 1977.

[43] M. Gen and Y. S. Yun, “Soft Computing Approach for Reliability Optimization: State-of-the-art Survey,” *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 1008–1026, Sep. 2006.

[44] A. Konak, D. W. Coit, and A. E. Smith, “Multi-Objective Optimization using Genetic Algorithms: A Tutorial,” *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, Jan. 2006.

[45] A. E. Rizzoli. (2010, Nov.) A Collection of Modelling and Simulation Resources on the Internet. [Online]. Available: http://www.idsia.ch/~andrea/sim/simnet.html

[46] E. Weingartner, H. vom Lehn, and K. Wehrle, “A Performance Comparison of Recent Network Simulators,” in *2009 IEEE Int. Conf. Commun.*, Dresden, Jun. 2009, pp. 1–5.

[47] A. Varga and R. Hornig, “An Overview of the OMNeT++ Simulation Environment,” in *2008 Proc. of the 1st Int. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Marseille, France, 2008.

[48] A. Varga, “The OMNeT++ Discrete Event Simulation System,” in *2001 Proc. of the European Simulation Multiconference*, 2001.

[49] A. Varga, “Using the OMNeT++ Discrete Event Simulation System in Education,” *IEEE Trans. Educ.*, vol. 42, no. 4, p. 11, Nov. 1999.

[50] K. Fall and K. Vardhan. (2010, Nov.) The Network Simulator - ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[51] [No Author]. (2011) The Network Simulator - ns-3. [Online]. Available: http://www.nsnam.org/

[52] OPNET Technolgies. (2010, Nov.) OPNET Modeler. [Online]. Available: http://www.opnet.com/

[53] Tetcos. (2009) PhySim: Communication Simulator and trainer. [Online]. Available: http://tetcos.com/software.html

[54] X. Zeng, R. Bagrodia, and M. Gerla, “GloMoSim: A library for parallel simulation of large-

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

scale wireless networks," in *1998 Twelfth Workshop on Parallel and Distributed Simulation*, Dresden, May 1998, pp. 154–161.

[55] Scalable Network Technologies, Inc. (2011) Qualnet Network Simulator. [Online]. Available: http://www.qualnet.com

[56] Tetcos. (2009) Physim: Communication simulator and trainer. [Online]. Available: http://tetcos.com/physim.html

[57] T. J. Schriber and D. T. Brunner, "Inside Discrete-Event Simulation Software: How it works and why it matters," in *2005 Proceedings of the Winter Simulation Conference*, Dec. 2005, pp. 113–123.

[58] A. Varga, OMNeT++ User Manual 4.1, 2010.

[59] Y. A. Şekercioğlu, A. Vargas, and G. K. Egan, "Parallel Simulation Made Easy With OMNeT++," in *2003 Proceedings of the European Simulation Symposium*, Delft, The Netherlands, Oct. 2003.

[60] M. I. Botha and H. Grobler, "A Simulation Framework for Evaluating the Behaviour of a Modular Automated Main Distribution Frame using OMNeT++," in *2011 Proceedings of the Southern Africa Telecommunication Networks and Applications Conference*, East London, South Africa, Sep. 2011.

[61] A. Varga, Y. A. Şekercioğlu, , and G. K. Egan, "A Practical Efficiency Criterion for the Null Message Algorithm," in *2003 15th European Simulation Symposium*, Delft, The Netherlands, Sep. 2003.

[62] H. Singh and N. Misra, "On Redundancy Allocations in Systems," *Journal of Applied Probability*, vol. 31, no. 4, pp. 1004–1014, Dec. 1994.

[63] P. J. Boland, F. Proschan, and Y. L. Tong, "Standby Redundancy Policies for Series System," Department of Statistics, Florida State University, Tallahassee, Florida, Tech. Rep. M-383, Jan. 1991.

[64] A. K. Bhunia, L. Sahoo, and D. Roy, "Reliability Stochastic Optimization for a Series System

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

with Interval Component Reliability via Genetic Algorithm," *Applied Mathematics and Computation*, vol. 213, no. 3, pp. 929–939, Apr. 2010.

[65] S. Xu and L. Chen, "A Novel Approach for Determining the Optimal Number of Hidden Layer Neurons for FNN's and Its Application in Data Mining," in *2008 5th International Conference on Information Technology and Applications*, San Diego, CA , USA, Jun. 2008, pp. 683–686.

[66] T. Murata and H. Ishibuchi, "MOGA: Multi-Objective Genetic Algorithms," in *1995 IEEE Int. Conf. Evol. Comput.*, vol. 1, Nov. 1995, pp. 289–294.

[67] V. Ravi, B. S. N. Murty, and P. J. Reddy, "Nonequilibrium Simulated Annealing-Algorithm Applied to Reliability Optimization of Complex Systems," *IEEE Trans. Rel.*, vol. 46, pp. 233–239, 1997.

[68] K. Aggarwal, J. Gupta, and K. Misra, "A Simple Method for Reliability Evaluation of a Communication System," *IEEE Trans. Commun.*, vol. 23, no. 5, pp. 563–566, May 1975.

[69] S. B. Twum, "Multicriteria Optimisation in Design for Reliability," Ph.D. dissertation, University of Birmingham, 2009.

[70] K. Fung, "A Philosophy for Allocating Component Reliabilities in a Network," *IEEE Trans. Rel.*, vol. R-34, no. 2, pp. 151–153, Jun. 1985.

[71] D. McLeod. (2011, March) Mega project to bring fibre to SA homes. [Online]. Available: http://www.techcentral.co.za/exclusive-megaproject-to-bring-fibre-to-sa-homes/21723/

© University of Pretoria