
Pós-Graduação em Ciência da Computação

# “Assessment to support the planning of sustainable data centers with high availability”

**By**

***Gustavo Rau de Almeida Callou***

**PhD Thesis**

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, NOVEMBER/2013

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO RAU DE ALMEIDA CALLOU

# "Assessment to support the planning of sustainable data centers with high availability"

*A THESIS SUBMITTED TO THE POST GRADUATION IN COMPUTER SCIENCE OF THE INFORMATICS CENTER OF FEDERAL UNIVERSITY OF PERNAMBUCO IN PARTIAL FULFILLMENT OF REQUIREMENTS FOR THE PHD DEGREE IN COMPUTER SCIENCE.*

Adviser: Prof. Dr. Paulo Romero Martins Maciel

RECIFE, NOVEMBER/2013

**Catalogação na fonte**
**Bibliotecária Jane Souto Maior, CRB4-571**

**Callou, Gustavo Rau de Almeida**
**Assessment to support the planning of sustainable data centers with high availability / Gustavo Rau de Almeida Callou. - Recife: O Autor, 2013.**
**xix, 142 f., il., fig., tab.**

**Orientador: Paulo Romero Martins Maciel.**
**Tese (doutorado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.**

**Inclui referências e apêndice.**

**1. Avaliação de desempenho. 2. Redes de Petri – Modelagem de sistemas. 3. Data Center. 4. Avaliação de sustentabilidade. I. Maciel, Paulo Romero Martins (orientador). II. Título.**

**004.029 CDD (23. ed.) MEI2014 – 005**

Tese de Doutorado apresentada por **Gustavo Rau de Almeida Callou** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Assessment to Support the Planning of Sustainable Data Centers with High Availability**" orientada pelo **Prof. Paulo Romero Martins Maciel** e aprovada pela Banca Examinadora formada pelos professores:

_____
Prof. Ricardo Massa Ferreira Lima
Centro de Informática / UFPE

_____
Prof. Nelson Souto Rosa
Centro de Informática / UFPE

_____
Prof. Dietmar Tutsch
Automation/Computer Science – University of Wuppertal

_____
Prof. Fábio Santana Magnani
Departamento de Engenharia Mecânica / UFPE

_____
Prof. Henrique Pacca Loureiro Luna
Instituto de Computação / UFAL

Visto e permitida a impressão.
Recife, 12 de novembro de 2013.

_____
**Profa. Edna Natividade da Silva Barros**
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

*This thesis is dedicated to my parents.*

# ACKNOWLEDGEMENTS

I would like to thank my parents, who have always been supporting and encouraging me to do what I have dreamed. Many thanks to my sisters who helped me keeping going at all difficult moments.

My acknowledgement to my friend and advisor Professor Paulo Maciel who have believed in my intellectual capacity and have given me the chance to be part of his research group, helping me whenever necessary, in order to achieve my goals.

I thank Professor Dietmar Tutsch for his hostage, orientation and friendship, who have become me a member of his research group in Wuppertal-Germany for one year. I also thank Professor Fábio Magnani for his valuable support on the sustainability topics.

Thanks to my friend and Professor Sérgio Galdino who gave me the opportunity to start my academic life as a researcher and who introduced me to Professor Paulo Maciel.

I also thanks my colleagues during my stay in Germany at the University of Wuppertal (BUW).

Thanks to all professors and staff in UFPE Informatics Center (CIn). Special remarks to my colleagues and friends Bruno Silva, Ermeson Andrade, Jair Figueirêdo, João Ferreira, Julian Menezes, and Rafael Souza for the motivating moments and for all support and help.

I would also like to thank the Informatics Group in the UFPE Clinical Hospital, for the support to realize this thesis.

*If we knew what it was we were doing,*
*it would not be called research,*
*would it?*

— ALBERT EINSTEIN

# RESUMO

O surgimento de serviços como computação nas nuvens, redes sociais e comércio eletrônico tem aumentado a demanda por recursos computacionais dos data centers. Preocupações decorrentes para os projetistas de data center são sustentabilidade, custo, e dependabilidade, os quais são significativamente afetados pelas arquiteturas redundantes requeridas para suportar tais serviços. Nesse contexto, modelos são ferramentas importantes para projetistas quanto a tentativa de quantificar esses problemas antes mesmo de implementar a arquitetura final.

Nessa tese, um conjunto de modelos é proposto para a quantificação integrada do impacto na sustentabilidade, custo e dependabilidade das infraestruturas de refrigeramento e potência de data centers. Isso é obtido com o suporte do ambiente de avaliação que é composto pelas ferramentas ASTRO, Mercury e o módulo de otimização. A avaliação de dependabilidade faz uso de uma estratégia de modelagem híbrida que usa as vantagens tanto das redes de Petri estocásticas como dos diagramas de blocos de confiabilidade. Além disso, um modelo é proposto para realizar a verificação se fluxo de energia não excede a capacidade máxima de potência que cada equipamento pode prover (considerando dispositivos elétricos) ou extrair (assumindo equipamentos de refrigeração). Adicionalmente, um método de otimização é proposto para melhorar os resultados obtidos através dos diagramas de blocos de confiabilidade, das redes de Petri estocásticas e do modelo de fluxo de energia pela seleção automática dos dispositivos apropriados a partir da lista de componentes candidatos. Essa lista corresponde a um conjunto de componentes que podem ser utilizados para compor a arquitetura de data center.

Vários estudos de casos são apresentados para analisar o impacto ambiental, a dependabilidade e o custo operacional de energia elétrica de arquiteturas reais de potência e refrigeração de data centers.

**Palavras-chave:** Redes de Petri, Diagrama de Blocos de Confiabilidade, Modelo de Fluxo de Energia, Sustentabilidade, Dependabilidade e Arquiteturas de Data Center.

# ABSTRACT

The advent of services such as cloud computing, social networks and e-commerce has led to an increased demand for computer resources from data centers. Prominent issues for data center designers are sustainability, cost, and dependability, which are significantly affected by the redundant architectures required to support these services. Within this context, models are important tools for designers when attempting to quantify these issues before implementing the final architecture.

This thesis proposes a set of models for the integrated quantification of the sustainability impact, cost, and dependability of data center power and cooling infrastructures. This is achieved with the support of an evaluation environment which is composed of ASTRO, Mercury and Optimization tools. The approach taken to perform the system dependability evaluation employs a hybrid modeling strategy which recognizes the advantages of both stochastic Petri nets and reliability block diagrams. Besides that, a model is proposed to verify that the energy flow does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling equipment). Additionally, an optimization method is proposed for improving the results obtained by Reliability Block Diagrams, Stochastic Petri nets and Energy Flow models through the automatic selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center architecture.

Several case studies are presented that analyze the environmental impact and dependability metrics as well as the operational energy cost of real-world data center power and cooling architectures.

**Keywords:** Petri nets, Reliability Block Diagrams, Energy Flow Model, Sustainability, Dependability and Data Center Architectures.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| **A** | Availability |
| $CO_2$ | Carbon Dioxide |
| **CRAC** | Computer Room Air Conditioning |
| **CTMC** | Continuous Time Markov Chain |
| **EFM** | Energy Flow Model |
| **GRASP** | Greedy Randomized Adaptive Search Procedure |
| **IT** | Information Technology |
| **MC** | Markov Chain |
| **MNRT** | Mean Non-Repair Time |
| **MTTF** | Mean Time to Failure |
| **MTR** | Mean Time to Restore |
| **MTTR** | Mean Time to Repair |
| **NAS** | Network Attached Storage |
| **PDU** | Power Distribution Units |
| **PUE** | Power Usage Effectiveness |
| **PLDA** | Power Load Distribution Algorithm |
| **PN** | Petri net |
| **R** | Reliability |
| **RI** | Reliability Importance |
| **RCI** | Reliability and Cost Importance |
| **RBD** | Reliability Block Diagram |
| **SAN** | Storage Area Network |

**SPN** Stochastic Petri Net

**SDT** Step Down Transformers

**STS** Static Transfer Switch

**UPS** Uninterruptible Power Supplies

**TTF** Time to Failure

**TTR** Time To Repair

**TPN** Timed Petri net

# CONTENTS

CHAPTER 1

# INTRODUCTION

Environmental impacts have received great attention from the scientific community and industry due to diverse concerns: climate change, pollution and environmental degradation. For instance, $CO_2$ emissions could rise between 9% to 27% by 2030 depending on the policies enacted [1]. Additionally, carbon dioxide released by the use of coal, petroleum, natural gas and other sources contributes to the global warming. In U.S., coal burning generates almost half of the electricity consumed, but it emits 78 percent of all $CO_2$ provided from electric power plants [2]. Therefore, many countries have been demanding the substitution of polluting energy sources with non-polluting ones (e.g., sun, wind and hydropower).

In Information Technology (IT) systems, the emergence of paradigms, such as cloud computing, e-commerce and social network services, has demanded data center infrastructures with several thousands of computers. To support those paradigms, a rapid increase in computing and communication capabilities were provided by data centers which has changed the global economy. For instance, more than two billion people now access the Internet, up from less than 250 million a decade ago [3]. However, this remarkable growth comes with an increase on the power consumption that accounts for about 2% of today's U.S. power generation [4]. Therefore, they also significantly contribute to the global carbon emissions which are predicted to increase another 70% by 2020 [3].

To accomplish the high availability levels demanded by those paradigms, data center infrastructures have evolved dramatically dealing with redundant strategies. However, since redundancy leads to additional devices, more power resources are required which may result in a negative impact on sustainability and cost of the system. Therefore, concerns about dependability, sustainability and cost of data center systems are in sharp focus by both the academic community and society.

Although in some organizations the sustainability concern may be a not important factor, in others, the sustainability concern has been in sharp focus. For instance, IT organizations are incorporating sustainable business practices into the design and operation of their products and systems [5]. Therefore, data center designers need to examine several trade-offs concerning in addition to dependability, sustainability impact and cost metrics to determine a data center architecture. However, at present, designers do not have many mechanisms to conduct the integrated evaluation and optimization of dependability, sustainability and cost on data center infrastructures. Indeed, two different data center architectures with similar availability levels and cost may have very different sustainability impacts. Additionally, a growing concern of data center designers is related

to the identification of components that may cause the system failure as well as systems parts that must be improved before implementing the architecture.

A prominent mechanism to compare data center equipment would be to assess their energy consumption, material utilization, environmental impact and irreversible natural losses for the next generations, both in the manufacturing as in the operational phase. A given equipment could be considered the best option even having a greater energy consumption, once it had a lesser negative impact on the future. As an index for this global assessment, the thermodynamic property exergy is of utter importance due to the fact that it estimates the energy conversion efficiency of a system. Exergy is defined as the maximal fraction of the energy that could be theoretically converted in useful work [6]. In this work, we quantify the environmental impact in terms of the carbon dioxide ($CO_2$) emissions [7] as well as the thermodynamic metric of exergy consumption.

In this thesis, we propose a set of formal models to the integrated quantification of sustainability impact, cost and dependability issues on data center power and cooling infrastructures. The adopted approach takes into account a hybrid modeling technique that considers the advantages of both Stochastic Petri Nets (SPNs) [8] and Reliability Block Diagrams (RBDs) [9] to evaluate system dependability. Additionally, we propose the Energy Flow Model (EFM) to verify that the energy flow does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling equipment).

Algorithms that traverse the EFM are proposed to perform the power verifications as well as to estimate data centers cost and sustainability impacts. Additionally, an optimization method is proposed for improving the results obtained by RBD, SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center architecture. The evaluation of all possible combinations from the list of candidate components provides the optimal result. However, this evaluation is quite time consuming. The adopted optimization method is an alternative approach that is proposed in this work that provides results close to the optimal ones in a reduced execution time.

The integrated environment, composed of the high-level interface named ASTRO as well as the kernel named Mercury, has been implemented to support the joint evaluation of sustainability, cost and dependability. ASTRO provides views that allow data center designers to specify power, cooling and IT systems. Mercury is the modeling and evaluation kernel for the fundamental models (EFM, SPN, RBD and Markov Chain (MC)). The models created in the views supported by ASTRO can be automatically converted to the fundamental models supported by Mercury (considering dependability models). This work focuses on the models supported by Mercury.

## 1.1 MOTIVATION

Sustainability has received great attention by the scientific community, due to concerns for meeting current needs of energy without compromising, for instance, non-renewable re-

sources for future generations. In addition, as a result of stringent availability constraints, dependability plays a prominent role in the infrastructure that supports business service through the Internet, particularly, the growth of cloud computing paradigm. Thus, there has been a tremendous growth in the number, size and power densities of data centers.

A data center is not only composed of IT equipment, but also the power and cooling infrastructures, which incur considerable energy consumption. Additionally, considering the increase in energy costs as well as the global attention focused on sustainability, the energy management and environment impacts of data centers are critical. In Western Europe [10], the electricity consumption is estimated to increase from 56 terawatt hours (TWh) in 2007 to 104 TWh by 2020, in which data center power consumption will play a significant part of that increase. Indeed, data centers consume about 2% of the U.S. power generation [4], and, so, they also significantly contribute to the global carbon emissions.

The data center infrastructures responsible for feeding and cooling a data center room consume almost the same amount of energy than servers, storage and networking hardware together [11]. Besides, power and cooling infrastructure costs can match, or even exceed, the cost of the data center's IT devices [12]. Therefore, a detailed study focusing in those two infrastructures are important as well. Additionally, data center energy requirements have grown massively in recent years going from 12GW in 2007 to 24GW in 2011 [13]. In the last year alone, it has increased to 38GW amid data explosion and business expansion. Besides that, the energy consumed by data centers is estimated to have a further rise of 17% going to 43GW in 2013. As IT as a whole is responsible for 2% of the global carbon emissions [3], a small percentage savings in the energy consumption of data centers will have a huge economic and environmental impact.

## 1.2 OBJECTIVES

The world has begun to understand the necessity to reduce the environmental impacts by consuming sustainable energy. In this context, this work proposes a set of models for the integrated quantification of sustainability impact, cost and dependability of data center power and cooling infrastructures. The proposed approach takes into account a hybrid modeling strategy that considers the advantages of both stochastic Petri nets (SPN) [8] and reliability block diagrams (RBD) [9] to evaluate system dependability. Besides, a model is proposed to allow the verification of the energy flowing does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling equipment). Furthermore, the specific goals of this research are:

- To propose a set of formal models for estimating dependability metrics of power and cooling data center infrastructures.
- To construct and evaluate models that represent data center power and cooling infrastructures in order to identify system parts that may be improved.

- To propose indexes that identify the components on data center architectures that most impact the system reliability. Therefore, data center architecture can be proposed based on the components that are indicated to be replicated through those indexes.
- To propose a model that represents electrical energy flow.
- To propose algorithms that traverse the energy flow model to verify the power restrictions, to estimate data center cost and sustainability impacts.
- To propose a methodology that allows data center designers to analyze the system piecewise and to combine the evaluation results.
- To develop a tool which supports the above models and allows data center designers/administrators to conduct the joint evaluation of sustainability, cost and dependability.
- To define a model that optimize the integrated sustainability, dependability and cost evaluation of data center architectures.

In addition, it is important to state that the main value of the adopted methodology lies in being able to provide assessment to support the planning of data center power and cooling infrastructures taking into account sustainability impact, dependability metrics and cost issues.

## 1.3 CONTRIBUTION

The expected contributions are a set of formal models for quantifying sustainability, dependability and cost values on data center architectures as well as methods for performing the verification of power restrictions. The computed metrics are dependability (e.g., availability, reliability and reliability importance), sustainability impact (exergy consumption and $CO_2$ emissions) and cost issues (acquisition and operational costs) on data center power and cooling infrastructures. Additionally, an optimization method is also proposed for optimizing the results achieved through the above models. To accomplish the integrated approach to evaluate and optimize dependability, cost and sustainability issues, a methodology is proposed for evaluating data center systems through a hybrid formal modeling, which utilizes RBD, SPN and EFM whenever they are best suited.

Regarding the expected results of this research, we assume that the achieved goals will have valuable importance for the academic community as well as for the IT companies. In the academic community, we have been publishing the results in international conferences and journals. Moreover, dissertations and thesis have been extending this research due to the importance of the topic for IT companies and society as a whole.

In IT business, we expect that the proposed models as well as the tool may be applied by companies to reduce the environmental impacts from data centers and also to improve

the companies' income and, as a consequence, to also increase the number and income of employees.

## 1.4 OUTLINE

This work is organized as follows:

In Chapter 2, we introduce basic concepts on data center infrastructures, dependability, sustainability, Petri nets, reliability block diagrams and optimization techniques. Afterwards, we present the proposed models for sustainability, dependability, cost as well as energy flow evaluation in Chapter 3. Next, Chapter 4 presents the conceived methodology as well as the evaluation environment that is composed of ASTRO, Mercury and the optimization module. Chapter 5 shows the experiments conducted using the proposed models as well as illustrates the applicability of the adopted methodology. Chapter 6 presents related work. Finally, Chapter 7 presents conclusions as well as direction for future work.

CHAPTER 2

# BACKGROUND

This chapter introduces a summary of the background information needed for a better understanding about this work. First of all, an overview of data center infrastructures are presented. After that, dependability, sustainability and total cost of ownership concepts are shown. Next, Petri nets, reliability block diagrams and optimization techniques are explained.

## 2.1 DATA CENTER INFRASTRUCTURE

Data centers can be defined as a computer facility designed for safeguarding company's data properties [14]. Typically, data centers are adopted as host for website, to process business transactions, to protect data (e.g., financial records, medical history of patients, manage emails). In addition, organizations are looking for IT departments aiming to receive support in many business sources such as productivity and revenue. Therefore, data centers must have high availability levels to support new technology requirements.

To build a data center room, many infrastructures must be analyzed. For instance, power, cooling, connectivity, space, protections (e.g., against fire) and temperature monitoring. Additionally, data center designers should focus on increase productivity and avoid downtime. Design strategies are important to accomplish those needs as shown bellow.

Data center has to be reliable. In order to accomplish high reliability values, data center infrastructures must be composed of: (i) multiply standby power supplies, which are responsible for support the data center's electrical load when the commercial utility power fails; (ii) redundant network to keep the system working if a networking device has failed; (iii) redundant data connections to provide alternative paths to the desire device.

Availability (or data center uptime) is a critical metric due to the fact that downtime usually affect company's production as well as business profit. Table 2.1 presents different availability levels in nines ($-log(1 - Availability(\%)/100)$) and its correspondent downtime considering one year period.

**Table 2.1:** Correspondent downtime to different availabilities in one year.

| Availabiliy (9s) | Percent | Downtime |
|---|---|---|
| Two nines | 99 | 3 days, 15 hours, 40 minutes |
| Three nines | 99.9 | 8 hours, 46 minutes |
| Four nines | 99.99 | 52 minutes, 35 seconds |
| Five nines | 99.999 | 5 minutes, 15 seconds |
| Six nines | 99.9999 | 32 seconds |

In addition to the building facility, a generic data center system (Figure 2.1) essentially consists of the following subsystems: (i) IT infrastructure; (ii) cooling infrastructure; and (iii) power infrastructure [15].

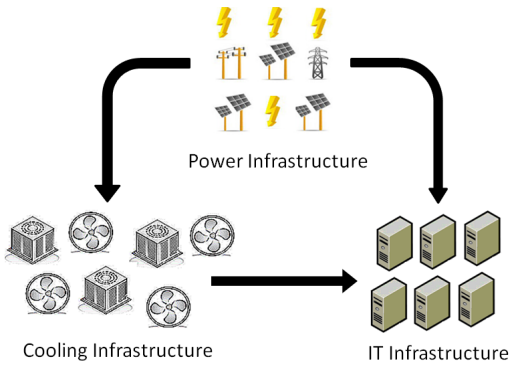


**Figure 2.1:** Data Center Architecture

**IT Infrastructure** consists of three main components - servers, networking and storage devices [15]. The storage devices are typically connected through a Storage Area Network (SAN). Servers may also connect to remote file systems on Network Attached Storage (NAS) over Ethernet. The network devices include the physical hardware adopted to transmit data electronically such as switches, hubs, routers, bridges and gateways. Software services are usually organized in a multi-tier architecture with separate tiers for web servers, applications and database servers [16].

**Cooling Infrastructure** is basically comprised of Computer Room Air Conditioning (CRAC) units, chillers and cooling towers [15]. The cooling infrastructure may account around 40% of the total power consumption of the data center [17].

Figure 2.2 illustrates the cooling infrastructure. Heat dissipated from IT devices is extracted by CRAC units and transferred to the chilled water distribution system. As the heat is transferred from the air stream to the chilled water stream, the temperature of the water increases. Heat is removed from the water via thermodynamic work in order to improve the chiller refrigeration cycle. The refrigerated water is returned to the CRAC unit, while the heat absorbed by the refrigerant is rejected to a secondary water stream. The secondary loop ultimately transfers the heat to the outside environment in a cooling tower.

**Power Infrastructure** is responsible for providing uninterrupted, conditioned power at the correct voltage and frequency to the IT equipment hosted in data center racks as well as to the cooling infrastructure [18]. From the electric utility, the power typically, goes through Step Down Transformers (SDT), transfer switches, Uninterruptible Power Supplies (UPS), Power Distribution Units (PDU), and finally to rack power strips. Fault-tolerance is directly related to either resource or timing replication, and hence is related to higher power consumption. The UPS conditions power and provides power backup in

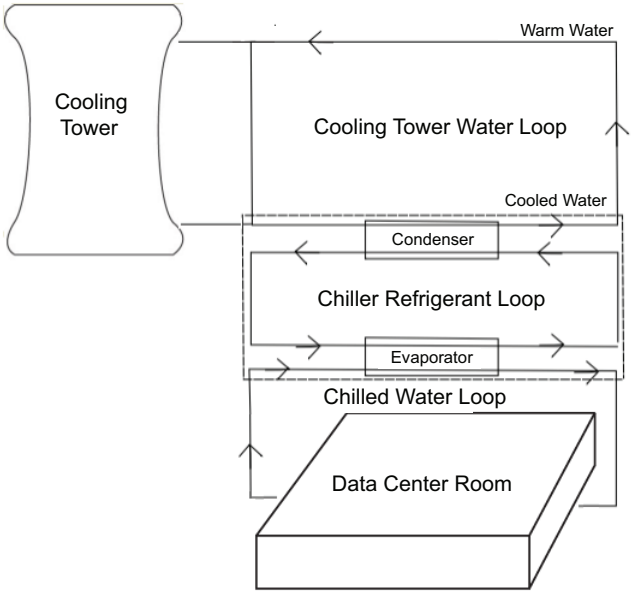


**Figure 2.2:** Cooling Infrastructure.

case of short outages. Generators or other local power sources may be used for longer outages, or even for satisfying critical parts of the total power demand on a regular basis.

Redundancy is an essential issue in power infrastructures, since its availability affects the overall system services. As depicted in Figure 2.1, the power infrastructure affects every other subsystems, and a system outages can be very expensive, easily surpassing millions of dollars per hour [19].

Power infrastructures are comprised of primary/secondary powers, UPS, Static Transfer Switch (STS), PDUs (SDT and electrical circuit breaker panels), junction box and rack power strip. The following lines detail the functionality of the main components of a typical power system.

The data center primary power corresponds to the electrical power that can be derived from different electrical substations. An electrical substation is a subsidiary station of an electricity generation, transmission and distribution system where voltage is transformed from high to low or the reverse using transformers. Electric power may flow through several substations between generating plant and consumer, and may be changed in voltage in several steps.

Considering the fact that the primary power may fail, a generator (secondary power) must be able to provide the input power to the data center. This component helps ensure continuous power availability to the racks and supporting cooling infrastructure by compensating for fluctuations from the electrical power or temporary power loss. Different techniques can be used for power generation in data centers, which can eliminate dependence on electrical power or be used for backup power in situations where the electrical power is expensive or unreliable. Power production is most commonly achieved

via diesel generators, although some smaller facilities have also utilized fuel cells [14].

UPS units are used to improve power source quality as well as to protect the electrical loads against disturbances such as frequency shifts, voltage spikes or interruptions. A PDU is a device that distributes electrical power. For instance, rack PDUs (or power-strips) are connected to breakers present on data center's PDU. Static transfer switches (STS) are responsible for switching from a power utility that has failed to backup batteries or to other power sources at any stage of the delivery process. Step down transformers (SDT) are used to reduce alternating current. In addition, SDT allows a device that requires a low voltage power supply to operate from a higher voltage.

It is important to highlight that the cooling infrastructure is more tolerant to unconditioned power than the IT devices [20]. Therefore, UPS devices are usually adopted on the IT data center system. Although uninterrupted cooling is required to the IT devices, the CRAC can still be treated as a non-critical load in order to reduce the power delivery costs.

## 2.2 DEPENDABILITY

The dependability [21, 22] of a system can be understood as the ability to deliver a set of services that can be justifiably trusted. Indeed, dependability is related to issues such as fault tolerance and reliability. Reliability is the probability that the system will deliver a set of services for a given period of time, whereas a system is fault tolerant when it does not fail even when there are faulty components [23]. Availability is also another important concept, which quantifies the mixed effect of both failure and repair process in a system [9].

For any given time period represented by the interval (0, $t$), $R(t)$ is the probability that the component has continued to function (not failed) from 0 until $t$. When an exponentially distributed Time to Failure (TTF) is considered, reliability is represented by:

$$R(t) = exp\left[-\int_0^t \lambda(t')dt'\right] \tag{2.1}$$

where $\lambda(t')$ is the instantaneous failure rate.

The simple definition of availability ($A$) can be outlined as the ratio of the expected system uptime by the expected system up and down times:

$$A = \frac{E\left[Uptime\right]}{E\left[Uptime\right] + E\left[Downtime\right]} \tag{2.2}$$

Consider that the system started operating at time $t$=$t$' and fails at $t$=$t$", thus $\Delta t =$

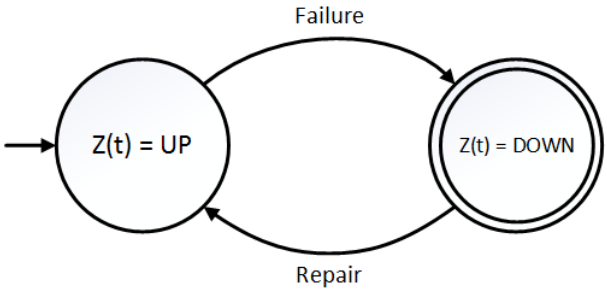


**Figure 2.3:** States of a repairable system.

$t'' - t' =$ Uptime (see Figure 2.3). The system availability can be thus expressed by:

$$A = \frac{MTTF}{MTTF + MTR} \tag{2.3}$$

where MTTF is the Mean Time to Failure, and MTR is the mean time to restore, defined by MTR = MNRT + MTTR (MNRT - mean non-repair time, MTTR - mean time to repair), so:

$$A = \frac{MTTF}{MTTF + MNRT + MTTR} \tag{2.4}$$

If $MNRT \cong 0$,

$$A = \frac{MTTF}{MTTF + MTTR} \tag{2.5}$$

The instantaneous availability is the probability that the system is operational at a specific time instant t, that is,

$$A(t) = P\left\{Z\left(t\right) = 1\right\} = E\left\{Z\left(t\right)\right\}, t \geq 0. \tag{2.6}$$

If system repairing is not possible, the instantaneous availability, $A(t)$, is equivalent to reliability, $R(t)$. If the system approaches stationary states as the time increases, it is possible to quantify the steady state availability, such that it is possible to estimate the long-term fraction of time the system is available.

$$A = \lim_{t \to \infty} A\left(t\right), t \geq 0. \tag{2.7}$$

Through transient analysis or simulation, the reliability ($R$) is obtained, and, then, the $MTTF$ can be calculated as well as the standard deviation of the Time To Failure

$(TTF)$:

$$MTTF = \int_0^\infty t f(t) dt = \int_0^\infty -\frac{dR(t)}{dt} t dt = \int_0^\infty R(t) dt \tag{2.8}$$

$$sd(TTF) = \sqrt{\int_0^\infty t^2 f(t) dt - (MTTF)^2} \tag{2.9}$$

One should bear in mind that, for computing reliability of a given system service, the repairing activity of the respective service must not be represented. Besides, taking into account $UA = 1 - A$ (unavailability) and Equation 2.5, the following equation is derived

$$MTTR = MTTF \times \frac{UA}{A} \tag{2.10}$$

The standard deviation of the Time To Repair (TTR) can be calculated as follows:

$$sd(TTR) = sd(TTF) \times \frac{UA}{A} \tag{2.11}$$

Next, $\frac{MTTF}{sd(TTF)}$ (and $\frac{MTTR}{sd(TTR)}$) are computed for choosing the expolynomial distribution that best fits the $TTF$ and $TTR$ distributions [24, 8].

In many situations, modeling is the method of choice either because the system might not yet exist or due to the inherent complexity for creating specific scenarios under which the system should be evaluated. In a very broad sense, models for dependability evaluation can be classified as simulation and mathematical models. However, this does not mean that mathematical models cannot be simulated. Indeed, many mathematical models, besides being analytically tractable, may also be evaluated by simulation. Mathematical models can be characterized as being either state-based or non-state-based.

Dependability metrics (e.g., availability and reliability) may be calculated either by using RBD or SPN (to mention only the models adopted in this work). RBDs allow to represent component networks and provide closed form equations. Nevertheless, such models face drawbacks for thoroughly handling failures and repairing dependencies that are often faced when representing maintenance policies and redundant mechanism, particularly those based on dynamic redundancy methods. On the other hand, state-based methods can easily consider those dependencies, so allowing the representation of complex redundant mechanisms as well as sophisticated maintenance policies. However, they suffer from the state-space explosion. Some of those formalism allow both numerical analysis and stochastic simulation, and SPN is one of the most prominent models of such class.

### 2.2.1 Time Distributions

The time to failure of a component is a non-negative continuous random variable. In this section we briefly present Exponential and Expolynomial continuous distributions (e.g. Erlang and Hyper-exponential) that are widely adopted in dependability evaluation.

**Exponential Distribution**
A random variable X representing a component's life time has exponential distribution if its probability density function (pdf) is given by:

$$f(t) = \lambda e^{-\lambda t}, t \geq 0, \tag{2.12}$$

where $\lambda > 0$ is a parameter of this distribution. The coefficient of variation (CV) is one, and the respective reliability function, cumulative distribution function, failure rate, mean (MTTF) and variance are, respectively:

$$R(t) = e^{-\lambda t}, t \geq 0, \tag{2.13}$$

$$F(t) = 1 - e^{-\lambda t}, t \geq 0, \tag{2.14}$$

$$h(t) = \lambda, \tag{2.15}$$

$$E(X) = MTTF = \frac{1}{\lambda}, \tag{2.16}$$

$$Var(X) = \sigma^2 = \frac{1}{\lambda^2}. \tag{2.17}$$

Exponential distribution is important due to the fact that it is the only continuous distribution that has the memoryless property. This property is related to the fact that future steps depend only on relevant information about the present, information about the past is completely irrelevant.

Exponential distribution is related to discrete Poisson random variable. Therefore, it has the following properties:

- If $n$ Poisson processes with distributions for the interarrival times $1 - e^{-\lambda_i t}$, $1 \leq i \leq n$, are merged into one single process, then it results in a Poisson process that has the distribution $1 - e^{-\lambda_i t}$ with $\lambda = \sum_{i=1}^{n} \lambda_i$ as shown in Figure 2.4(a).
- If a Poisson process (distribution $1 - e^{-\lambda t}$) is split into $n$ processes, then $ith$ subprocess are created (distribution $1 - e^{-q_i \lambda t}$) as depicted in Figure 2.4(b).

The exponential distribution is not a good approximation for experiments in which the coefficient of variation is different of one. Therefore, other distributions are considered to represent those systems.

**Hyperexponential Distribution**
The hyperexponential distribution is adopted to approximate empirical distributions that have a coefficient of variation larger than one. Figure 2.5 depicts an example of a model

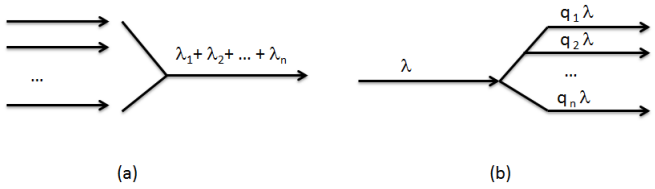


**Figure 2.4:** (a) Merging, (b) Splitting.

with hyperexponential distribution time. The model is composed of $k$ phases and $\lambda_1$, $\lambda_2$, ..., $\lambda_k$ rates that are arranged in parallel. The probability of each $i-th$ phase together is $q_i$, where $\sum_{i=1}^{k} q_i = 1$. Additionally, it is important to stress that only one phase can be occupied at a time.

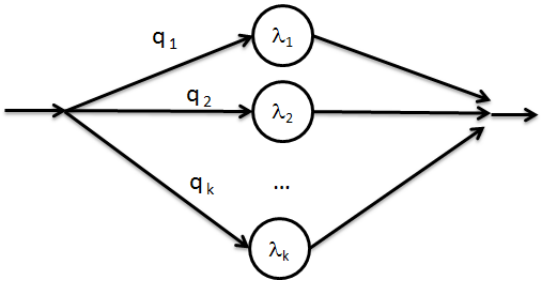


**Figure 2.5:** A random variable with hyperexponential distribution.

### Erlang Distribution

In this distribution the coefficient of variation is less than one. Figure 2.6 shows a model that has an Erlang distribution in which $k$ is the number of identical exponential phases in series. Therefore, if the interarrival times of processes in a system are identical exponentially distributed, it follows an Erlang distribution.

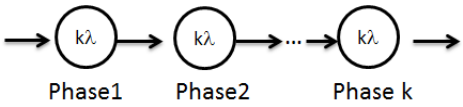


**Figure 2.6:** A random variable with erlang distribution.

Table 2.2 presents a summary of continuous distributions. In this table, the probability density function, the respective reliability function, cumulative distribution function, failure rate, mean (MTTF) and variance are represented. For more details, the reader should refer to [25].

**Table 2.2:** Distribution Summary.

| Distributions | f(t) | R(t) | F(t) | h(t) | E[X] | Var[X] | Parameter |
|---|---|---|---|---|---|---|---|
| Exponential | $\lambda e^{-\lambda t}$, $t \geq 0$ | $e^{-\lambda t}$, $t \geq 0$ | $1 - e^{-\lambda t}$, $t \geq 0$ | $\lambda$ | $\frac{1}{\lambda}$ | $\frac{1}{\lambda^2}$ | $\lambda > 0$, $\lambda : rate$ |
| Erlang | $\frac{k\lambda(k\lambda t)^{k-1}}{(k-1)!}e^{-k\lambda t}$, $t \geq 0$ | $e^{-k\lambda t} \times \sum_{i=0}^{k-1} \frac{(k\lambda t)^i}{i!}$, $t \geq 0$ | $1 - e^{-k\lambda t} \times \sum_{i=0}^{k-1} \frac{(k\lambda t)^i}{i!}$, $t \geq 0$ | $\lambda$ | $\frac{1}{\lambda}$ | $\frac{1}{k\lambda^2}$ | $k \in \mathbb{Z}^+, \lambda > 0$ $rates : k\lambda$ k: num of phases |
| Hyper-exponential | $\sum_{i=1}^{k} q_i \lambda_i e^{-\lambda_i t}$, $t \geq 0$ | $1 - \sum_{i=1}^{k} q_i (1 - e^{-\lambda_i t})$, $t \geq 0$ | $\sum_{i=1}^{k} q_i (1 - e^{-\lambda_i t})$, $t \geq 0$ | $\frac{1}{\sum_{i=1}^{k} \frac{q_i}{\lambda_i}}$ | $\sum_{i=1}^{k} \frac{q_i}{\lambda_i} = \frac{1}{\lambda}$ | $2\times \sum_{i=1}^{k} \frac{q_i}{\lambda_i^2} - \frac{1}{\lambda^2}$ | $k \in \mathbb{Z}^+$(n of phases) rates: $\lambda_1 ... \lambda_k$ probabilities: $q_1 ... q_k$ $\lambda_i > 0, q_i > 0$; $\sum_{i=1}^{k} q_i = 1$ |

### 2.2.2 Redundancy

Common used redundancy structures in reliability systems are the series and parallel arrangements. In a series system, each of the components is essential for the function of the whole system. The reliability of a series system is affected by the number of components in the system, in which more components reduce the reliability of the system. On the other hand, a parallel system requires at least one component to work. In parallel systems, the reliability of the system is higher than the reliability of the best component. Including components in a parallel arrangement increases the reliability of the whole system.

Figure 2.7 depicts a comparison between series and parallel organizations as a function of the number of components in the system. The component reliability $p$ adopted for all the components of the series system was 0.8, and for the parallel system the adopted $p$ value was 0.4. It is possible to notice that the reliability of the parallel system increases when more components are considered. On the other side, in a series arrangement, the reliability reduces once more device are adopted.

### 2.2.3 Measure of Component Importance

Component importance is a metric that indicates the impact of a particular component in the system's overall reliability. In this work, we adopt Reliability Importance (RI) and Reliability and Cost Importance (RCI) to identify the most critical components for system reliability.

**Reliability Importance** is a metric that may be useful for identifying the most critical components. RI (or Birnbaum Importance) of a component $i$ corresponds to the amount of improvement in system reliability, when the reliability of component $i$ is increased by one unit [9]. The $RI$ of component $i$ can be computed as:

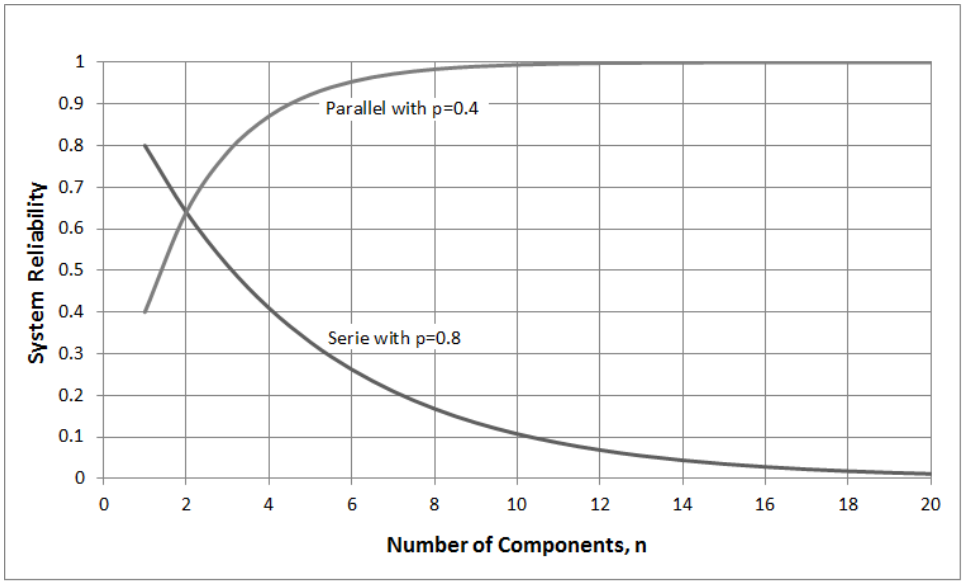


**Figure 2.7:** Comparing the reliability of series and parallel systems.

$$RI_i = R_s(1_i, \mathbf{p}^i) - R_s(0_i, \mathbf{p}^i), \tag{2.18}$$

where, $RI_i$ is the reliability importance of component $i$, $R_s$ is the reliability of the system, $\mathbf{p}^i$ represents the component reliability vector with the $i$th component removed, $0_i$ represents the component $i$ failure, and $1_i$ denotes the component $i$ properly working.

As an example, consider the RBD in Figure 2.8 and the reliability values provided in Table 2.3. The Reliability Importance of *Component 1* is calculated as follows:

$$RI_1^B = 1_1 \times p_2 \times p_3 - 0_1 \times p_2 \times p_3, \tag{2.19}$$

in which $p_i$ is the reliability value of component $i$.

In a series structure, the least reliable component has the highest $RI$, as can be seen in Table 2.3. In a parallel structure, the most reliable component has the highest $RI$ [9].

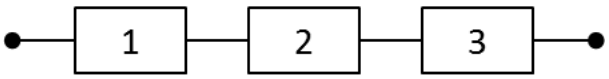


**Figure 2.8:** Reliability Block Diagram Series with 3 Component

**Reliability and Cost Importance** (RCI) is a metric that relates the previous RI considering the acquisition cost of components. The $RCI$ of component $i$ can be computed as:

**Table 2.3:** Reliability Importance (Figure 2.8)

| | Component 1 | Component 2 | Component 3 |
|---|---|---|---|
| Reliability Value | 0.80 | 0.60 | 0.65 |
| Reliability Importance | 0.39 | **0.52** | 0.48 |

$$RCI_i = RI_i \times (1 - \frac{C_i}{C_{sys}}), \tag{2.20}$$

in which $RCI_i$ is the proposed reliability and cost importance index; $RI_i$ is the reliability importance of component $i$; $C_i$ is the acquisition cost of the component $i$ and $C_{sys}$ is acquisition cost of the whole system.

## 2.3 SUSTAINABILITY

This work quantifies the environmental impact in terms of the carbon dioxide ($CO_2$) emissions [7] as well as the thermodynamic metric of exergy. Exergy is defined as the maximal fraction of the energy that could be theoretically converted in useful work [6]. The exergy contained in one $kJ$ of oil is greater than the exergy contained in one $kJ$ of water in the environment temperature. Oil can be used to move an automobile but water in the environment temperature cannot. As given in equation 2.21, exergy is calculated as the product of energy and a quality factor.

$$Exergy = Energy \times F, \tag{2.21}$$

where $F$ is a quality factor represented by the ratio of $Exergy/Energy$ as show in Table 2.4 [6]. For example, $F$ is 0.16 for water at 80 degrees Celsius.

**Table 2.4:** Energy and exergy values of different energy forms.

| Source | $Energy(J)$ | Exergy (J) | F |
|---|---|---|---|
| Water at 80 degrees Celsius | 100 | 16 | 0.16 |
| Steam at 120 degrees Celsius | 100 | 24 | 0.24 |
| Natural gas | 100 | 99 | 0.99 |
| Electricity | 100 | 100 | 1.00 |

Although exergy might be considered a better index for quantifying sustainability than simply energy, it must be used cautiously [6]. For example, although the exergy

of solar radiation is high, the environmental impact in terms of carbon emissions is low. Therefore its destruction or consumption is not as harmful as coal, which has high carbon emission.

Notwithstanding these imperfections, exergy destruction is an important index to express in a simple way both the energetic consumption and the environmental impact of one equipment. Several works [26, 27, 28] present results and valuable conclusions about the use of this index to compare different technological designs.

The exergy destruction metric is useful for quantifying the efficiency of a component or architecture in relation to the energy consumption. Such a metric may be not enough to estimate the environmental impact of identical data center architectures located in different places around the world. This comparison may be assessed by estimating the greenhouse gas emissions such as $CO_2$, which represented 84% of the U.S. greenhouse gas emitted in 2011 [29]. Due to the fact that different energetic mixes are adopted by different countries, quantifying the energy consumption of different energetic mixes may provide estimates about the corresponding environmental impact. Therefore, estimating $CO_2$ emissions related to determinate energetic mix may be considered as a metric for the quantification of environmental impact.

## 2.4 PETRI NETS

Petri nets (PNs) were introduced in 1962 by the PhD dissertation of Carl Adams Petri [30], at Technical University of Darmstandt, Germany. The original theory was developed as an approach to model and analyze communication systems. Petri nets [31] are a graphical and mathematical modeling tool that can be applied in several types of systems and allow the modeling of parallel, concurrent, asynchronous and non-deterministic systems. Since its seminal work, many representations and extensions have been proposed for allowing more concise descriptions and for representing systems feature not observed on the early models. Thus, the simple Petri net has subsequently been adapted and extended in several directions, in which timed, stochastic, high-level, object-oriented and coloured nets are a few examples of the proposed extensions.

### 2.4.1 Place-Transition nets

Place/Transition Petri nets are one of the most prominent and best studied class of Petri nets, and it is sometimes called just by Petri net (PN). A marked Place/Transition Petri net is a bipartite directed graph, usually defined as follows:

**Definition 2.4.1. (A Petri net)** [31] is a 5-tuple:

$$PN = (P, T, F, W, M_0)$$

where:

1. $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places;
2. $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions;
3. $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
4. $W : F \rightarrow \{1, 2, 3, ...\}$ is a weight function;
5. $M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking;

This class of Petri net has two kinds of nodes, called places ($P$) represented by circles and transitions ($T$) represented by bars, such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$. Figure 2.9 depicts the basic elements of a simple PN. The set of arcs $F$ is used to denote the places connected to a transition (and vice-versa). $W$ is a weight function for the set of arcs. In this case, each arc is said to have multiplicity $k$, where $k$ represents the respective weight of the arc. Figure 2.10 shows multiple arcs connecting places and transitions in a compact way by a single arc labeling it with its weight or multiplicity $k$.

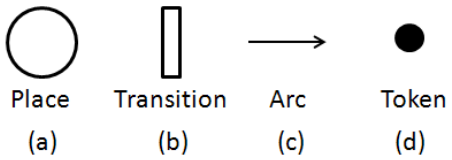


**Figure 2.9:** Petri net basic elements.

Places and transitions may have several interpretations. Using the concept of conditions and events, places represent conditions, and transitions represent events, such that, an event may have several pre-conditions and post-conditions. For more interpretations, Table 2.5 shows other meanings for places and transitions [31].

In the following definitions, we present a different representation for the PN's elements.

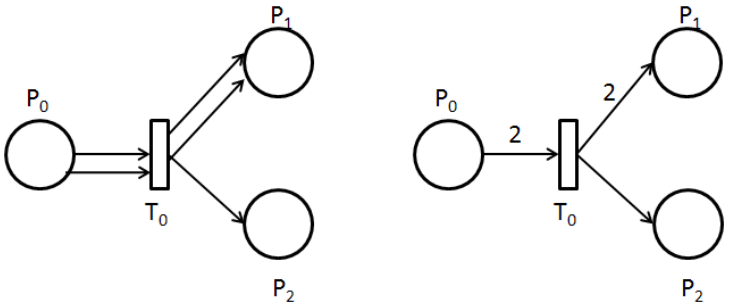


**Figure 2.10:** Compact representation of a PN

| Input Places | Transitions | Output Places |
|---|---|---|
| pre-conditions | events | post-conditions |
| input data | computation step | output data |
| input signals | signal processor | output signals |
| resource needed | tasks | resource releasing |
| conditions | logical clauses | conclusions |
| buffers | processor | buffers |

**Table 2.5:** Interpretation for places and transitions.

**Definition 2.4.2. (Input and Output Transitions of a place)** The set of input transitions (also called pre-set) of a place $p_i \in P$ is:

$$\text{label} = \bullet p_i = \{t_j \in T | (t_j, p_i) \in F\}.$$

and the set of output transitions (also called post-set) is:

$$\text{label} = p_i \bullet = \{t_j \in T | (p_i, t_j) \in F\}.$$

**Definition 2.4.3. (Input and output places of a transition)** The set of input places of a transition $t_j \in T$ is:

$$\text{label} = {}^{\bullet}t_j = \{p_i \in P | (p_i, t_j) \in F\}.$$

and the set of output places of a transition $t_j \in T$ is:

$$\text{label} = t_j{}^{\bullet} = \{p_i \in P | (t_j, p_i) \in F\}.$$

**Marked Petri nets**

A marking (also named token) has a primitive concept in PNs such as place and transitions. Markings are information attributed to places; the number and mark distributions consist of the net state in determined moment. The formal definitions are presented as follows.

**Definition 2.4.4. (Marking)** Considering the set of places $P$ in a net $N$, the marking definition is a function that maps the set of places P into non negative integers $M : P \rightarrow \mathbb{N}$.

**Definition 2.4.5. (Marking vector)** Considering the set of places $P$ in a net $N$, the marking can be defined as a vector $M = (M(p_1), ..., M(p_n))$, where $n = \#(P)$, $\forall p_i \in P$ / $M(p_i) \in \mathbb{N}$. Thus, such vector gives the number of tokens in each place for the marking $M_i$.

**Definition 2.4.6. (A marked Petri net)** is defined by a tuple $NM = (N; M_0)$, where $N$ is the net structure and $M_0$ is the initial marking.

A marked Petri net contains tokens, which reside in places, travel along arcs, and their flow through the net is regulated by transitions. A peculiar distribution *(M)* of the tokens in the places, represents a specific state of the system. These tokens are denoted by black dots inside the places as shown in Figure 2.9 (d).

**Transition enabling and firing**

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state (or marking) in a Petri net is changed according to the following firing rule:

1. A transition $t$ is said to be enabled, if each input place $p$ of $t$ is marked with at least the number of tokens equal to the multiplicity of its arc connecting $p$ with $t$. Adopting a mathematical notation, an enabled transition $t$ for given marking $m_i$ is denoted by $m_i[t >$, if $m_i(p_j) \geq W(p_j, t), \forall p_j \in P$.

2. An enabled transition may or may not fire (depending on whether or not the respective event takes place).

3. The firing of an enabled transition $t$ removes tokens (equal to the multiplicity of the input arc) from each input place $p$, and adds tokens (equal to the multiplicity of the output arc) to each output place $p'$. Using a mathematical notation, the firing of a transition is represented by the equation $m_j(p) = m_i(p) - W(p,t) + W(t,p), \forall p \in P$. If a marking $m_j$ is reachable from $m_i$ by firing a transition $t$, it is denoted by $m_i[t > m_j$ .

Figure 2.11 (a) shows the mathematical representation of a Petri net model with three places ($p_0$, $p_1$, $p_2$) and one transition ($t_0$). Additionally, there is one arc connecting the place $p_0$ to the transition $t_0$ with weight two, one arc from the place $p_1$ to the transition $t_0$ with weight one, and one arc connecting the transition $t_0$ to the place $p_2$ with weight two. The initial marking ($m_0$) is represented by three tokens in the place $p_0$ and one token in the place $p_1$. Figure 2.11 (b) outlines its respective graphical representation, and Figure 2.11 (c) provides the same graphical representation after the firing of $t_0$. For this example, the set of reachable markings is m = $\{m_0 = (3,1,0), m_1 = (1,0,2)\}$. The marking $m_1$ was obtained by firing $t_0$, such that, $m_1(p_0)$ = 3 - 2 + 0, $m_1(p_1)$ = 1 - 1 + 0, and $m_1(p_2)$ = 0 - 0 + 2.

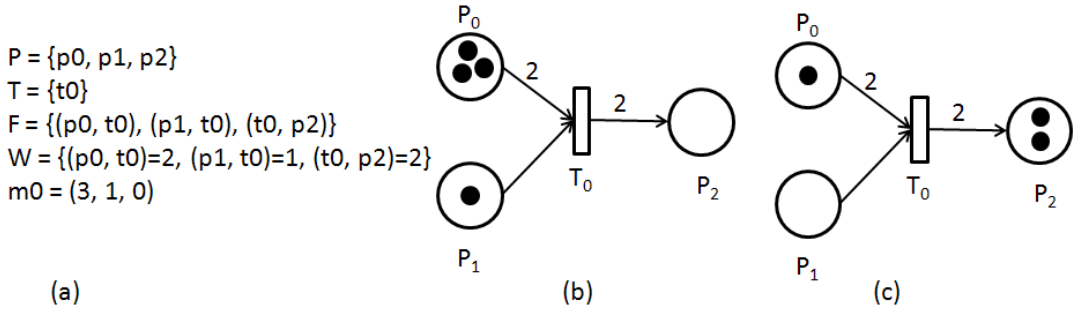


**Figure 2.11:** (a) Mathematical formalism; (b) Graphical representation before the firing of $t_0$; (c) Graphical representation after the firing of $t_0$.

There are two particular cases which the firing rule happens differently. The first one is a transition without any input place that is called as a *source* transition, and the other one is a transition without any output place, named *sink* transition. A *source* transition is unconditionally enabled, and the firing of a *sink* transition consumes tokens, but does not produce any. Figure 2.12 (a) shows a *source* transition, and Figure (b) 2.12 depicts

a *sink* transition. In both, the markings are represented before and after their respective firing.

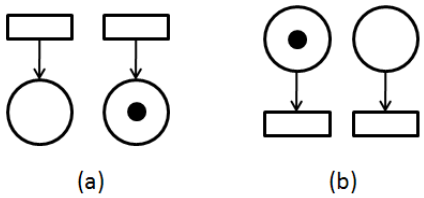


**Figure 2.12:** (a) *Source* transitions; (b) *Sink* transitions.

**Definition 2.4.7. (Source transitions)** A transition is said to be source if, and only if, $I(p,t) = 0$, $\forall p \in P$.

**Definition 2.4.8. (Sink transitions)** A transition is said to be sink if, and only if, $O(p,t) = 0$, $\forall p \in P$.

**Definition 2.4.9. (Inhibitor arc)** Originally not present in PN, the introduction of the concept of inhibitor arc increases the modeling power of PN, adding the ability of testing if a place does not have tokens. In the presence of an inhibitor arc, a transition is enabled to fire if each input place connected by a normal arc has a number of tokens equal to the arc weight, and if each input place connected by an inhibitor arc has no tokens. Figure 2.13 illustrates an inhibitor arc connecting the input place $p_0$ to the transition $t_0$, which is denoted by an arc finished with a small circle. In such Figure, the transition $t_0$ is enabled to fire.

**Definition 2.4.10. (Pure net)** A Petri net is said to be pure if it has no self-loops. A pair of a place $p$ and transition $t$ is called a self-loop if $p$ is both an input and output place of $t$. Figure 2.14 shows a self-loop net.

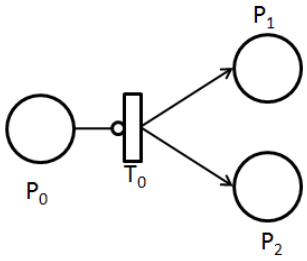


**Figure 2.13:** PN with an inhibitor arc.

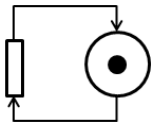

**Figure 2.14:** Self-Loop.

### 2.4.2 Elementary structures

Elementary nets are used as building blocks in the specification of more complex applications. Figure 2.15 shows five structures, namely, (a) sequence, (b) fork, (c) synchronization, (d) choice, and (e) merging.

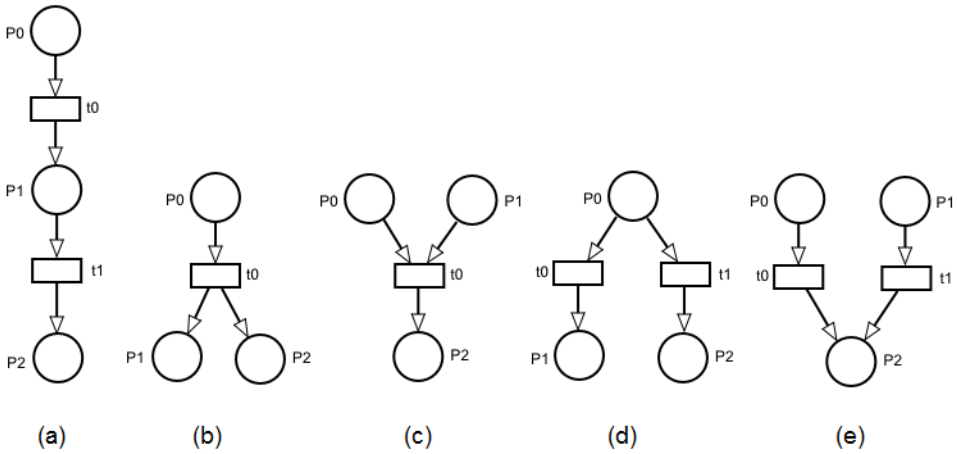


**Figure 2.15:** Elementary PN Structures.

#### Sequence

Sequence structure represents sequential execution of actions, provided that a condition is satisfied. After the firing of a transition, another transition is enabled to fire.

Figure 2.15(a) depicts an example of this structure in which a mark in place $p_0$ enables the transition $t_0$. The firing of transition $t_0$ enables the transition $t_1$ ($p_1$ is marked).

**Fork**

Figure 2.15(b) shows an example of a fork structure that allows the creation of parallel processes.

**Join**

Generally, concurrent activities need to synchronize with each other. This net (Figure 2.15(c)) combines two or more nets, allowing that another process continues this execution only after the end of predecessor processes.

**Choice**

Figure 2.15(d) depicts a choice model, in which the firing of the transition $t0$ disables the transition $t_1$. This building block is suited for modeling if-then-else statement, for instance.

**Merging**

The merging is an elementary net that allows the enabling of the same transition by two or more processes. Figure 2.15(e) shows a net with two independent transitions ($t_0$ and $t_1$) that have an output place in common ($P2$). Therefore, firing of any of these two transitions, a condition is created ($p_2$ is marked) which allows the firing of another transition (not shown in the figure).

**Confusions**

The mixing between conflict and concurrency is called confusion. While conflict is a local phenomenon in the sense that only the pre-sets of the transitions with common input places are involved, confusion involves firing sequences. Figure 2.16 depicts two types of confusions: (a) symmetric confusion, where two transitions $t_1$ and $t_3$ are concurrent while each one is in conflict with transition $t_2$; and (b) asymmetric confusion, where $t_1$ is concurrent with $t_2$, but will be in conflict with $t_3$ if $t_2$ fires first.

### 2.4.3 Petri nets modeling example

In this section, a simple example is given in order to introduce how to model some basic concepts such as parallel process in Petri nets.

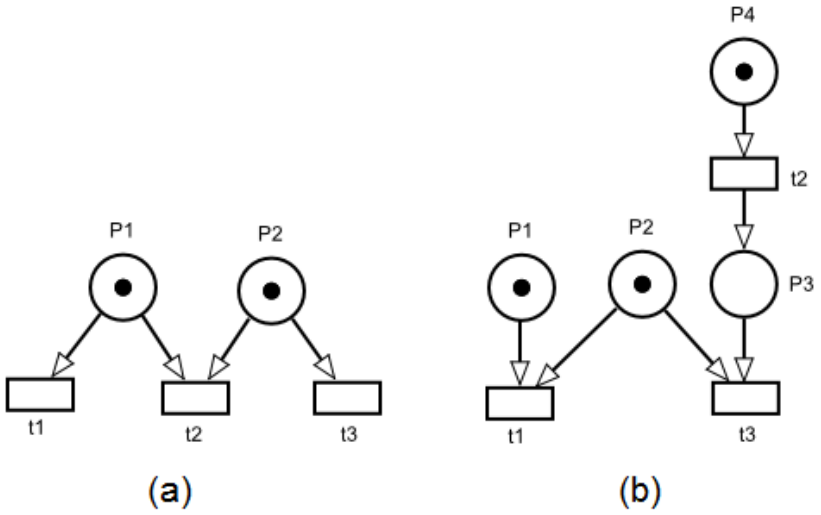


**Figure 2.16:** (a) symmetric confusion; (b) asymmetric confusion.

### Parallel processes

In order to represent parallel processes, a model may be obtained by composing the model for each individual process with a fork and synchronization models. Two transitions are said to be parallel (or concurrent), if they are causally independent, i.e., one transition may fire either before (or after) or in parallel with the other.

Figure 2.17 depicts an example of parallel process, where transitions $t_1$ and $t_2$ represent parallel activities. When transition $t_0$ fires, it creates marks in both output places ($p_0$ and $p_1$), representing a concurrency. When $t_1$ and $t_2$ are enabled for firing, each one may fire independently. The firing of $t_3$ depends on two pre-conditions, $p_2$ and $p_3$, implying that the system only continues if $t_1$ and $t_2$ have been fired.

Figure 2.17 presents a net in which each place has exactly one incoming arc and exactly one outgoing arc. Thus, such model represents a sub-class of Petri nets known as marked graphs. Marked graphs allow representation of concurrency but not decisions or conflicts.

## 2.4.4 Petri nets properties

The PN properties allow a detailed analysis of the modeled system. For this, two types of properties have been considered in a Petri net model: behavioral and structural properties. Behavioral properties are those which depend on the initial marking. Structural properties, on the other hand, are those that are marking-independent. The behavioral property reachability is explained in the following section. For other properties, the reader is redirected to [31].

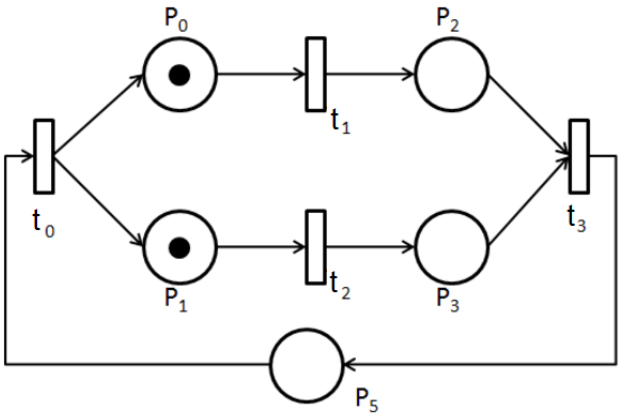


**Figure 2.17:** A Petri net representing parallel activities.

### Reachability

The firing of an enabled transition changes the token marking in a Petri net, and a sequence of firings results in a sequence of markings. A marking $M_n$ is said to be reachable from a marking $M_0$ if there exists a sequence of firings that transforms $M_0$ to $M_n$.

A firing (or occurrence) sequence is denoted by $\sigma = t_1, t_2, ..., t_n$. In this case, $m_i$ is reachable from $m_0$ by $\sigma$, and it is denoted by $m_0[\sigma > m_i$. The set of all possible reachable markings from $m_0$ in a net (PN, $m_0$) is denoted by R(PN,$m_0$), or simply R($m_0$). The set of all possible firing sequence from $m_0$ in a net (PN,$m_0$) is denoted by L(PN, $m_0$), or simply L($m_0$).

## 2.4.5 Petri Net Analysis Methods

Petri net analysis methods may be divided into three groups: the reachability tree method, analysis based on the matrix-equations and reduction techniques [31].

The first method involves essentially the enumeration of all reachable markings to build the reachability graph (reachability tree). This method can be applied to all classes of nets, however due to the complexity of the system modeled it may suffer from the state-space explosion issue. The reachability graph is initial marking dependent and so it is used to analyze behavioral properties. The main problem in using a reachability tree is the high computational complexity, even if some interesting techniques are used, such as reduced graphs, graph symmetries, symbolic graph, etc. The second method is based on state equations. The main advantage of this method, over the reachability graph, is

the existence of simple linear algebraic equations that aid in determining net properties. However, it gives only necessary or sufficient conditions to the analysis of properties when it is applied to general Petri nets. The third method is based on reduction laws. This method provides a set of transformation rules which reduces the size of models while preserves system's properties. However, it is possible that, for a given system and some set of rules, the reduction can not be completed. The matrix equations and reduction techniques are applicable only to special subclasses of Petri nets. Therefore, this work presents the analysis based on reachability tree.

### Reachability Based Methods

The analysis method namely Reachability Tree is based on the building of a tree that makes possible to represent all reachable markings of a net [32].

From the initial marking of a PN, it is possible to obtain some markings through the fireable transitions. Such possibilities can be represented as a tree, where the nodes correspond the markings and the arcs represent the fired transitions.

The reachability tree has been generated through initial marking of the net and adding directly reachable markings as leaves. Next, the process proceeds by these new markings in order to determine their directly reachable markings. These markings now become the new leaves of the already generated part of the reachability tree. If the desired marking is reached, it is not necessary to continue building the tree any further at that node. Such Reachability trees can be transformed directly into graphs by removing multiple nodes and connecting the nodes appropriately. This graph is called a reachability graph.

**Definition 2.4.11. (Reachability Tree)** Considering a Marked Petri net $MN = (N; M_0)$, a reachability tree is defined by RT = (S, A), where $S$ represents the markings and $A$ the labeled arcs by $t_j \in T$.

Some PN's properties such as boundedness, safetness, deadlock freedom and reachability can be analyzed through these reachability tree $T$ by adopting the following rules [31]:

(i) A Marked Petri net $(N; M_0)$ is bounded and thus $R(M_0)$ is finite if and only if (iff) $W$ (from the weight function - Definition 2.4.1) does not appear in any node labels in $T$;

(ii) A Marked Petri net $(N; M_0)$ is safe iff only 0's and 1's appear in code labels in $T$;

(iii) A transition $t$ is dead iff it does not appear as an arc label in $T$;

(iv) If $M$ is reachable from $M_0$, then there will be a node labeled $M'$ such that $M \leq M'$.

A major problem of this approach arises with the analysis of systems in which the number of reachable markings is infinite (unbounded systems). Due to the infinite number of markings, such systems are not easily represented by enumeration.

### 2.4.6 Time Extensions

The original definition of Petri nets does not include any notion of time and their aims are to model the logical behavior of systems by describing the causal relations between their events. Many researches have been proposing different ways for incorporating timing in Petri Nets, and the first ones related to them were presented by P.M Merlin et al. [33] and J.D Noe et al. [34]. In Timed Petri nets (TPNs), time may be associated to places, transitions or tokens [35] such that:

- *Places* : When the time is associated to places, the markings are only available after a determinate amount of time.
- *Token* : The time can be added to the token, in which it have an information indicating when the token will be available to fire a transition.
- *Transitions* : When the time is associated to a transition, the transition is only able to fire after the delay correspondent to the time associated to it.

In TPN, the time can be deterministic, stochastic or between intervals.

- *Deterministic* : In this case, a deterministic time is adopted to represent the events.
- *Interval Computations:* In this case, intervals are adopted in order to describe the higher and shorter limits related to the time of each activity.
- *Stochastic* : This model adopts a probabilistic approach.

Since transitions represent activities that change the state (marking) of the net, it seems natural to associate time to transitions. For this, there are two different firing policies in TPN:

- *Three-phase firing:* a first instantaneous phase in which an enabled transition removes tokens from its input places, then a timed phase in which the transitions are working, and a final instantaneous phase in which tokens are deposited into the output places. Such time information is called duration;
- *Atomic firing:* Tokens remain in input places during the whole transition delay; after that period such tokens are consumed from input places and generated in output places when the transition fires. The firing itself does not consume any time.

In atomic firing, when a transition is able to fire, a timer associated to the transition is started. Such timer decreases in a constant way, and the transition is fired when the timer value goes to zero. There is an issue related to the other transitions timers and, in order to solve such issue, the following approaches have been adopted to represent the memory policies whenever a transition fires [35]:

- *Resampling* : the timers of all transitions are discarded (restart mechanism). New values of timers are reset for all enabled transitions at a new marking;
- *Enabling memory:* transitions that are still enabled in the new marking keeps the value of the timer; transitions that are not enabled have their timers reset. The enabling time of a transition is measured since the last instant of time it became enabled;
- *Age memory:* the timer value is kept, even if the transition is not enabled in the new marking. Whenever this transition becomes enabled, the counting is resumed from the kept value.

### 2.4.7 Stochastic Petri nets

Petri nets [36] are a classic tool for modeling and analyzing discrete event systems that are too complex to be described by automata or queueing models. Time (stochastic delays) and probabilistic choices are essential aspects for a performance evaluation model. In this work, we adopt the usual association of delays and weights to transitions [37]. The transition firing times in SPNs correspond to exponential distribution. Due to the memoryless property of this distribution, the stochastic process associated with SPNs corresponds to a continuous time Markov chain. The Markov chain can be easily constructed from the reachability graph given the firing rates of the transitions of the SPN. The extended stochastic Petri net [38] definition adopted in this work is:

Let $SPN = (P, T, I, O, H, \Pi, M_0, Atts)$ be a stochastic Petri net, where

- $P = \{p_1, p_2, ..., p_n\}$ is the set of places, which may contain tokens and form the discrete state variables of a Petri net.
- $T = \{t_1, t_2, ..., t_m\}$ is the set of transitions, which model active components.
- $I \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities of input arcs, where $i_{jk}$ entry of $I$ gives the (possibly marking-dependent) arc multiplicity of input arcs from place $p_j$ to transition $t_k$ [$A \subseteq (P \times T) \cup (T \times P)$ — set of arcs]. A transition is only enabled if there are enough tokens in all input places.
- $O \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ is a matrix of marking dependent multiplicities of output arcs, where $o_{jk}$ entry of $O$ specifies the possibly marking-dependent arc multiplicity of

output arcs from transition $t_j$ to place $p_k$. When a transition fires, it removes the number of tokens specified by the input arcs from input places, and adds the amount of tokens given by the output arcs to all output places.

- $H \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities describing the inhibitor arcs, where $h_{jk}$ entry of $H$ returns the possibly marking-dependent arc multiplicity of an inhibitor arc from place $p_j$to transition $t_k$. In the presence of an inhibitor arc, a transition is enabled to fire only if every place connected by an inhibitor arc contains fewer tokens than the multiplicity of the arc.
- $\Pi \in \mathbb{N}^m$ is a vector that assigns a priority level to each transition. Whenever there are several transitions fireable at one point in time, the one with the highest priority fires first and leads to a state change.
- $M_0 \in \mathbb{N}^n$ is a vector that contains the initial marking for each place (initial state).
- $Atts : (Dist, W, G, Policy, Concurrency)^m$ comprises a set of attributes for the $m$ transitions, where
  - $Dist \in \mathbb{N}^m \to \mathcal{F}$ is a possibly marking dependent firing probability distribution function. In a stochastic timed Petri net, time has to elapse between the enabling and firing of a transition. The actual firing time is a random variable, for which the distribution is specified by $\mathcal{F}$. We differ between immediate transitions ($\mathcal{F} = 0$) and timed transitions, for which the domain of $\mathcal{F}$ is $(0, \infty)$.
  - $W \in \mathbb{R}^+$ is the weight function, that represents a firing weight $w_t$ for immediate transitions or a rate $\lambda_t$ for timed transitions. The latter is only meaningful for the standard case of timed transitions with exponentially distributed firing delays. For immediate transitions, the value specifies a relative probability to fire the transition when there are several immediate transitions enabled in a marking, and all have the same probability. A random choice is then applied using the probabilities $w_t$.
  - $G \in \mathbb{N}^n \to \{true, false\}$ is a function that assigns a guard condition related to place markings to each transition. Depending on the current marking, transitions may not fire (they are disabled) when the guard function returns false. This is an extension of inhibitor arcs.
  - $Policy \in \{prd, prs\}$ is the preemption policy (*prd — preemptive repeat different* means that when a preempted transition becomes enabled again the previously elapsed firing time is lost; *prs — preemptive resume*, in which the firing time related to a preempted transition is resumed when the transition becomes enabled again),
  - $Concurrency \in \{ss, is\}$ is the concurrency degree of transitions, where $ss$ represents single server semantics and $is$ depicts infinity server semantics in the same sense as in queueing models. Transitions with policy $is$ can be understood as having an individual transition for each set of input tokens, all running in parallel.

In many circumstances, it might be suitable to represent the initial marking as a mapping from the set of places to natural numbers ($m_0 : P \rightarrow \mathbb{N}$), where $m_0(p_i)$ denotes the initial marking of place $p_i$ and $m(p_i)$ denotes a reachable marking (reachable state) of place $p_i$. In this work, the notation $\#p_i$ has also been adopted for representing $m(p_i)$.

**SPN modeling example**

A simple multi-processor system in which the processor requesting and gaining access to the common memory can be modeled through SPN. A processor executes locally for some time (mean duration $1/\lambda$), and then requests access to common memory (gaining access has mean duration 1/r). Once it has gained access, the duration of common memory access is assumed to be $1/\mu$ on average. We can model the behavior of a single processor interacting with the common memory using the SPN depicted in Figure 2.18. Figure 2.19 shows the correspondent reachability graph for that SPN model. In the SPN:

- place *Executing* represents the local state of the processor when it is executing;
- place *Requesting* corresponds to the state of the processor when it is ready to start accessing to the common memory;
- place *Acessing* is the state when the process is using the common memory;
- place *Memory* represents the local state of the common memory when it is not in use;
- transition $T_0$ models the action of the processor executing. The rate of this transition is $\lambda$;
- transition $T_1$ represents the processor gaining access to the common memory. The rate of this transition is r;
- transition $T_2$ represents the processor accessing the common memory. The rate of this transition is $\mu$.

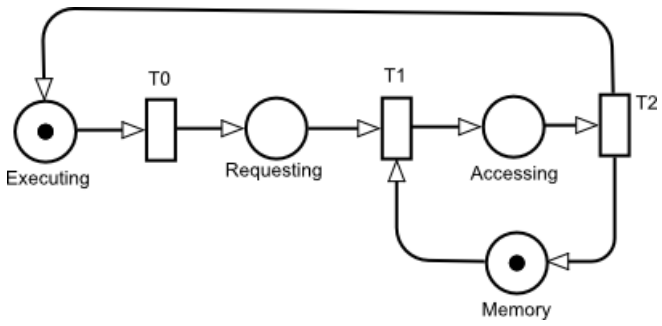


**Figure 2.18:** SPN representing a single processor in a shared memory system.

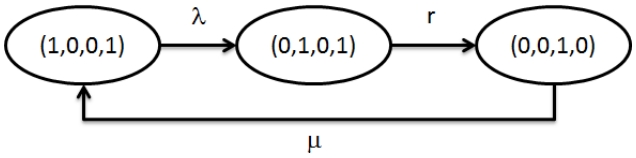


**Figure 2.19:** Reachability Graph of the SPN model of Figure 2.18.

### Time Distributions

Figure 2.20 depicts the generic simple component model using SPN, which provides a high-level representation of a subsystem. One should notice the trapezoidal shape of transitions (high-level transition named s-transition). This shape means that the time distributions of such transitions are not exponentially distributed, instead they should be refined by subnets. The delay assigned to s-transition $f$ is the $TTF$ and the delay of s-transition $r$ is the $TTR$. If the $TTF$ and $TTR$ are exponentially distributed, the shape of the transitions should be the regular one (white rectangles) and $TTF$ and $TTR$ should be summarized by the respective $MTTF$ and $MTTR$.

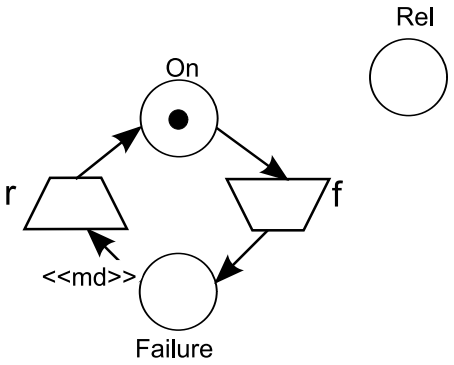


**Figure 2.20:** Generic simple model - SPN

A well-established method that considers expolynomial distribution random variables is based on distribution moment matching. The moment matching process presented in [24] takes into account that Hypoexponential and Erlangian distributions have the average delay ($\mu$) greater than the standard-deviation ($\sigma$) -$\mu > \sigma$-, and Hyperexponential distributions have $\mu<\sigma$, in order to represent an activity with a generally distributed delay as an Erlangian or a Hyperexponential subnet referred to as s-transition[1]. One should note that in cases where these distributions have $\mu = \sigma$, they are, indeed, equivalent to an exponential distribution with parameter equal to $\frac{1}{\mu}$. Therefore, according to the coefficient of variation associated with an activity's delay, an appropriate s-transition

[1]In this work, $\mu$ could be $MTTF$ or $MTTR$ and the $\sigma$ could represent $sd(TTF)$ or $sd(TTR)$, for instance.

implementation model could be chosen. For each s-transition implementation model (see Figure 2.21), a set of parameters should be configured for matching their first and second moments. In other words, an associated delay distribution (it might have been obtained by a measuring process) of the original activity is matched with the first and second moments of s-transition (expolynomial distribution). According to the aforementioned method, one activity with $\mu<\sigma$ is approximated by a two-phase Hyperexponential distribution with parameters

$$r_1 = \frac{2\mu^2}{(\mu^2+\sigma^2)}, \tag{2.22}$$

$$r_2 = 1 - r_1 \tag{2.23}$$

and

$$\lambda = \frac{2\mu}{(\mu^2+\sigma^2)}, \tag{2.24}$$

where $\lambda$ is the rate associated to phase 1, $r_1$ is the probability of related to this phase, and $r_2$ is the probability assigned to phase 2. In this particular model, the rate assigned to phase 2 is assumed to be infinity, that is, the related average delay is zero.

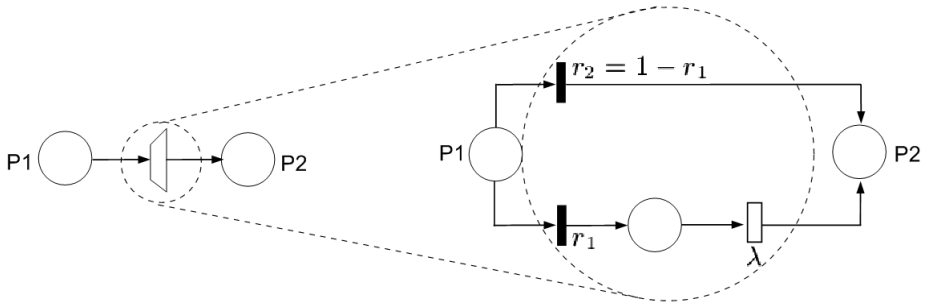


**Figure 2.21:** Hyperexponential Model

Activities with coefficients of variation less than one might be mapped either to Hypoexponential or Erlangian s-transitions. If $\frac{\mu}{\sigma} \notin \mathbb{N}, \frac{\mu}{\sigma} \neq 1, (\mu, \sigma \neq 0)$, the respective activity is represented by a Hypoexponential distribution with parameters $\lambda_1$, $\lambda_2$(exponential rates); and $\gamma$, the integer representing the number of phases with rate equal to $\lambda_2$, whereas the number of phases with rate equal to $\lambda_1$ is one. In other words, the s-transition is represented by a subnet composed of two exponential and one immediate transitions. The average delay assigned to the exponential transition $t_1$ is equal to $\mu_1$ ($\lambda_1 = 1/\mu_1$), and the respective average delay assigned to the exponential transition $t_2$ is $\mu_2(\lambda_2 = 1/\mu_2)$. $\gamma$ is the integer value considered as the weight assigned to the output arc of transition $t_1$ as well as the input arc weight value of the immediate transition $t_3$ (see Figure 2.22).

These parameters are calculated by the following expressions:

$$(\frac{\mu}{\sigma})^2 \quad - \quad 1 \leq \gamma < (\frac{\mu}{\sigma})^2, \tag{2.25}$$

$$\lambda_1 \quad = \quad \frac{1}{\mu_1} \; and \, _2 = \frac{1}{\mu_2}, \tag{2.26}$$

where

$$\mu_1 \quad = \quad \frac{\mu \pm \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1}, \tag{2.27}$$

$$\mu_2 \quad = \quad \frac{\gamma\mu \mp \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{2.28}$$

If $\frac{\mu}{\sigma} \in \mathbb{N}, \frac{\mu}{\sigma} \neq 1, (\mu, \sigma \neq 0)$, an Erlangian s-transition with two parameters, $\gamma = (\frac{\mu}{\sigma})^2$ is an integer representing the number of phases of this distribution; and $\mu_1 = \mu/\gamma$, where $\mu_1(1/\lambda_1)$ is the average delay value of each phase. The Erlangian model is a particular case of a Hypoexponential model, in which each individual phase rate has the same value.

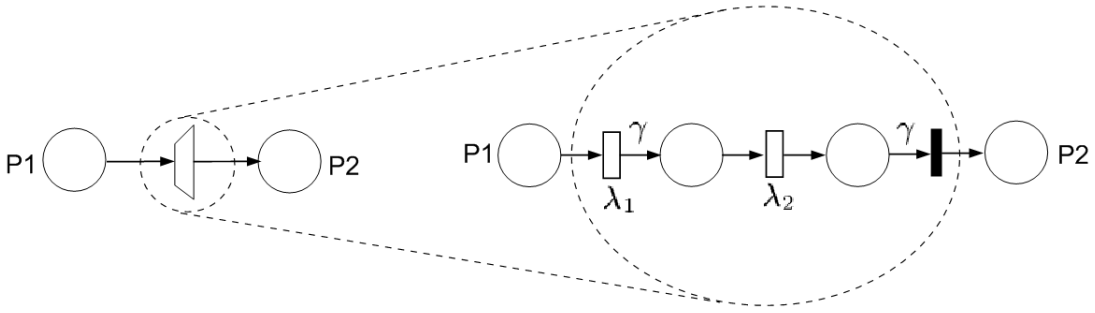


**Figure 2.22:** Hypoexponential Model

For the sake of simplicity, the SPN models presented in the next chapters consider only exponential distributions.

## 2.5 RELIABILITY BLOCK DIAGRAMS

Reliability Block Diagram (RBD) is a combinatorial model that was initially proposed as a technique for calculating reliability of systems by intuitive block diagrams. The block diagrams provide a graphical description of the system components and connectors, which can be adopted to determine the overall system state given the state of its components. The structure of RBD establishes the logical interaction among components defining which combinations of failed and active elements are able to sustain system operation. More specifically, the system is represented by subsystems or components connected according to their function or reliability relationship [39]. Such a technique has also been

extended to calculate other dependability metrics, such as availability and maintainability [40].

In RBD models, it is possible to represent a physical component in the operational mode by a block. On the other hand, to represent a failure of a component, it is necessary to remove the correspondent block of the component that has failed. If there is at least one path connecting input and output points, the system is still operating properly. In other words, if enough blocks are removed in an RBD to interrupt the connection between the input and output points, the system fails [9] [41].

### 2.5.1 Structure Properties

In this section, the arrangement of components in a system as well as the structure function for determining if the system is functioning or not is presented. The structure functions are adopted to present the relationship of individual components and the state of the system. Assume that both components and system can either be functioning or failed. The failed and functioning states (of both system and components) are denoted by 0 and 1, respectively.

**Definition 2.5.1.** The state of component $i$ is denoted by $x_i$ as follows:

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is functioning} \\ 0 & \text{if component } i \text{ has failed} \end{cases}$$

for $i$= 1, 2, ..., $n$.

where: $n$ represents the number of components.

Additionally, a vector $x = (x_1, x_2, ..., x_n)$ may be adopted to represent the system state taking into account those $n$ component values. The structure function $\phi$ maps the system state vector $x$ to 1 or 0 as shown below:

**Definition 2.5.2.** The function $\phi$ determines the system state:

$$\phi(x) = \begin{cases} 1, & \text{if the system is functioning} \\ 0, & \text{if the system has failed} \end{cases}$$

The blocks (e.g., components) are usually arranged using the following composition mechanisms: series, parallel, bridge, $k$-out-of-$n$ blocks, or, even, a combination of previous approaches. The following lines detail each one of those composition methods.

**Series system**

Figure 2.23 shows a reliability block diagram of series system, which only functions if all of its components are functioning. Therefore, the structure function $\phi(x)$ assumes the value 1 when $x_1 = x_2 = ... = x_n = 1$, and 0 otherwise. These can be represented by three different ways as shown below:

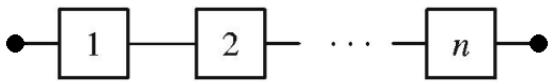


**Figure 2.23:** Reliability block diagram of series system.

**Definition 2.5.3.** The function that determines the system state is:

$$\phi(x) = \begin{cases} 1, & \text{if } x_i = 1 \text{ for all i=1, 2, ..., n} \\ 0, & \text{if there exists an i such that } x_i{=}0 \end{cases}$$

$$\phi(x) = min(x_1, x_2, ..., x_n)$$

$$\phi(x) = \prod_{i=1}^{n} x_i$$

The three ways of defining the function $\phi(x)$ that represents series system are equivalent. However, the third option is more adopted due to its compactness [42].

**Parallel system**

Figure 2.24 shows a reliability block diagram of series system, which only functions if one or more of its components are functioning. Therefore, the structure function $\phi(x)$ assumes the value 0 when $x_1 = x_2 = ... = x_n = 0$, and 1 otherwise. These can be represented by different ways as shown below:

**Definition 2.5.4.** The function that determines the system state is:

$$\phi(x) = \begin{cases} 1, & \text{if there exists an i such that } x_i{=}1 \\ 0, & \text{if } x_i = 0 \text{ for all i=1, 2, ..., n} \end{cases}$$

$$\phi(x) = max(x_1, x_2, ..., x_n)$$

$$\phi(x) = 1 - \prod_{i=1}^{n}(1 - x_i)$$

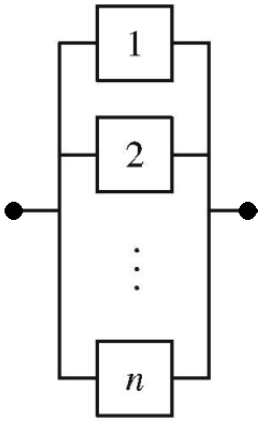


**Figure 2.24:** Reliability block diagram of parallel system.

Similarly to the series system definition, the three function $\phi(x)$ definitions are equivalent. However, the third option is more adopted due to its compactness.

### K-out-of-n system

A k-out-of-n system functions if and only if $k$ or more of its $n$ components are functioning. Therefore, series and parallel systems represent particular cases of k-out-of-n systems. The series system is an n-out-of-n system (all components of the system must be functioning) whereas parallel system is an 1-out-of-n system (at least one components must be functioning). The function $\phi$ for a k-out-of-n system is defined as follows:

**Definition 2.5.5.** The function that determines the system state is:

$$\phi(x) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} x_i \geq k \\ 0, & \text{if } \sum_{i=1}^{n} x_i < k \end{cases}$$

Figure 2.25 depicts a RBD example of 2-out-of-3 system. The block diagram indicates that if at least two out of 3 components are functioning (e.g., 1 and 2, or 1 and 3, or 2 and 3) the system is also functioning. Thus, the structure function $\phi(x)$ for a 2-out-of-3 system is: $\phi(x) = 1 - (1 - x_1x_2)(1 - x_1x_3)(1 - x_2x_3)$.

Let $p$ be the success probability of each of those blocks. The system success probability (reliability or availability) is depicted by

$$\Sigma_{i=k}^{n} \binom{n}{i} p^k (1-p)^{n-k} \tag{2.29}$$

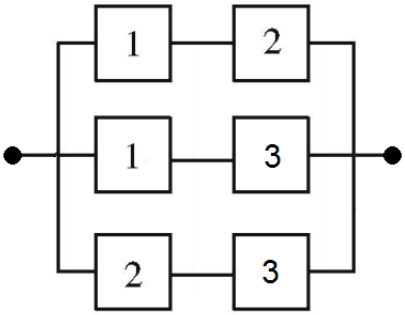


**Figure 2.25:** Reliability block diagram of 2-out-of-3 system.

### Bridge system

To find the structure function related to the bridge system depicted in Figure 2.26, we should create a block diagram considering repeated components. It is possible to note that the system represented on that bridge system operates when at least specific components are functioning (e.g., (1,3,5); (1,4); (2,3,4); (2,5)). Therefore, an alternative block diagram that is equivalent (in the failure point of view) is the one shown in Figure 2.27. The correspondent structure function is then $\phi(x) = 1-(1-x_1x_3x_5)(1-x_1x_4)(1-x_2x_3x_4)(1-x_2x_5)$

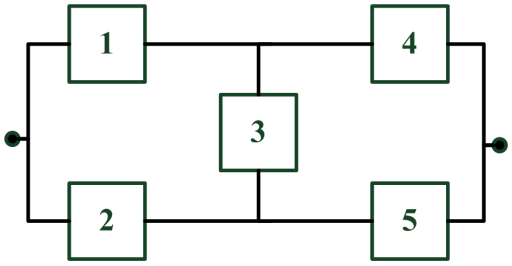


**Figure 2.26:** Reliability block diagram of bridge system.

## 2.5.2 Reliability Functions

In this section, we present the reliability functions of those previous structures. In order to compute reliability of a system, two assumptions must be considered. The first one is related to the fact that the components are nonrepairable. The structure functions previously presented may be adopted for both repairable and nonrepairable system since that functions only relates the components state with the system one. Another important assumption is related to the components independence in the sense that a failure of one device does not impacts the probability of failure of the others.

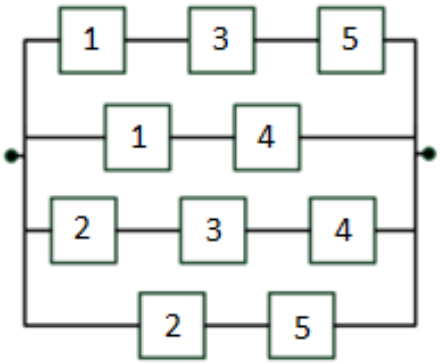


**Figure 2.27:** Reliability block diagram of bridge system.

Different from the structure functions, to compute the system reliability *(R(t))* a time $t$ must be considered. For instance, assuming independent failures, Equation 2.30 is adopted for computing the reliability of series system. Therefore, the system reliability cannot be greater than the smallest component reliability.

$$R_s(t) = \prod_{i=1}^{n} R_i(t) = exp(-\lambda_s t), \tag{2.30}$$

where $R_i(t)$ is the probability that the $i$th component does not fail before time $t$; $\lambda_s = \sum_{i=1}^{n} \lambda_i$; $\lambda_i =$ is the failure rate of the $i$th component.

Additionally, the reliability of parallel systems can be obtained taking 1 minus the probability that all $n$ components fail (i.e., probability that at least one component does not fail). Equation 2.31 computes the reliability of parallel systems. For other examples and closed-form equations, the reader should refer to [9] [23].

$$R_s(t) = 1 - \prod_{i=1}^{n}(1 - R_i(t)) = 1 - \prod_{i=1}^{n}(1 - e^{-\lambda_i t}). \tag{2.31}$$

### Combined series-parallel systems

Usually, systems are composed of both series and parallel arrangement. Consider the example depicted in Figure 2.28. To compute the system reliability, the block diagram may be analyzed piecewise. For instance, starting from the subsystem A (composed of two components in parallel); going to the subsystem in series (e.g., analyzing regions A and B). Finally, the whole system can be analyzed for computing the system reliability. Thus, the system reliability is computed as follows:

$R_A = 1 - (1 - R_2)(1 - R_3)$

$R_B = (R_A)(R_3)$

$$R_S = [1 - (1 - R_B)(1 - R_4)](R_5)$$

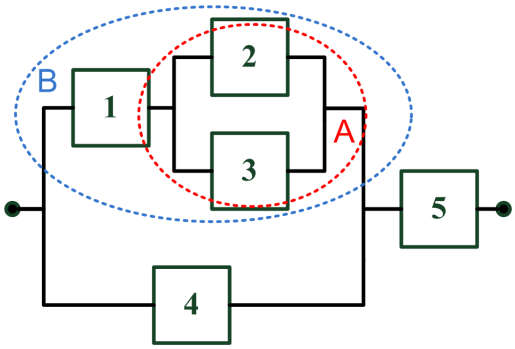


**Figure 2.28:** Reliability block diagram of series and parallel systems combined.

### Levels of redundancy

System redundancy may be achieved by two different approaches: (i) low-level and (ii) high-level redundancies. The first case considers one or more parallel components, whereas the high-level replicates the entire system. For instance, let us consider a system composed of two serial components (1 and 2) which adopts the low-level redundancy as shown in Figure 2.29. Similarly, Figure 2.30 depicts two components in high-level redundancy.

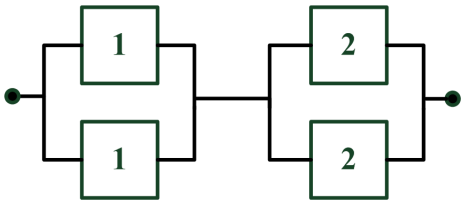


**Figure 2.29:** Reliability block diagram of low-level redundancy.

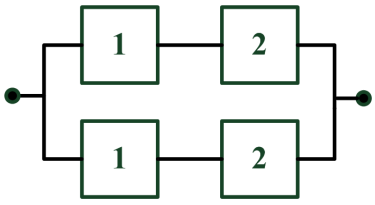


**Figure 2.30:** Reliability block diagram of high-level redundancy.

An interesting question is related to which redundancy method has the higher reliability values. There can be no doubt that for both redundant systems, in case both

components 1 or both components 2 fails, the system will also fail. The reader should observe that the high-level redundancy system may fail if one component 1 and one component 2 fails. It is important to state that this behavior does not happen in the low-level redundancy. Therefore, low-level redundant systems have higher reliability than high-level redundancy.

The following lines present the reliability for both low-level and high-level redundant systems. Assume that both components (1 and 2) have the same reliability $R$. The reliability of low-level redundancy is:

$R_{low} = [1 - (1 - R)^2]^2 = [1 - (1 - 2R + R^2)]^2 = (2R - R^2)^2$.

Considering high-level redundancy, the reliability is:

$R_{high} = 1 - (1 - R^2)^2 = 1 - (1 - 2R^2 + R^4) = 2R^2 - R^4$.

Additionally, Table 2.6 shows a comparison between the reliability values obtained for those two examples considering low and high-level redundancies. It is possible to note that when the reliability of those components ($R$) is greater, the reliability difference computed for those different redundant systems is reduced.

**Table 2.6:** Reliability comparison of low and high-level redundancies.

| R | low-level | high-level | difference |
|---|---|---|---|
| 0.8 | 0.870400000000 | 0.921600000000 | 0.051200000000 |
| 0.9 | 0.963900000000 | 0.980100000000 | 0.016200000000 |
| 0.99 | 0.999603990000 | 0.999800010000 | 0.000196020000 |
| 0.999 | 0.999996003999 | 0.999998000001 | 0.000001996002 |
| 0.9999 | 0.999999960004 | 0.999999980000 | 0.000000019996 |

For other examples and closed-form equations, the reader should refer to [9].

## 2.6 OPTIMIZATION

Many real-world problems can be modeled as an optimization problem that seeks to maximize or minimize a mathematical function of a number of variables while respecting the system constraints [43]. This mathematical function that is optimized is called objective function. The objective function can be composed of a single or multi variables depending on the optimization problem.

### Model

A general optimization model can be represented as follows:

**Find x to**

**Maximize** $f(\mathbf{x})$

**Subject to:**

$g_i(\mathbf{x}) \leq 0$, **i=1, ..., m**

$h_j(\mathbf{x}) = 0$, **j=1, ..., p**

where $\mathbf{x}$ is an n-dimensional vector named design vector, $f(\mathbf{x})$ is the objective function, and $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are the inequality and equality constraints, respectively. The number of variables $n$, the number of constrains $m$ and $p$ are not necessary related. The model represented above is know as constrained optimization problem [44].

Let $\overline{x}$ be a set of variables as $\overline{x}$=($x_1$, $x_2$, ..., $x_n$), then the optimization model for multiple variables can be defined as follows:

**Maximize** $f(\overline{\mathbf{x}})$

**Subject to:**

$g_i(\overline{\mathbf{x}}) \leq gb_i$ , i=1, ..., m

$h_j(\overline{\mathbf{x}}) = hb_j$ , j=1, ..., p

Assume that $g_i(\overline{x})$ and $f(\overline{x})$ are linear functions, then we can represent those functions as follows:

$f(\overline{\mathbf{x}}) = c_1x_1 + c_2x_2 + ... + c_nx_n$

*and*

$g_1(\overline{\mathbf{x}}) = a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \leq gb_1$

$g_2(\overline{\mathbf{x}}) = a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \leq gb_2$

...

Where $c_i$ and $a_{in}$ are the coefficients of the objective and constraint functions, respectively.

### 2.6.1 GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) [45] is an optimization algorithm that has been applied for combinatorial optimization problems [46]. GRASP is able to achieve values close to the optimal results[47] without analyzing all the possible solutions for the problem. In computer science, covering problems [48] represent the problems that ask whether a certain combinatorial structure "covers" another, or how large the structure has to be to do that. Covering problems are minimization problems and usually linear programs.

The GRASP algorithm consists basically of two phases: construction and local search. The construction phase builds a feasible solution in each iteration of the algorithm, whose neighborhood is investigated until a local minimum is found during the local search phase. The best solution overall iterations is returned as the result.

Similarly to other optimization methods, the algorithm starts from an initial solution and then perform local searches to improve the quality of the first solution. In order to accomplish this, greedy randomized procedures are considered and then the local search is performed from the constructed model/solution. This two-phase process is repeated until the stopping condition is satisfied.

## 2.7 SUMMARY

This chapter presented concepts related to the proposed set of models, ranging from the definition of data center infrastructures to the Petri net formalism. Initially, data center systems were presented focusing on the relation between each infrastructure that compose a typical data center. Afterwards, the concept of dependability and its application was presented. Next, sustainability was conceptualized in the context of data centers. After that, attention was devoted to Petri nets models, giving particular focus to stochastic Petri nets. SPNs are a family of formalisms very suitable for dependability modeling of systems. Next, reliability block diagrams were presented focusing in dependability evaluation of systems. Finally a brief review on the concepts related to optimization techniques was performed with the focus on the adopted optimization method GRASP.

CHAPTER 3

# MODELS

In this Chapter, we present the models adopted for verifying the energy flow as well as to quantify system dependability, cost and sustainability of data center infrastructures. First, we present the energy flow model, which verifies the energy flow between the system components. Next, we describe the sustainability model, presenting equations for estimating the operational exergy destroyed. Finally, the dependability models, which are composed of RBD and SPN, and the proposed optimization model are presented.

## 3.1 ENERGY FLOW MODEL

This model represents the energy flow between the system components considering the respective efficiency and the maximum energy that each component can provide (considering electrical devices) or extract (assuming cooling devices).

The system under evaluation can be correctly arranged, in the sense that the required components are properly connected, but they may not be able to meet system demand for electrical energy or thermal load. For example, consider the power infrastructure depicted in Figure 3.1 (a). Assuming the demanded power by the IT data center system corresponds to 20kW (value associated to the target node $T$), and the maximum power capacity (Figure 3.1 (b)) of the internal nodes (i.e., components) UPSs and Power Strips are for both 15kW. Figure 3.1 (c) depicts a possible energy flow, in which the energy provided by each UPS is 10kW which is transferred to the Power Strips (10 kW). In another example, instead of adopting two Power Strips, only one is considered. This system is depicted in Figure 3.1 (d) and it is possible to support 15kW of the demanded power (value associated to the target node $T$). Thus, the system is not able to cope with the demanded power.

Figure 3.1 (e) depicts a system with two Power Strips (Power Strip1 and Power Strip 2). The Power Strip1 is able to provide three times more power than the Power Strip2. This system behavior is specified by weights on the edges of the graph.

Figure 3.2 (a) shows another example considering a data center cooling infrastructure. Assume the amount of heat that should be extracted from the data center room is 10kW (the value associated with the source node $S$), and the maximum cooling capacity (Figure 3.2 (b)) of the internal nodes (i.e., components) CRACs and chiller are 8kW and 18kW, respectively. Figure 3.2 (c) depicts a possible energetic flow, in which heat is extracted by both CRACs (with a load of 5 kW in each component) and transferred to the chiller

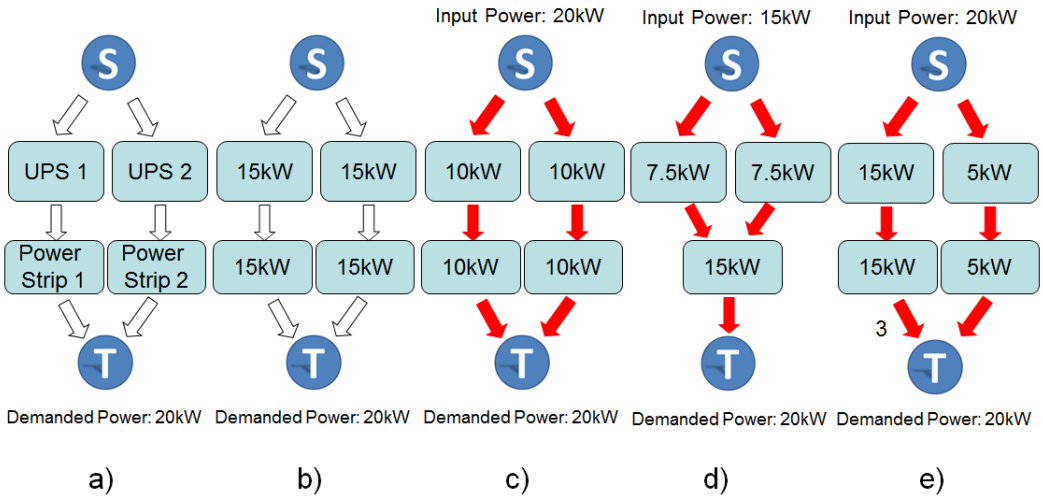


**Figure 3.1:** a) Power System example; b) Maximum Power Capacity; c) Successful Energy Flow; d) Failed Energy Flow; e) Representation with Weight.

(10 kW). The reader should notice here that the filled edges are representing the energy flowing. Therefore, different from power systems (Figure 3.1), the edges are directed from the source node $S$ to the target node $T$ when EFM is adopted for modeling cooling systems. Figure 3.2 (d) depicts another example, in which instead of using two CRACs, only one is considered. The system depicted in Figure 3.2 (d) is able to extract 8kW of heat (the value associated to the target node $T$). Thus, the system is not able to cope with the required thermal load.

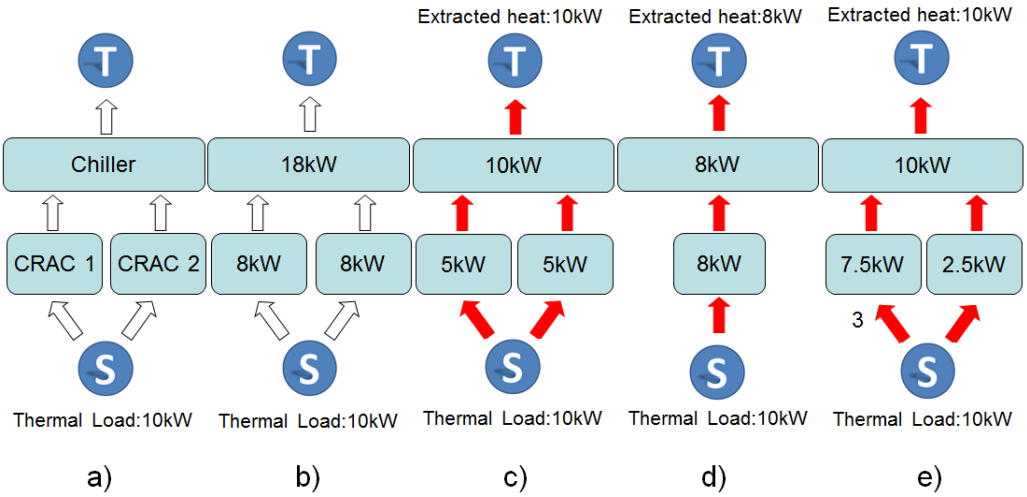


**Figure 3.2:** a) Cooling System example; b) Maximum Cooling Capacity; c) Successful Energetic Flow; d) Failed Energetic Flow; e) Representation with Weight.

Figure 3.2 (e) depicts a system with 2 CRACs (CRAC1 and CRAC 2). The CRAC1 extracts three times more heat than the CRAC2. Algorithms are proposed to compute the cost and sustainability impact thought the EFM as further detailed in the following sections. In this work, an energy flow model (EFM) is proposed to represent the energy flow. This model is a directed acyclic graph and is defined as follows:

$G = (N, A, w, f_d, f_c, f_p, f_\eta)$, where:

- $N = N_s \cup N_i \cup N_t$ represents the set of nodes (i.e., the components), in which $N_s$ is the set of source nodes, $N_t$ is the set of target nodes and $N_i$ denotes the set of internal nodes, $N_s \cap N_i = N_s \cap N_t = N_i \cap N_t = \oslash$;
- $A \subseteq (N_s \times N_i) \cup (N_i \times N_t) \cup (N_i \times N_i) = \{(a,b) \mid a \neq b\}$ denotes de set of edges (i.e., the component connections);
- $w : A \rightarrow \mathbf{R}^+$ is a function that assigns weights to the edges (the value assigned to the edge $(j, k)$ is adopted for distributing the energy assigned to the node $j$ to the node $k$ according to the ratio $w(j,k)/\sum_{i \in j^\bullet} w(j, i)$, where $j^\bullet$ is the set of output nodes of $j$);
- $f_d : N \rightarrow \begin{cases} \mathbf{R}^+ & \text{if } n \in N_s \cup N_t, \\ 0 & otherwise; \end{cases}$
  is a function that assigns to each node the heat to be extracted (considering cooling models) or the energy to be supplied (regarding power models);
- $f_c : N \rightarrow \begin{cases} 0 & \text{if } n \in N_s \cup N_t, \\ \mathbf{R}^+ & otherwise; \end{cases}$
  is a function that assigns each node with the respective maximum energy capacity;
- $f_p : N \rightarrow \begin{cases} 0 & \text{if } n \in N_s \cup N_t, \\ \mathbf{R}^+ & otherwise; \end{cases}$
  is a function that assigns each node (a node represents a component) with its retail price;
- $f_\eta : N \rightarrow \begin{cases} 1 & \text{if } n \in N_s \cup N_t, \\ 0 \leq k \leq 1, k \in \mathbf{R} & otherwise; \end{cases}$
  is a function that assigns each node with the energetic efficiency;

**Verifying the Energy Flow**

An algorithm is proposed to verify the power capacity of each component in a system represented by the Energy Flow Model. The algorithm checks whether the demanded power does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling devices). The algorithm adopts a depth-first approach to traverse the graph. In addition, a vector ($ccu$), which

has the size of the number of internal nodes, is adopted to associate to each node $i \in N_i$ the current power capacity used of the component.

**Algorithm 1** *initializeEnergyFlow(G, m)*

1: **if** $(m = cooling)$ **then**
2: $S_s := N_s$;
3: **else**
4: $S_s := N_t$;
5: **end if**
6: **for** $i \in N_i$ **do**
7: $ccu_i := 0$;
8: **end for**
9: $result := FALSE$;
10: **for** $n \in S_s$ **do**
11: $result := verifyEnergyFlow(G, f_d(n), n, m)$ :
12: **if** $result = FALSE$ **then**
13: **break;**
14: **end if**
15: **end for**
16: **return** $result$;

In the example depicted in Figure 3.2, only one source node ($S$) and one target node ($T$) are adopted. However, a set of source nodes ($N_s$) representing different electrical or heating energy sources may be adopted. Similarly, a set of target nodes ($N_t$) may also be considered.

In the case of power systems, the analysis starts from target nodes (or from source nodes in the case of cooling systems). The Energy Flow Algorithm (Algorithm 1) begins by checking the type of model to be analyzed (in the line 1, assume $m = \{cooling, power\}$). In the example depicted in Figure 3.2, the set of starting nodes ($S_s$) corresponds to $N_s$ (cooling model). Line 7 initializes the vector *ccu* with zero. The algorithm proceeds by performing calls to the function *verifyEnergyFlow* (line 11).

The function *verifyEnergyFlow* (Algorithm 2) starts by checking the type of model under analysis to determine the set $O_n$. Assuming cooling systems, $O_n$ represents the output nodes of the node $n$ (or the input nodes, considering power systems). In the example depicted in Figure 3.2, the source node ($n = S$) is the first node to be visited in $G$. The respective set of output nodes ($O_n$) is obtained in line 2. Line 6 tests if $O_n$ is

**Algorithm 2** *verifyEnergyFlow(G, $d_n$, n, m)*

1: **if** $(m = cooling)$ **then**
2: $O_n := \{o|(n,o) \in A; n \in N_s \cup N_i; o \in N_i\}$;
3: **else**
4: $O_n := \{o|(n,o) \in A; n \in N_t \cup N_i; o \in N_i\}$;
5: **end if**
6: **if** $(O_n = \emptyset)$ **then**
7: **return** $TRUE$;
8: **end if**
9: $ws := \Sigma_{o \in O_n}\ w(n,o)$;
10: **for** $o \in O_n$ **do**
11: **if** $\left(f_c(o) \geq (d_n \times \frac{w(n,o)}{ws}) + ccu_o\right)$ **then**
12: $ccu_o := ccu_o + (d_n \times \frac{w(n,o)}{ws})$;
13: **else**
14: **return** $FALSE$;
15: **end if**
16: **end for**
17: **for** $o \in O_n$ **do**
18: $verifyEnergyFlow(G,\ out(o,\ d_n \times \frac{w(n,o)}{ws}),\ o, m)$;
19: **end for**

empty, in which case it returns TRUE. The node $n$ has two output edges that connect it to the internal nodes CRAC1 and CRAC2. Next, the sum of $n's$ output edge weights ($ws$) is calculated to estimate the amount of energy that should be extracted from $n$ (by the cooling system).

For each $o \in O_n$ (line 10), the function *verifyEnergyFlow* checks if the output node has capacity to cope with the demanded power. Function $f_c(o)$ represents the maximum cooling or power capacity of a component (information supplied by the designer); $d_n$ represents the demanded energy transferred to (e.g, electricity) or extracted (e.g., heat) from the node $n$; $w(n,o)/ws$ is the ratio of the weight of the edge that connects node $n$ and node $o$ to the sum of weights of all output nodes of $n$. Thus, line 11 evaluates the capacity of the node $o$ to support an additional demand from a node $n$. The function *verifyEnergyFlow* may terminate and return FALSE (meaning that the demanded power is greater than the maximum capacity of node $o$). Otherwise, $ccu$ (of the node $o$) is updated (line 12).

**Table 3.1:** Output power of different devices.

| **Device** | **Equation** |
|---|---|
| Electrical | $\frac{E_c}{\eta}$ |
| CRAC | $\frac{1}{\eta} \times Q_{IN}$ |
| Chiller | $Q_F \times \left(1 + \frac{1}{COP}\right)$ |
| Cooling Tower | $Q_q \times \left(1 + \frac{1}{\mu}\right)$ |

The function *verifyEnergyFlow* is recursively executed (line 18) for each output node $o$, which corresponds to a depth-first approach that traverses the graph. This approach visits every node in the graph and checks every edge. Therefore, Depth-first approach complexity is O(N + A). This results in O($N^2$) complexity for the Algorithm 2. The Algorithm 1 complexity is then O($N^3$). Additionally, the algorithm adopts specific equations for computing the output energy of different devices as shown in Table 3.1.

In Table 3.1, $\eta$ represents the equipment's efficiency according to the second law of thermodynamic; $Q_{IN}$ is the input thermal energy; $Q_F$ is the thermal energy of the CRACs fluid; $COP$ is the coefficient of performance; $Q_q$ corresponds to the thermal load that flows to the cooling tower; and $\mu = \frac{MaximumCoolingPower}{MaximumPowerConsumption}$. Since the physical behavior of each device is not the focus of this work, the reader should refer to [49][50].

**Quantifying Cost**

For the purpose of this study, data center cost is represented by acquisition and operational costs. The acquisition cost (AC) corresponds to the financial resources required to purchase the data center infrastructure (according to the retail prices of equipment). To compute the operational cost (OC), this work only recognizes the costs corresponding to

the energy consumed, as illustrated in Equation 3.1. Other factors however, could also be included.

$$OC = P_{input} \times T \times C_{energy} \times (A + \alpha(1 - A)), \tag{3.1}$$

where $P_{input}$ is the electrical power consumed; $T$ is the assumed period; $C_{energy}$ corresponds to the energy price; $A$ is the availability; and $\alpha$ is the factor adopted to represent the amount of energy that continues to be consumed after a component has failed.

In order to calculate OC, designers need to first define the energy demanded by the IT system during the period $T$. However, since no electrical component is 100% energy efficient, this figure will differ from the actual energy consumed. Therefore the true figure ($P_{input}$) must first be calculated. A depth-first approach that traverses the EFM is adopted to compute the OC.

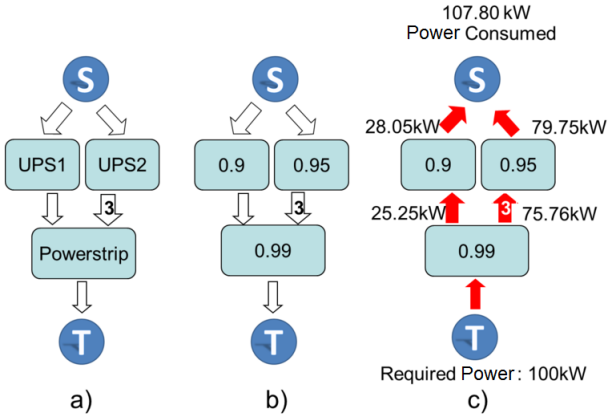


**Figure 3.3:** a) System example; b) Efficiency values; c) Computing the energy consumed.

Figure 3.3 (a) shows an example of a power system, composed of two UPSs and one powerstrip, which provides 100kW of power to IT devices. In the example, the weights assigned to the edges that connect the internal nodes UPS1 and UPS2 to the Powerstrip node are, respectively, one and three. In this case, UPS2 component provides three times more power than UPS1. The efficiencies of UPS1, UPS2 and Powerstrip are 0.9, 0.95 and 0.99 (Figure 3.3 (b)), respectively. To provide the demanded power of 100kW, the power consumed is 107.80kW as depicted in Figure 3.3 (c).

Algorithm 3 is adopted to compute the data center cost. A depth-first approach is performed to traverse the graph. This approach visits every node in the graph and checks every edge. Therefore, depth-first approach complexity is O(N + A). This results in O($N^2$) complexity for the Algorithm 4 and O($N^3$) for the Algorithm 3.

Similarly to the *EnergyFlow* Algorithm (Algorithm 1), Algorithm 3 begins by identifying (line 1) the model under analysis as cooling or power. In the example depicted in Figure 3.3, the set of source nodes ($S_s$) corresponds to $N_t$ (power system). The algorithm

proceeds by performing calls to the function *powerConsumed*, which computes the power consumed by each node $n \in S_s$. A local variable $ec$ is adopted to hold the power returned by that function (line 8). Afterwards, the algorithm computes the operational cost $oc$ (line 10) and the acquisition cost $ac$ (line 11). Finally, the algorithm finishes and the data center cost is returned (line 12).

In the function *powerConsumed*, $n$ corresponds to the target node ($n = T$) which is the first node to be visited in $G$. Considering the node $T$, the set of income nodes is composed of Powerstrip internal node ($O_n = \{Powerstrip\}$). The algorithm continues by checking if the set $O_n$ is empty (line 6). Next, the sum of its income edge weights ($ws$) is calculated (line 9).

**Algorithm 3** $Cost(G, m, t, p, a)$

1: **if** $(m = cooling)$ **then**
2: $S_s := N_s$;
3: **else**
4: $S_s := N_t$;
5: **end if**
6: $ec := 0$;
7: **for** $n \in S_s$ **do**
8: $ec := ec + powerConsumed(G, f_d(n), n, m)$ :
9: **end for**
10: $oc := result \times t \times p \times a$;
11: $ac := \Sigma_{i \in N_i}\ f_p(i)$;
12: **return** $ac + oc$;

In addition, a vector ($o_e$), which has the size of the set $O_n$, is adopted to associate to each node $o \in O_n$ the value that represents the power that flows through the component. For each node $o \in O_n$, the function *powerConsumed* updates the vector $o_e$ (line 11). In the function *power Consumed*, $f_\eta$ is a function that relates each component with the efficiency (supplied by the designer); $d_n$ represents the demanded power transferred to (e.g, electricity) or extracted (e.g., heat) from the node $n$; $w(n, o)/ws$ is the ratio of the weight of the edge that connects the node $n$ to the the node $o$ by the sum of weights of all output nodes of $n$. Thus, $o_e + ((d_n \times (w(n, o)/ws))/f\eta(i))$ represents the power that flows from the component represented by the node $o$ to the component represented by the node $n$, taking into account the efficiency of each component. Afterwards, the function *powerConsumed* is recursively called (line 15) to compute the power consumed of the system.

**Algorithm 4** $powerConsumed(G, d_n, n, m)$

**if** (m = cooling) **then**
  $O_n := \{o|(n,o) \in A; n \in N; o \in N_i \cup N_t\}$;
**else**
  $O_n := \{o|(n,o) \in A; n \in N; o \in N_i \cup N_s\}$;
**end if**
**if** $(O_n = \emptyset)$ **then**
  **return** $o_e$;
**end if**
$ws = \Sigma_{o \in O_n}\ w(n,o)$;
**for** $o \in O_n$ **do**
  $o_e = o_e + \frac{d_n \times \frac{w(n,o)}{ws}}{f\eta(o)}$;
**end for**
$energySum := 0$;
**for** $o \in O_n$ **do**
  $energySum := energySum + powerConsumed(G,\ \frac{d_n \times \frac{w(n,o)}{ws}}{f\eta(o)},\ o, m)$;
**end for**
**return** $energySum$;

**Quantifying Operational Exergy Consumption**
The operational exergy destroyed (or consumption) can be understood as the fraction of heat dissipated by each item of equipment that cannot be theoretically converted into useful work. The following equation represents the system operational exergy consumption.

$$Ex_{op} = \sum_{i=1}^{n} \dot{Ex}_{op_i} \times T \times (A + \alpha(1 - A)), \tag{3.2}$$

where $\dot{Ex}_{op_i}$ is the rate of the exergy destroyed by each device (Table 3.2); $T$ is the period of analysis; $A$ is the system availability; and $\alpha$ is the factor adopted to represent the amount of energy that continue to be consumed when a component has failed.

Specific equations are adopted to compute the operational exergy consumption by each component due to the fact that each device may consume or destroy exergy in different ways. Table 3.2 presents those equations.

**Table 3.2:** Operational exergy Equations of different devices.

| **Device** | **Operational Exergy Equation** |
|---|---|
| Electrical | $P_{in} \times (1 - \eta)$ |
| Diesel Generator | $P_{in} \times \left(\frac{\varphi}{\eta} - 1\right)$ |
| CRAC | $Q_{in} \times \left(1 - \frac{T_{cold}}{T_{room}} + \frac{1}{\mu}\right)$ |
| Chiller | $Q_{in} \times \left(\frac{1}{COP} - \frac{T_{tower} - T_{chilled}}{T_{chilled}}\right)$ |
| Cooling Tower | $Q_{in} \times \left(1 - \frac{T_{amb}}{T_{warm}} + \frac{1}{\mu}\right)$ |

In this table, $\eta$ is the delivery efficiency; $P_{in}$ is the total input power of the electrical device; $\varphi$ is the exergy correction; $Q_{in}$ is the total thermal input of the device; $T_{room}$ is the data center room temperature; $T_{cold}$ corresponds to the CRAC's cold water temperature; $\mu$ is the ratio of the maximum cooling power by the maximum power consumption; $COP$ is the coefficient of performance; $T_{tower}$ is the water temperature that goes to the cooling tower; $T_{chilled}$ corresponds to the chilled water temperature; $T_{amb}$ is the ambient temperature; $T_{warm}$ corresponds to the warm water from chillers. All the temperatures must be in Kelvin. Since the physical behavior of each device is not the focus of this work, the reader should refer to [49][50].

The experiments present in Chapter 5 consider the following values: for a diesel generator, the exergy correction $\varphi$ is 1.06 and the energetic efficiency $\eta$=25%; for CRAC devices, the $T_{room}$ adopted is 293.15 K and $T_{cold}$ is 280.15 K; for chillers, the adopted COP is 4, the considered $T_{tower}$ is 308.15 K and the adopted $T_{chilled}$ is 280.15 K; for cooling towers, the considered $T_{amb}$ is 298.15 K and the $T_{warm}$ is 308.15 K.

**Algorithm 5** $computeExergyOperational(G, m, T, A)$

1: **if** $(m = cooling)$ **then**
2: $S_s := N_s$;
3: **else**
4: $S_s := N_t$;
5: **end if**
6: $ex := 0$;
7: **for** $n \in S_s$ **do**
8: $exa := 0$;
9: $ex := ex + exergyOperational(G, f_d(n), n, m)$ :
10: **end for**
11: **return** $ex \times T \times A$;

**Algorithm 6** $exergyOperational(G, d_n, n, m)$

1: **if** $(m = cooling)$ **then**
2: $O_n := \{o|(n, o) \in A; n \in N_s \cup N_i; o \in N_i\}$;
3: **else**
4: $O_n := \{o|(n, o) \in A; n \in N_t \cup N_i; o \in N_i\}$;
5: **end if**
6: **if** $(O_n = \emptyset)$ **then**
7: **return** $exa$;
8: **end if**
9: $ws := \Sigma_{o \in O_n}\ w(n, o)$;
10: **for** $o \in O_n$ **do**
11: $exa := exa + exergy(o, d_n)$;
12: **end for**
13: **for** $o \in O_n$ **do**
14: $exergyOperational(G,\ out(o,\ d_n \times \frac{w(n,o)}{ws}),\ o, m)$;
15: **end for**

Algorithms 5 computes the system operational exergy destroyed by traversing the graph $G$. The algorithm starts by checking the type of model that is analyzed (assume $m = \{cooling, power\}$). Depending on the model under analysis, the set of source nodes ($S_s$) may be $N_s$ or $N_t$ (lines 1-5). A local variable $ex$ is adopted to hold the exergy destroyed returned by the function *exergyOperational* (line 9). The function *exergyOperational* returns the operational exergy of the system where $f_d(n)$ is the demanded energy specified by the designer. When the algorithm finishes, it provides the system operational exergy (sum of the operational exergies of each device multiplied by the period ($T$) and availability ($A$).

The function *exergyOperational* starts checking the type of model under analysis to determine the set $O_n$. For cooling models, $O_n$ represents the set of output nodes of the node $n$ (or the income nodes when considering power models). A global variable ($exa$) is adopted to compute the sum of operational exergies of each component (line 11). Function $exergy(o, d_n)$ is adopted to compute the exergy according to the Table 3.2. Additionally, the complexity for both Algorithm 5 and 6 are O($N^3$) and O($N^2$), respectively.

## 3.2 DEPENDABILITY MODELS

This section presents the basic SPN building blocks for quantifying system dependability and the approach utilized for combining them. The proposed models are generic enough to represent a wide variety of complex redundancy mechanisms and scenarios, and, are applicable to any equipment in a data center. Regarding SPN basic definitions and semantic, the reader is referred to [38] for a thorough explanation. This section also summarizes the well established RBD dependability closed form solutions. Firstly, the adopted modeling strategy is presented. Next, RBD and SPN buiding blocks are explained.

### 3.2.1 Modeling Strategy

A hybrid modeling strategy combining combinatorial and state-based models is employed to represent system dependability features. This strategy recognizes the advantages of both reliability block diagrams (combinatorial model) and stochastic Petri nets (state-based model). Such a hierarchical approach mitigates the complexity of representing large systems. Depending on the particular complexity and size, the system may be either represented by one model or split into smaller models, corresponding to the system parts. RBDs are used for modeling systems (or subsystems) when dynamic dependencies are absent [51].

Subsystems represented by RBD may include dynamic dependencies between them, SPN is then employed to compose the final model which corresponds to the entire system. Therefore, SPN is used for subsystems that show dynamic dependencies within

themselves. The subsystem results may then be used as parameters for the higher level RBD or SPN model. It should be stressed that each subsystem may be represented by a different formalism. Bearing in mind the characteristics of the system parts, subsystems may be modeled using either SPN or RBD (Continuous Time Markov Chain - CTMC and Fault Trees - FT could also be adopted). The results are then combined employing either SPN or RBD depending on the interactions between the system parts.

### 3.2.2 RBD Models

The Reliability Block Diagram (RBD) [23] is a combinatorial model that was initially proposed as a technique for calculating reliability of systems by using intuitive block diagrams. Such a technique has also been extended to calculate other dependability metrics, such as availability and maintainability [9]. Figure 3.4 depicts two examples, in which independent blocks are arranged through series (Figure 3.4(a)) and parallel (Figure 3.4(b)) compositions.

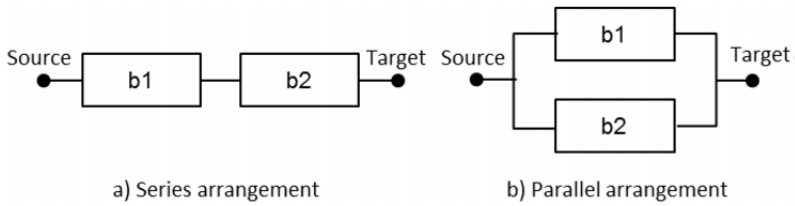


**Figure 3.4:** Reliability Block Diagram

In the series arrangement, if a single component fails, the whole system is no longer operational. Assuming a system with $n$ independent components, the reliability (instantaneous availability or steady state availability) is obtained by

$$P_s = \prod_{i=1}^{n} P_i, \tag{3·3}$$

where $P_i$ is the reliability $R_i(t)$ (instantaneous availability $(A_i(t))$ or steady state availability $(A_i)$) of block $b_i$.

For a parallel arrangement (see Figure 3.4b), if a single component is operational, the whole system is also operational. Assuming a system with $n$ independent components, the reliability (instantaneous availability or steady state availability) is obtained by

$$P_p = 1 - \prod_{i=1}^{n} (1 - P_i), \tag{3·4}$$

where $P_i$ is the reliability $R_i(t)$ (instantaneous availability $(A_i(t))$ or steady state availability $(A_i)$) of block $b_i$.

A k-out-of-n system functions if and only if $k$ or more of its $n$ components are functioning. Let $p$ be the success probability of each of those blocks. The system success probability (reliability or availability) is depicted by:

$$\Sigma_{i=k}^{n} \binom{n}{i} p^k (1-p)^{n-k} \tag{3.5}$$

For other examples and closed-form equations, the reader should refer to [9].

### 3.2.3 SPN Models

Petri nets (PN) [52] are a family of formalisms very well suited for modeling several types of system, since concurrency, synchronization, communication mechanisms as well as deterministic and probabilistic delays are naturally represented. In general, Petri nets are a bipartite directed graph, in which places (represented by circles) denote local states and transitions (depicted as rectangles) represent actions. Arcs (directed edges) connect places to transitions and vice-versa.

Petri nets were extended by associating time with the firing of transitions, resulting in timed Petri nets [53]. The firing time of a transition is the time that must elapse from the instant that the transition is enabled until the instant it actually fires in isolation. Stochastic Petri nets (SPN) [54] are formally defined as a special case of timed Petri net where the firing times are considered to be random variables with exponential distributions.

This work adopts Stochastic Petri Net, which allows the association of probabilistic delays to transitions using the exponential distribution, for conducting dependability analysis of data center architectures. In SPN, the underlying stochastic process is a homogeneous CTMC with state space isomorphic to the reachability graph of the PN [8]. Besides that, SPN allows the adoption of simulation techniques for obtaining dependability metrics, as an alternative to the Markov chain generation.

In SPN, transitions are allowed to be either timed (exponentially distributed firing time, drawn as rectangular boxes) or immediate (zero firing time, represented by thin black bars). Immediate transitions always have priority over timed transitions. In addition, if both timed and immediate transitions are enabled in a marking then timed transitions are treated as if they are not enabled. SPN also introduced the concept of inhibitor arc (represented by a small hollow circle at the end of the arc) which connects a place to a transition. A transition with an inhibitor arc can not fire if the input place of the inhibitor arc contains more tokens than the multiplicity of the arc.

Next sections describe the SPN Simple component followed by the cold standby model which are proposed for estimating the dependability (e.g., availability and reliability) of data center components. After that, we present the model for common mode failure

which is adopted when events are not statically independent. Finally, the approach for combining the components in series and parallel arrangement is described.

**3.2.3.1 Simple Component** The simple component is a representative building block characterized by the absence of redundancy, that is, the component might be in two states, either functioning or failed. In order to compute its availability, the $TTF$ and $TTR$ should be represented. If both $TTF$ and $TTR$ are exponentially distributed, $MTTF$ and $MTTR$ are the only parameters needed for computing its availability. The reliability is also straightforwardly computed by $R(t) = \exp^{\frac{-t}{MTTF}}$, considering that the repairing activity is not allowed.

The respective SPN model of the "simple component" is shown in Figure 3.5. This component has two parameters (not depicted in the figure), namely $X\_MTTF$ and $X\_MTTR$, which represent delays associated to transitions $X\_Failure$ and $X\_Repair$, respectively. Both transitions are exponentially distributed ($exp$) and adopt single server ($ss$) concurrency policy (or infinity server, depending on the application). Label "X" is instantiated according to the component name, like $UPS\_FAILURE$ and $UPS\_REPAIR$. When the delay associated with transitions $X\_Repair$ and $X\_Failure$ are marking dependent, additional constants, such as $X\_MTTR\_i$ or $X\_MTTF\_i$, are considered. Unless stated, the reader should assume the delays of both transitions being not marking dependent as well as having single server concurrency policy.

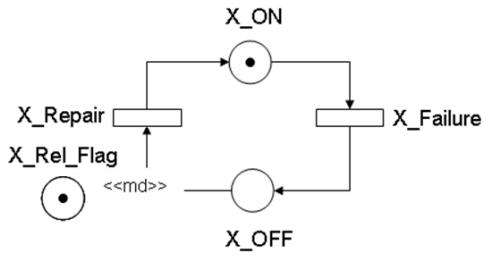


**Figure 3.5:** Simple component model

Table 3.3 depicts the attributes related to transitions of the simple component model.

Places $X\_ON$ and $X\_OFF$ are the model component's activity and inactivity states, respectively. The simple component also includes an arc from $X\_OFF$ to X_Repair with multiplicity depending on place marking. The multiplicity is defined through the expression IF($\#X\_Rel\_Flag = 1$) 2 ELSE 1, where place $X\_Rel\_Flag$ models the evaluation of reliability/availability. Hence, if condition $\#X\_Rel\_Flag = 1$ is false (no token in place $X\_Rel\_Flag$), then the evaluation refers to availability. Otherwise, the evaluation concerns reliability. This approach enables us to parameterize the model and it grants the system evaluation considering repairing (i.e., availability) or not (i.e., reliability).

Table 3.4 shows the conditions that represent the operational and failure states. A component is operational if the number of tokens (#) in place $X\_ON$ is greater than 0 and in a failure state, otherwise. Hence, if $\#X\_Rel\_Flag = 1$, $P\{\#X\_ON > 0\}$

**Table 3.3:** Simple Component transition attributes.

| Transition | Type | Delay | Markup | Concurrency |
|---|---|---|---|---|
| X_Failure | exp | X_MTTF(_i) | constant/dependent | SS |
| X_Repair | exp | X_MTTR(_i) | constant/dependent | SS |

means the component's availability (steady-state evaluation). If $\#X\_Rel\_Flag = 0$, then $P\{\#X\_OFF > 0\}$ allows computing the component's reliability, if transient evaluation is carried out and the initial marking is $\#X\_ON = 1$ and $\#X\_OFF = 0$. It should be highlighted that the presented model is bounded (has only two states), reversible and live [52]. The last two properties only hold if $\#X\_Rel\_Flag = 0$.

**Table 3.4:** Conditions representing states of the simple component.

| State | Condition |
|---|---|
| Operational | #X_ON> 0 |
| Failure | #X_ON= 0 |

**3.2.3.2 Simple Component Models considering First and Second Moment Matching** In this section, an extension of the previous simple component model is presented for modeling systems that consider other distributions (e.g., Erlang). Therefore, a variation of the initial simple component presented in Section 3.2.3.1 considers the refinement of transitions in order to take into account first and second moments. In this particular model, transition $X\_Failure$ of the model depicted in Figure 3.5 has been refined for representing an Erlangian distribution (Erlangian s-transition). The parameter $X\_MTTF$ ($\mu$) associated to this event has not an exponential distribution and the ratio ($X\_MTTF$/standard deviation - $\frac{\mu}{\sigma} \in \mathbb{N}, \frac{\mu}{\sigma} \neq 1, (\mu, \sigma \neq 0)$) is an integer not equal to 1. According to the moment matching method presented, this transition should be refined into as an Erlang s-transition (see Figure 3.6).

Timed transitions $X\_Failure\_Erlang1$ and $X\_Failure\_ErlangN$ represent the exponential phases of Erlang distribution. The arcs connecting place $X\_ON$ to time transition $X\_Failure\_Erlang1$ and place $p_{10}$ to immediate transition $t_{10}$ has multiplicity depending on marking, and their arc expression are $(\gamma - 1) = [(\frac{\mu}{\sigma})^2 - 1]$ and $\gamma = (\frac{\mu}{\sigma})^2$, respectively (see Table 3.5). The delay assigned to transitions $X\_Failure\_Erlang1$ and $X\_Failure\_ErlangN$ is $\mu_1 = \mu/\gamma$. It is worth observing that control elements have been adopted in this model, so that one token is kept stored in place $X\_ON$ during all Erlangian phases' firing; and then being only removed when transition $t_{10}$ fires, ending all phase firing and completing the Erlang s-transition execution. For depicting such behavior, one single arc from transition $X\_Failure\_Erlang1$ to place $X\_ON$ and another from that place to transition $t_{10}$ had to be assigned as well as one inhibitor arc from the output place

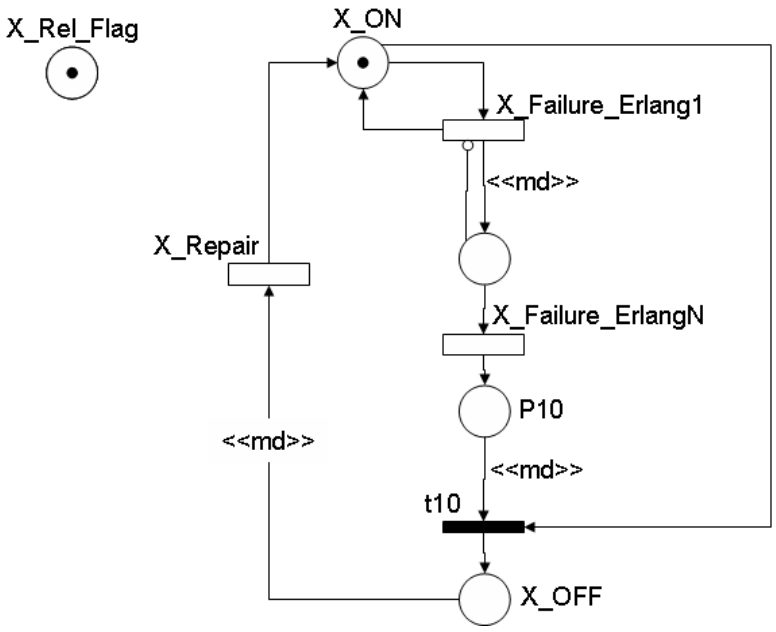


**Figure 3.6:** Simple Component with failure represented by Erlang distribution.

of transition $X\_Failure\_Erlang1$ back to itself. In this case, the inhibitor arc avoids the first phase transition ($X\_Failure\_Erlang1$) unbounded firing of Erlangian model. This transition might only be enabled again when no token remains in output place of transition $X\_Failure\_ErlangN$. One might argue that after removing all tokens from output place of transition $X\_Failure\_ErlangN$, transition $X\_Failure\_Erlang1$ could be immediately fired again. However, one should bear in mind that immediate transitions have firing priority over timed ones, so the transition $X\_Failure\_Erlang1$ could never be fired again before transition $t_{10}$ firing.

**Table 3.5:** Simple Component with failure represented by Erlang distribution - Transitions' attributes

| Transition | Type | Delay | Weight | Concurrency |
|---|---|---|---|---|
| X_Failure_Erlang1 | exp | $\frac{X\_MTTF}{\gamma}$ | - | SS |
| X_Failure_ErlangN | exp | $\frac{X\_MTTF}{\gamma}$ | - | SS |
| X_Repair | exp | MTTR | - | SS |
| $t_{10}$ | im | - | 1 | - |

This component also includes an arc from place $X\_OFF$ to transition $X\_Repair$ with multiplicity depending on a place marking. The arc's multiplicity is defined by the expression IF($\#X\_Rel\_Flag1 = 1$) 0 ELSE 1, where place $X\_Rel\_Flag$ allows the evaluation of reliability or availability metrics as depicted in previous sections. Table 3.6

presents the state conditions that represent the operational and the failure component's states.

**Table 3.6:** States conditions of simple component with failure represented by Erlang distribution.

| State | Condition |
|---|---|
| Operational | #X_ON>0 |
| Failure | #X_OFF>0 |

Many other variations of this model have been defined, including further $X\_Failure$ (see Figure 3.6) transition refinements (hyperexponential and hypoexponential subnets), refinements of transition $X\_Repair$ (see Figure 3.6) and the inclusion of failure coverage and imperfect repairing.

The RBD availability closed form expression for a "simple component" with $TTF$ distributed according to an Erlang distribution and a $TTR$ exponentially distributed is $A = \frac{\gamma \times \mu}{(\gamma \times \mu) + MTTR}$.

**3.2.3.3 Cold standby** A cold standby redundant system includes a non-active spare module that waits to be activated when the main active module fails. Figure 3.7 is the SPN model of such a system, which includes the places, $X\_ON$, $X\_OFF$, $X\_Spare1\_ON$, $X\_Spare1\_OFF$ which respectively represent the operational and failure states of both the main and spare modules. Initially the spare module (Spare1) is deactivated, hence at the start no tokens are stored in places $X\_Spare1\_ON$ and $X\_Spare1\ \_OFF$. When the main module fails, transition $X\_Activate\_Spare1$ is fired to activate the spare module.

Table 3.7 details the attributes of each transition. In this table, MTActivate corresponds to the mean time to activate the spare module. The availability may be computed by the probability $P\{\#X\_ON = 1 \text{ OR } \#X\_Spare1\_ON = 1\}$.

**Table 3.7:** Cold standby model - Transition attributes.

| Transition | Priority | Delay or Weight |
|---|---|---|
| X_Failure | - | X_MTTF |
| X_Repair | - | X_MTTR |
| X_Activate_Spare1 | - | MTActivate |
| X_Failure_Spare1 | - | X_MTTF_Spare1 |
| X_Repair_Spare1 | - | X_MTTR_Spare1 |
| X_Desactivate_Spare1 | 1 | 1 |

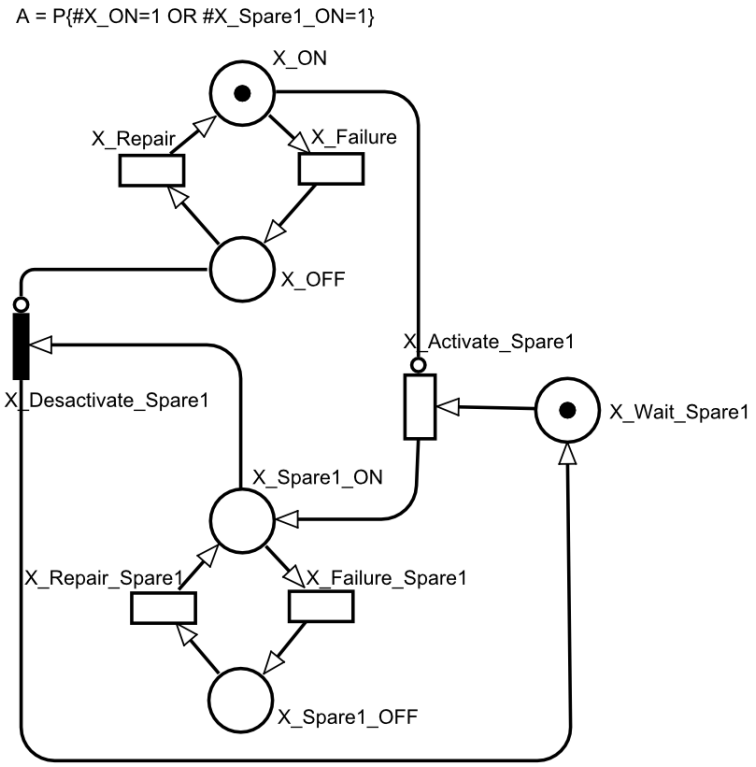


**Figure 3.7:** Cold standby model.

**3.2.3.4 Common mode failure** A prominent feature of SPN regards the representation of common mode failure (CMF), which usually takes place when events are not statically independent. In this case, one component failure may affect other system parts, for instance, impacting the time to failure. Representing common mode failure through RBD is not trivial, because the system structure changes over time. In the context of SPN, the combination of simple component blocks can model such failure mode by associating marking dependent delays to transitions of related components.

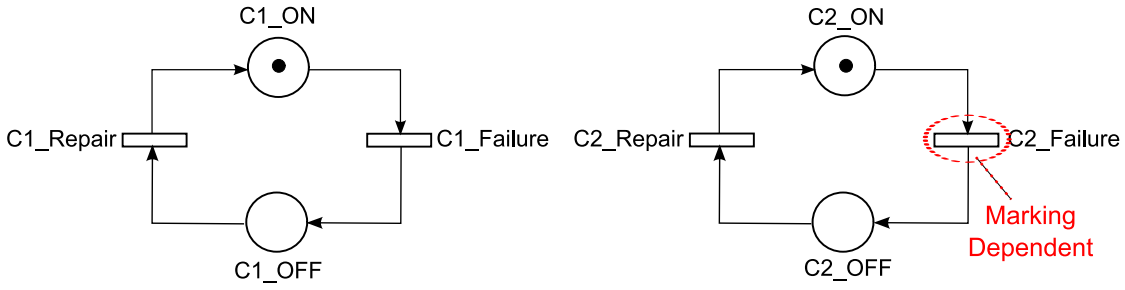


**Figure 3.8:** Common mode failure example

Figure 3.8 depicts an example composed of two components, in which the failure of C1 impacts the C2's MTTF. Assuming a failure rate $\lambda$, C1 may affect C2 in the

following way: C2_MTTF = $1/(\lambda \times (\#C1\_ON + 1))$, where #C1_ON represents the number of tokens in place C1_ON. Additional components may be taken into account, and the respective MTTRs can be adjusted to consider a marking dependent failure delay, similar to component C2.

**3.2.3.5 Aggregation Component** The proposed models also takes into account an aggregation component (Figure 3.9), which represents the activity ($X\_ON$) and inactivity ($X\_OFF$) states of a subsystem or system (e.g., group of simple component models). Transitions $X\_Failure$ and $X\_Repair$ have one parameter associated (not shown in the figure), namely, $X\_Fail\_GE$ and $X\_Repair\_GE$, respectively, which are guard expressions. More specifically, a transition (e.g., X_Failure) can only fire if the respective guard expression (e.g., $X\_Fail\_GE$) is true. Such transitions are immediate transitions (see Table 3.8).

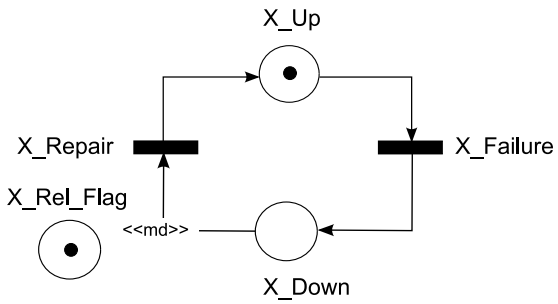


**Figure 3.9:** Simple logical component model

In the same way as simple component model, if condition #X_Rel_Flag=1 is false (no token in place X_Rel_Flag), then the evaluation refers to availability. Otherwise, the evaluation concerns reliability.

**Table 3.8:** Aggregation Component attributes.

| Transition | Type | Guard Expression |
|---|---|---|
| X_Failure | im | X_Fail_GE |
| X_Repair | im | X_Repair_GE |

**3.2.3.6 Model composition** The model composition is carried out using building block models and an aggregation component (Figure 3.9) , which is adopted to represent the relation among the components using appropriate guard expressions on the immediate transitions.

Without loss of generality, the subsystems are combined by series, parallel, (non) series-parallel, and hierarchical compositions. RBDs models are similarly combined and

well-established closed-form equations are available for those compositions. As an example, series composition is presented as follows considering SPN building blocks.

**Series Composition.** Two components in a series composition means that if just one of them fails then the whole composition also fails, i.e., the system is operational if both components are operational. Figure 3.10 presents an equipment named Device that is composed of two basic non-redundant components, $C1$ and $C2$, arranged serially. The aggregation component, which is represented by the places *Device_Up* and *Device_Down*, may be adopted to represent operational and failure states, respectively.

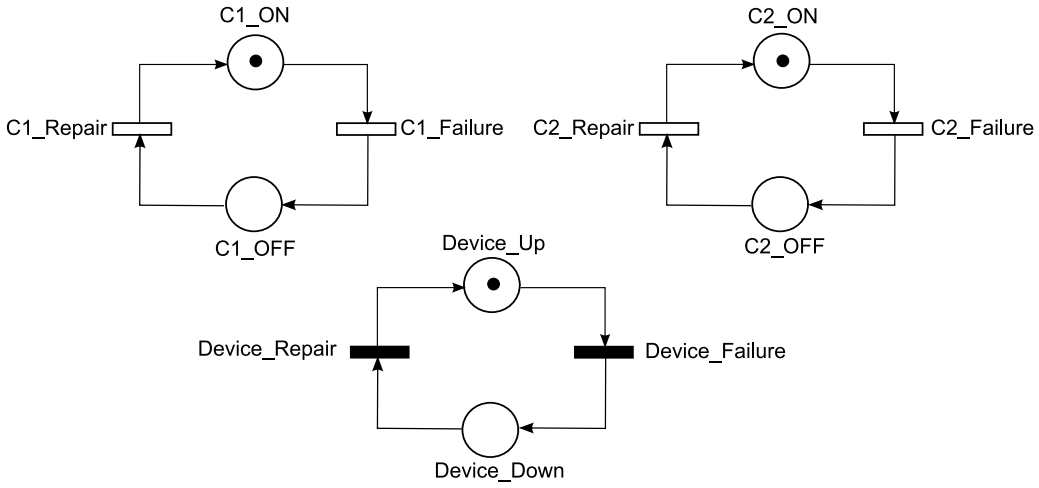


**Figure 3.10:** Series composition of two basic non- redundant components

To completely specify the aggregation component model, enabling functions are defined through guarded expressions that are associated to the immediate transitions *Device_Repair* and *Device_Failure*. The guard expression associated to the transition *Device_Failure* represents the condition in which the system is in the failure state. Likewise, the guard expression associated to the transition *Device_Repair* defines the logic expression that describes the conditions referring to the operational state. Table 3.9 presents the guarded expressions of the aggregation model depicted in Figure 3.10. In the first row, the transition Failure is enabled when at least $C1$ or $C2$ has failed.

**Table 3.9:** Transitions' guarded expressions of series composition

| Transition | Guard Expression |
|---|---|
| Device_Failure | (#C1_ON=0) OR (#C2_ON=0) |
| Device_Repair | negation of previous guard expression |

The system availability is then computed by the probability expression $P\{\#Device\_Up > 0\}$, where "#" represents the number of tokens in the place *Device_Up*. Although the use of aggregation component is very useful when modeling, its adoption increases the

number of states of the associated CTMC. Therefore, its use is indicated for modeling system and testing the behavior through the token game functionality that is described in the Chapter 4. Exact the same availability results can be obtained (without the aggregation component) by the evaluation of the probability expression $P\{(\#C1\_ON = 1)AND(\#C2\_ON = 1)\}$ which represents the series characterization. Additionally, such an approach can be adopted to combine N components represented by either a single aggregation component model or probability expression.

**Parallel Composition.** The simplest example of redundancy could be achieved by combining two devices in a parallel configuration. This composition only fails if both components have failed. Assuming the SPN model depicted in Figure 3.10, the the availability of the parallel composition of C1 and C2 is obtained by the probability $P\{(\#C1\_ON = 1)OR(\#C2\_ON = 1)\}$.

**Series-Parallel Composition.** The rules defined to serial and parallel compositions may be combined to define more complex arrangements. For instance, considering a system composed of three components C1, C2 and C3. Assuming that the system is operational if C1 is working and at lest one of C2 or C3 is operational. In this case, the component C1 is in series to the parallel composition of C2 and C3. The SPN model of this system is shown in Figure 3.11. The availability is computed by the probability expression $P\{(\#C1\_ON = 1)AND((\#C2\_ON = 1)OR(\#C3\_ON = 1))\}$.

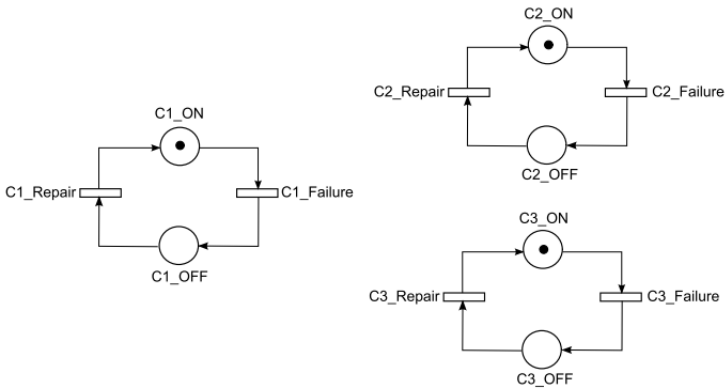


**Figure 3.11:** SPN model of Series-Parallel Composition

## 3.3 OPTIMIZATION

Many problems found in practice are either computationally intractable by their nature or sufficiently large that may preclude the use of exact algorithms [45]. In this work we adopted an algorithm based on the GRASP to optimize the dependability, sustainability and cost issues estimated through the EFM and dependability models. GRASP is a metaheuristic applied for combinatorial optimization problems [46]. GRASP is able to achieve values close to the optimal results with error around 2% for solving covering

problems[47]. In a cover problem, we deal with a given set of objects $X$ and a set $F$ of subsets of $X$. The goal is to pick some subset of $F$ that satisfies some constraints and optimizes some objective value. For example, one might want to pick the smallest subset of $F$ that contains (covers) all objects of $X$. Covering problems are minimization problems and usually linear programs. As further detailed in this section, the optimization model in this thesis focuses in minimizing the downtime, cost and energy consumption of data center architectures and a GRASP based algorithm was implemented to allow us to perform such optimization technique.

The algorithm consists basically of two phases: construction and local search. The construction phase builds a feasible solution in each interaction of the algorithm, whose neighborhood is investigated until a local minimum is found during the local search phase. The best solution overall iterations is returned as the result.

Similarly to other optimization methods, the adopted algorithm starts from an initial model and then perform local searches to improve the quality of the first solution obtained. In order to accomplish this, some greedy randomized procedures are adopted and then the local search is performed from the constructed model/solution. This two-phase process is repeated until the stopping condition is satisfied.

The Optimization algorithm has seven parameters. These parameters are: the graph $G$ (EFM), *maxItr* that represents the maximum number of iterations executed by the Optimization method, *solution* which is the variable that holds the computed solutions, $\beta$ is a value between the range [0,1] which represents the greediness degree, *nTriesNeighbor* is the number of tries to find other solutions by the neighbors, *sizePollSol* is the size of the poll of solutions, CL is the candidate list. The following lines present the algorithm, its functions, and briefly describe the method.

The Optimization method (Algorithm 7) begins by setting the *result* variable to an infinity number (line 1). The algorithm proceeds by calling (the number of calls is determined by the *maxItr* variable) the *Construction* and *LocalSearch* functions (lines 3 and 4). The *Construction* function constructs a solution through the candidate list of components. The *LocalSearch* function generates a new solution created by the neighborhood of the previous solution generated by the *Construction* function. Afterwards, the algorithm checks if the new solution is better (in terms of cost) than the solution held by the *result* variable (line 5). The *result* variable holds the new solution (line 6) when that check returns TRUE. Otherwise, the algorithm proceeds to another iteration until reach the maximum number of iterations to return the best solution obtained by the method. The following lines present both *Construction* and *LocalSearch* functions in more details.

**Construction**

The construction phase, which is based on a greedy heuristic, creates feasible solutions iteratively [55]. The *Construction* function (Algorithm 8) begins by emptying the *solution* variable (line 1). Afterwards, a restricted candidate list (RCL) is created for each node element of the graph $G$ according to the $\beta$ greediness degree (line 3). Let $\beta \in [0, 1]$, in which $\beta = 1$ produces a random construction and $\beta = 0$ corresponds to a pure greedy

**Algorithm 7** *Optimization(G, maxItr, solution, $\beta$, nTriesNeighbor, sizePollSol, CL)*

1: $result := \infty$;
2: **for** $i = 1$ to maxItr **do**
3: $solution := Construction(G, \beta, CL)$;
4: $solution := LocalSearch(solution, nTriesNeighbor, sizePoll- Sol)$;
5: **if** $(computeCost(solution) < result)$ **then**
6: $result := computeCost(solution)$;
7: **end if**
8: **end for**
9: **return** *result*;

algorithm. For instance, $\beta = 0.2$ means that the 20% of the better solutions according to the computed cost are selected and returned to the RCL. A node is then randomly selected from the RCL and added to the *solution* vector (line 4). Next, the LC is updated (line 5). This process is repeated for all node elements that compose the graph G.

**Algorithm 8** *Construction(G, $\beta$, CL)*

1: $solution := \oslash$;
2: **for** (node = 0 to size(G)) **do**
3: $RCL := getBestCandidates(CL, node, \beta)$;
4: $solution[node] := getRandomNode(RCL)$;
5: $CL := removeNode(CL, node)$;
6: **end for**
7: **return** *solution*;

**Local Search**

The solutions created by the *Construction* function are not guaranteed to be locally optimal [56]. A solution is locally optimal if there is no better solution in its neighborhood. The *LocalSearch* function improves the previous obtained solution by the successive replacement of the current solution by a better one from the neighborhood of the current solution. The success of the *LocalSerach* function depends on the starting solution received as parameter as well as on the suitable choice of a neighborhood structure [56].

The *LocalSearch* function (Algorithm 9) starts by initializing the neighborhood solution (*neighborSol*), the poll of solutions (*pollSol*), cont1 and cont2 to *null* (lines 1 and 2). The algorithm proceeds by computing a new solution from the current solution (line

4). This new solution is created after randomically changing one node of the current solution by another element obtained from the RCL. If the neighborhood solution obtained is feasible as well as its cost is smaller than the previous solution (line 5), then the current position in the poll of solutions (*cont2*) receives this new solution (line 6). Next, the *cont2* is incremented (line 7) and the current solution (*solution*) is set as the new solution created. These steps are repeated until the poll of solutions is filled in or the number of tries to find new solutions in the neighborhood is reached. Afterwards, in case the poll is not empty, a solution is randomically selected from the poll of solutions (line 13) and it is returned to the Algorithm 7 (line 15).

**Algorithm 9** *LocalSearch(solution, nTriesNeighbor, sizePollSol)*

1: $neighborSol, pollSol := \oslash$;
2: $cont1, cont2 := 0$;
3: **repeat**
4: $neighborSol := getNeighborSol(solution)$;
5: **if** (neighborSol is feasible) && (computeCost (neighborSol) < computeCost (solution)) **then**
6: $pollSol[cont2] := neighborSol$;
7: $cont2++$;
8: $solution := neighborSol$;
9: **end if**
10: $cont1++$;
11: **until** $(cont2 = sizePollSol) or (cont1 = nTriesNeighbor)$
12: **if** $(pollSol \neq \oslash)$ **then**
13: $solution := getRandom(pollSol)$;
14: **end if**
15: **return** *solution*;

### 3.3.1 Optimization Model

In this section, we present the proposed optimization model. This optimization model as well as the algorithm are implemented with the support of the Mercury environment as detailed in the proposed methodology (Section 4.1).

The objective is to minimize the overall costs by increasing the availability and the energy efficiency while trading-off the acquisition costs subject to a given set of restric-

tions. The following lines describe the optimization model.

**Parameters**
$G$ is an EFM graph,
$D$ is a dependability model (RBD or SPN),
$dc$ is the downtime cost per hour of unavailability of the system under analysis.

**Decision variables**
$MTTF_i$ is the mean time to failure of the equipment $i$,
$\eta_i$ is the energetic efficiency of the equipment $i$,
$p_i$ is the retail price of the equipment $i$.

**Objectives**
The objective is quantified by the following equations.

$f_1$ : To minimize the cost related to the downtime of the system by increasing the availability.

$$f_1 = (1 - A) \times T \times dc, \tag{3.6}$$

where $A$ is the system availability, $T$ is the period in hours, and $dc$ is the mean downtime cost per hour.

$f_2$ : To minimize the operational cost by increasing the energetic system efficiency.

$$f_2 = P_{input} \times C_{energy} \times T \times (A + \alpha(1 - A)), \tag{3.7}$$

where $P_{input}$ is the electrical energy consumed, $T$ is the assumed period, $C_{energy}$ is the energy cost, $A$ is the availability, and $\alpha$ is the factor adopted to represent the amount of energy that continues to be consumed when a component has failed.

$f_3$ : To minimize the acquisition cost.

$$f_3 = \sum_{i=1}^{n} p_i. \tag{3.8}$$

**Objective Function**
We reduced the goals in the following objective function:

$$\boxed{\text{To minimize } f = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3,}$$

where $\lambda_{1..3}$ represents the weight given to the $f_{1..3}$ functions and the $\sum_{i=1}^{3} \lambda_i = 1$.

**Restrictions**
For all data center devices, the acquisition cost of each internal node of the EFM is greater than or equal to zero,

$\forall_{i=1}^{n} p_i \geq 0.$

The energetic efficiency ($\eta$) of all devices must be in the range [0,1].

$\forall_{i=1}^{n} 0 < \eta_i \leq 1.$

For all data center devices, the MTTF must be greater than or equal to zero.

$\forall_{i=1}^{n} MTTF_i \geq 0.$

The maximum power capacity of all devices must not be exceeded.

$\forall_{i=1}^{n} CCU_i \leq f_c,$

where $CCU$ is the current capacity used by each device $i$, and $f_c$ is a function that assigns each node with the respective maximum energy capacity.

## 3.4 SUMMARY

In this thesis models are proposed to conduct the integrated evaluation of sustainability, dependability and cost issues on data center architectures. This chapter presented those proposed models. Initially, we presented the energy flow model and the algorithms that traverse the graph to estimate the cost and sustainability impacts as well as that verifies the power restrictions of each component. Next, the modeling strategy that considers the advantages of both SPN and RBD models for dependability evaluation was described. Afterwards, the adopted SPN and RBD models were presented. Finally, the proposed GRASP based optimization algorithm that improves the evaluation results of EFM and dependability models through the selection of the appropriate components from the list of candidate devices was presented.

CHAPTER 4

# EVALUATION ENVIRONMENT

This chapter presents an overview of the conceived methodology for evaluating data center infrastructures taking into account dependability, cost and sustainability issues. In addition, this chapter briefly presents the evaluation environment that has been developed to provide support to the integrated evaluation of dependability, sustainability and cost issues on data center infrastructures. The evaluation environment is composed of ASTRO, Mercury and optimization module. This chapter starts by presenting the adopted methodology that takes the advantages of RBD, SPN and EFM models. After that, the ASTRO tool is shown. Next, the Mercury environment that supports for modeling proposes RBD, SPN, CTMC and EFM is presented. Finally, it shows the optimization module that supports two optimization algorithms: a GRASP based and the PLDA.

## 4.1 METHODOLOGY: AN OVERVIEW

The main goal of this thesis is to propose a set of models for the integrated quantification of sustainability impact, cost and dependability of data center power and cooling infrastructures. A hybrid modeling strategy which combines combinatorial and state-based models is adopted for representing the system dependability features. On one hand, RBDs allow to represent component networks and provide closed form equations. Nevertheless, such models face drawbacks for thoroughly handle failures and repairing dependencies that are often faced when representing dynamic redundancy methods. On the other hand, state-based methods can easily consider those dependencies, allowing to represent complex redundant mechanisms for instance. However, they suffer from the state-space explosion. Some of those formalisms allow both numerical analysis and stochastic simulation, and SPN is one of the most prominent models of such class. This work adopts closed form expressions for solving RBD, and analysis or simulation for computing the SPN results.

In addition, a model is proposed for verifying that the energy flow does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling equipment). In order to accomplish that, algorithms that traverse the EFM are proposed to perform the power verifications as well as to estimate data centers cost and sustainability.

An optimization technique is also adopted for improving the results obtained by RBD,

SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center power and cooling infrastructures. The optimal result can be achieved by evaluating all possible combinations from the list of candidate components. However such evaluation is quite time consuming. Therefore, the adopted optimization method represents an alternative approach that provides similar results in relation to the optimal ones in a reduced time.

An integrated evaluation environment has been developed to support both the proposed set of models and algorithms previously mentioned. This engine is composed of ASTRO, Mercury and an optimization module. ASTRO provides views (e.g. power and cooling data center views) from which data center designers are able to model systems without the need to know the formalisms that are adopted to compute the desired metrics. The models created in ASTRO can be converted to the fundamental models that are supported by the evaluation environment Mercury. Mercury provides support to EFM, SPN, RBD and CTMC models. In the Mercury engine, algorithms were implemented to traverse the EFM to perform the power verifications as well as to estimate data centers cost and sustainability impacts. Additionally, an optimization engine is able to communicate with Mercury to conduct the evaluation and provide the results for different optimization algorithms.

Figure 4.1 depicts an overview of the proposed methodology for evaluating dependability, cost and sustainability issues of data center infrastructures. The methodology first step concerns understanding the system, its components, their interfaces and interactions. This phase should also provide (as product) the set of metrics (e.g., availability, reliability, costs) that should be evaluated. The next broad phase aims at creating the high-level models (e.g., from ASTRO power view) that represent the data center architecture. The high-level models allow data center designers to specify power, cooling and IT systems following the standard adopted by engineers. These models can be converted to dependability models (e.g., Fault tree, Continuous Time Markov Chain (CTMC), SPN and RBD). It is important to state that submodels may be generated to mitigate the complexity of the final model. The evaluation of each submodel provides the system results.

The following step is the creation of dependability (e.g., RBD or SPN) and energy flow models. These models allow the integrated evaluation of dependability, cost and sustainability. Additionally, the EFM verifies if the energy flow does not exceed the maximum power capacity that each component can provide (considering electrical devices) or extract (assuming cooling equipment). Both models (dependability and EFM) may be the input of optimization methods (see Section 2.6). The optimization methods can be either executed or not.

An evaluation process is directly conducted to provide the estimate results (e.g., acquisition and operational costs, availability, downtime, exergy consumption) when the optimization process is not selected. Otherwise, the data center designer informs the metrics (e.g., cost, availability) that should be improved. A list of candidate elements

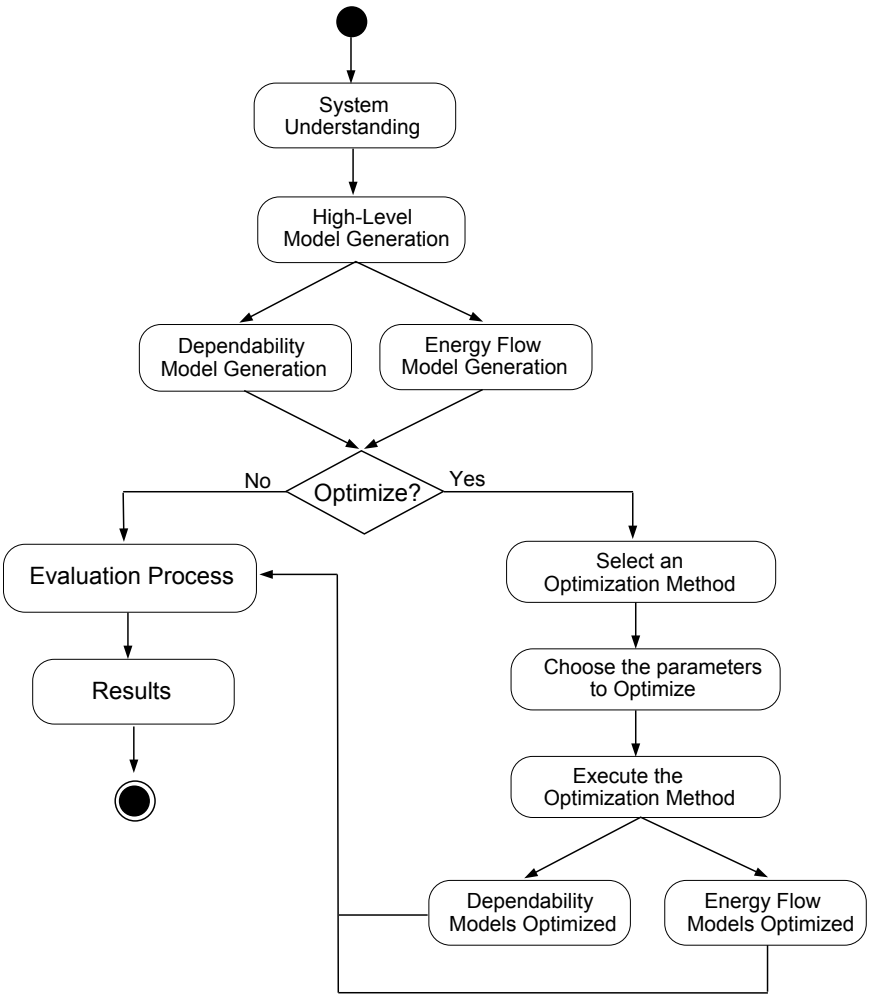


**Figure 4.1:** Methodology

is also provided representing a library of components (e.g., UPS, STS, power strip) with different MTTFs, acquisition cost and energetic efficiency, for instance. The method evaluates first the dependability model to obtain the availability result. Next, it adopts the availability result to compute the sustainability and cost issues while respecting the system restrictions quantified by the EFM. The adopted optimization approach provides both dependability and EFM models optimized. These models are composed of the appropriate components from the candidate list that optimizes the desired metrics. Lastly, the evaluation of those models provide the metrics (e.g., availability, cost and sustainability) optimized.

Optimization algorithms have been adopted to optimize the fundamental models (RBD, SPN and EFM). This work considers a GRASP based algorithm and a Power Load Distribution Algorithm (PLDA) [57]. The GRASP based method improves the results obtained by RBD, SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative

components that may compose the data center architecture. The PLDA is an algorithm that improves the power load distribution among data center power devices by changing the weights of the edges present on EFM models. The main goal of this algorithm is to automatically provide a power distribution that help data center designers to define power architectures respecting the system restrictions that are quantified by the EFM. Additionally, it is important to highlight that this work has considered two optimization algorithms, however, others methods can be easily included.

## 4.2 ASTRO

ASTRO [58] provides interfaces that allow data center designers to model power, cooling and IT systems without knowing the formalism (e.g., SPN, RBD, CTMC and EFM) that has been employed. This interface is according to the engineering standards [59]. The models created through ASTRO can be converted into the fundamental models. Mercury provides support for modeling and evaluating the fundamental models that are SPN, RBD, EFM and CTMC. Mercury is an independent environment with its graphical interfaces that supports other tools such as ASTRO and the algorithms implemented in the optimization module. Figure 4.2 illustrates the relationship between ASTRO, Mercury and optimization module. The following sections present ASTRO.

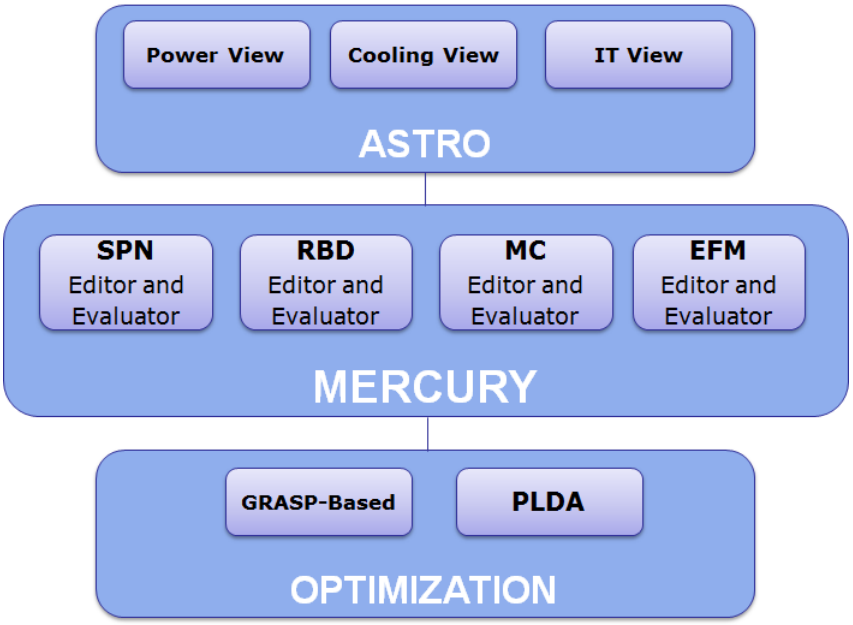


**Figure 4.2:** Evaluation Environment.

### 4.2.1 Power, Cooling and IT System editors

The power system editor provides a high-level view for modeling data center power system infrastructures. The designer may define the component attributes of a system. For instance, dependability, cost and sustainability attributes. Basically, the editor provides icons to represent each component (i.e., equipment), in such a way that the designer specifies the respective connections through arcs. For instance, if a component 'A' provides power to a component 'B', there is an arc that connects 'A' to 'B'. Figure 4.3 depicts the power system editor. Since ASTRO provides high-level models, the models created through these editors must be converted into RBD, SPN, CTMC or EFM to allow dependability, sustainability and cost evaluations.

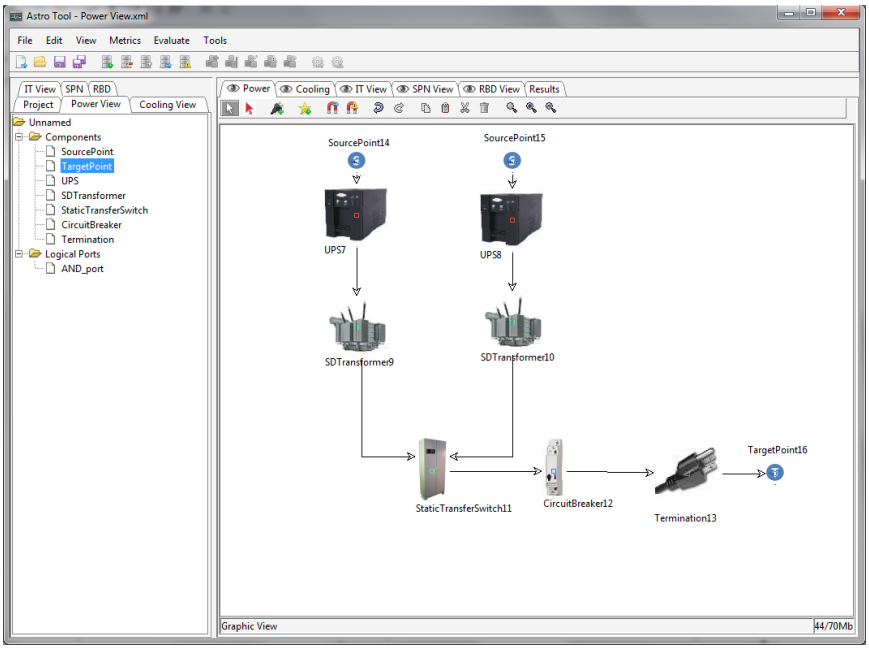


**Figure 4.3:** ASTRO environment - power system editor

Similarly to the power editor, the cooling and IT system editors adopt high-level models for representing cooling and IT data center infrastructures. These editors also provide functionalities to translate the high-level models into RBD, SPN, CTMC and EFM models to compute dependability and sustainability metrics, for instance.

## 4.3 MERCURY

Dependability models are classified into state-space and non-state-space models[60]. Mercury provides the support of RBDs for non-state-space models. Dependability evaluation using RBDs can be conducted under the assumption that the failure of the system components are independent. To model systems with more complex interactions between

components, Mercury provides support for Markov Chain (MC). The hand construction of Markov models are tedious and error-prone when the number of states becomes very large [60]. Therefore, Mercury also provides the support for the creation and evaluation of other state-space model, SPNs. SPN models can be automatically converted into Markov models to be solved.

In addition, Mercury also support the creation and evaluation of EFM. The EFM computes the sustainability and cost estimates of the power and cooling infrastructures whilst conforming to the energy constraints of each device. These estimates are provided by algorithms which compute the metrics of interest by traversing the EFM. The following subsections present the SPN, CTMC, RBD and EFM editors and evaluators.

### 4.3.1 SPN editor and evaluator

In Markov Chain and SPN models, representative numerical techniques are available for both stationary (i.e., GTH and Gauss-Seidel [61] [25]) and transient (i.e., Uniformization and Runge-Kutta [8]) analysis. Time-dependent metrics are obtained by transient analysis or simulations, whilst steady-state metrics are achieved with stationary analysis or simulations. Regarding SPN models, the evaluation environment also allows dependability evaluation through transient and stationary simulations [62]. The simulation is adopted when the state space becomes very large.

Figure 4.4 depicts the SPN editor, in which the models can be obtained from a high level model translation or created by the user from scratch. To assist the validation of SPN models, the editor has a feature, namely, token game, which makes possible the firing of transitions graphically and interactively according to the current net marking. The following subsection presents the simulation process adopted in Mercury.

#### Simulation

Mercury adopts simulation as one of the mechanisms for evaluating SPN models. Two modalities are available: (i) transient simulation, which is time-dependent; and (ii) stationary simulation, which assumes an infinite length of time (i.e., long-run behavior). The simulation process for both modalities is almost the same, discerning in some aspects of the stop criteria (e.g., the simulation time for the transient approach). In general, Mercury environment allows setting the maximum relative error, the confidence interval, the minimum number of firings for each transition and the maximum simulation time. For a better understanding, the adopted simulation process (Figure 4.5) is presented as follows.

Firstly, statistical counters (variables adopted for storing statistical information about system performance) are initialized and the parameters of the stop criteria are set. Next, the simulation clock (variable indicating the current time of the simulation) is set to “0” and the event list is created (i.e., list that stores the enabled transitions). Afterwards, the simulation enters a loop, which corresponds to the actual evaluation of the SPN model. In each iteration, the enabled transitions are added to the list and the disabled transitions

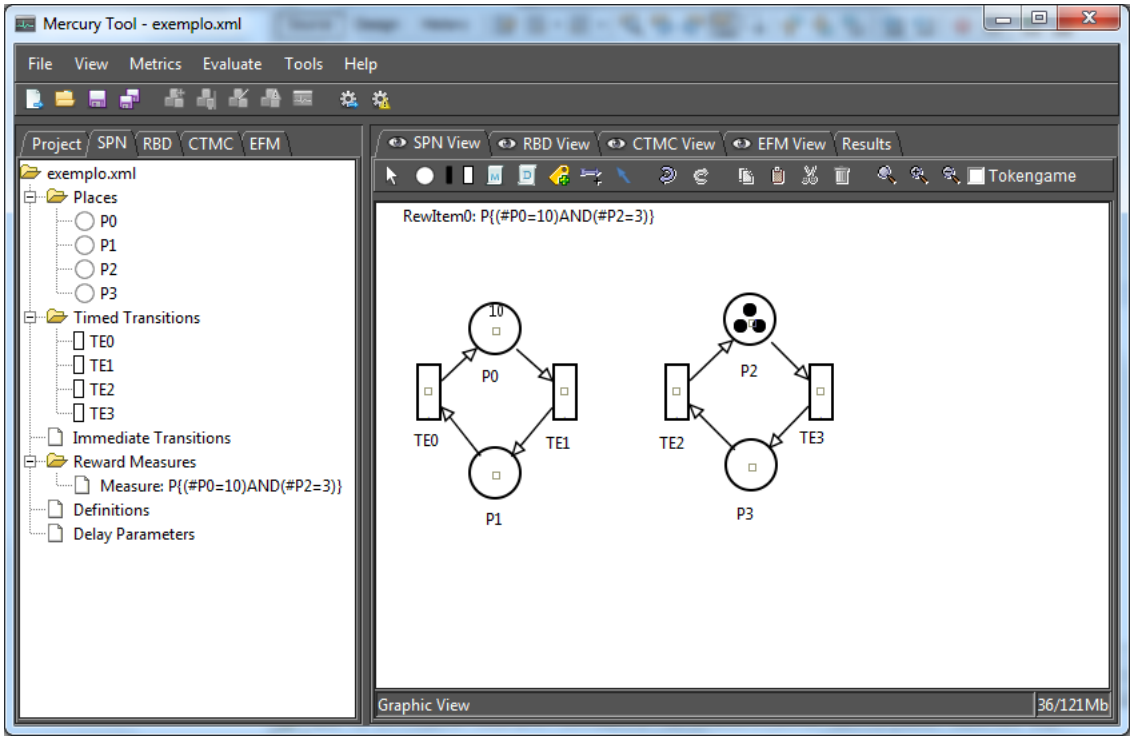


**Figure 4.4:** Mercury - SPN editor and evaluator

are removed.

From the event list, the simulation engine selects the transition with the smallest time, which was generated according to a random variable satisfying the transition's distribution (e.g., exponential distribution). The selected transition is fired, and the simulation clock, statistical counters and the marking of the SPN model are updated. If the simulation criteria are achieved (e.g., the relative error is satisfied), the simulation stops.

It is important to state that batches of simulation runs are performed (i.e., groups of simulations) to estimate the metrics of interest. Basically, the number of batches depends on the specified confidence degree, relative error and the initial number of runs set by the designer. Besides, the number of simulation batches are updated, whenever the error calculated for the metrics (after the initial batches) does not satisfy the relative error informed by the user [63].

### 4.3.2 CTMC editor and evaluator

Mercury environment provides an editor and evaluator (Figure 4.6), in which users can create, edit and evaluate CTMC models. Representative numerical techniques are avail-

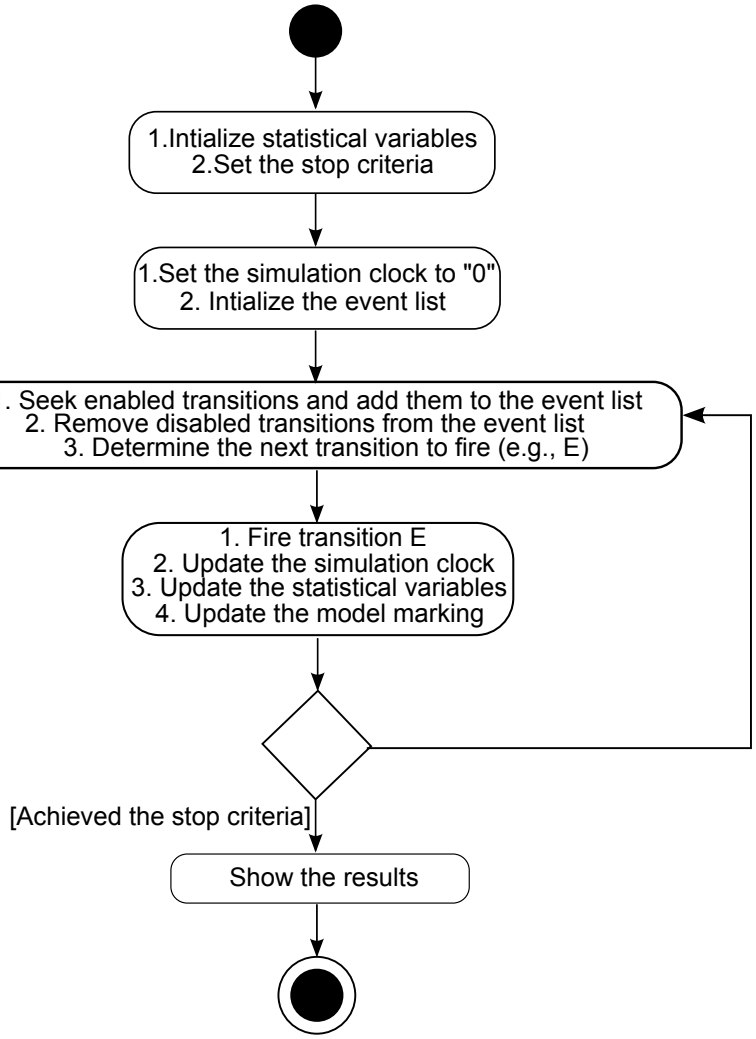


**Figure 4.5:** Mercury - Simulation process

able for stationary analysis (i.e., GTH, Gauss-Seidel [8]) as well as transient evaluation (i.e., Uniformization, Runge-kutta and Trapezoid-Euler [8]).

### 4.3.3 RBD editor and evaluator

The RBD editor and evaluator performs reliability and availability analysis using block diagrams. The types of reliability configurations supported by the RBD editor and evaluator are: Series, Parallel, K-out-of-n and Bridge. In this editor, a diagram should only contain one input and one output node. RBDs provide closed-form equations, so the results are usually obtained faster than using SPN simulation. However, there are many situations (e.g., dependency among components) in which modeling using RBD is harder than adopting SPN. Figure 4.7 shows the RBD view of the tool, such that some

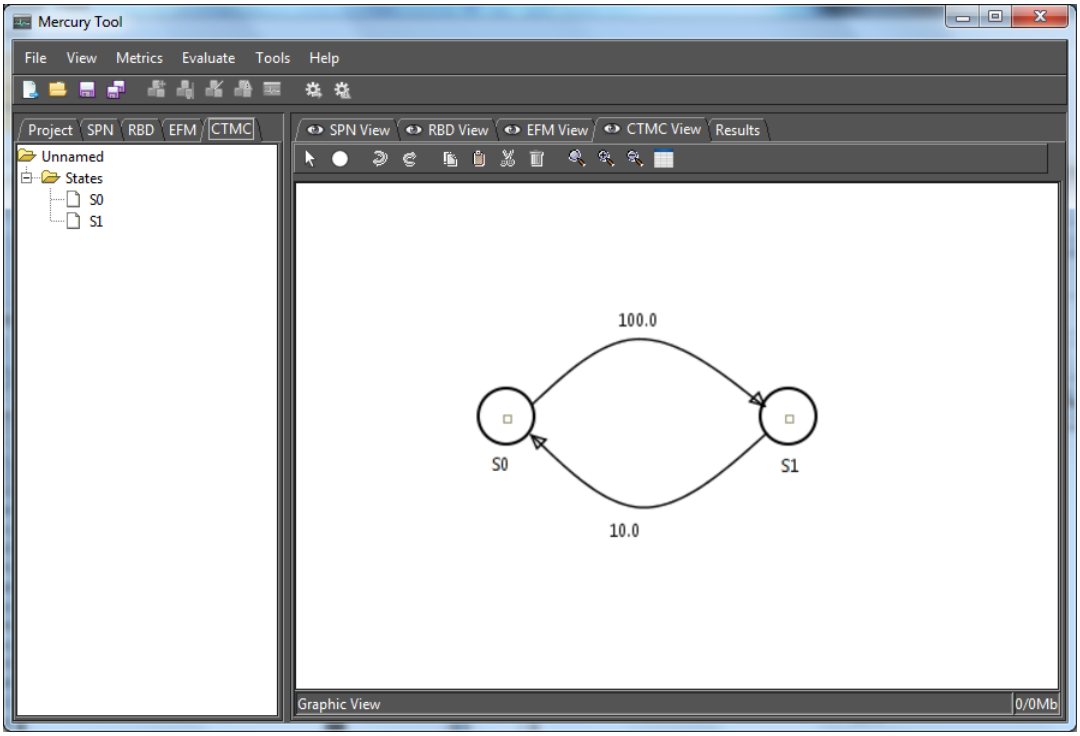


**Figure 4.6:** Mercury environment - CTMC editor and evaluator

components are in series and parallel arrangement.

Additionally, the RBD evaluator allows the calculation of reliability importance, structural and logical functions as well as bounds of dependability measures. Reliability Importance (RI) is a technique that allows to quantify the improvement in system reliability due to a component, when the respective reliability is increased by one unit. Structural and logical functions are alternative ways of representing the system mathematically, in which the former adopts algebraic functions and the latter utilizes logical expressions. The benefits of such representations contemplate additional tools which may indicate system parts that can be replicated to increase overall availability in a simpler way, for instance.

Bounds of dependability measures is a method to calculate dependability metrics (e.g., reliability) when the RBD model is too large. In this case, such a technique can provide approximations for the exact metric faster than solving all closed-form equations precisely. In short, the bounds are obtained using minimal paths and/or minimal cuts. Minimal path is the minimal number of components in operational state that guarantee the system functioning, whereas minimal cut is the minimal number of components in failure state that lead to system failure. As the bounds are calculated iteratively, ASTRO environment shows the estimated bounds according to the user parameters and, also, all intermediary values calculated (see Figure 4.8).

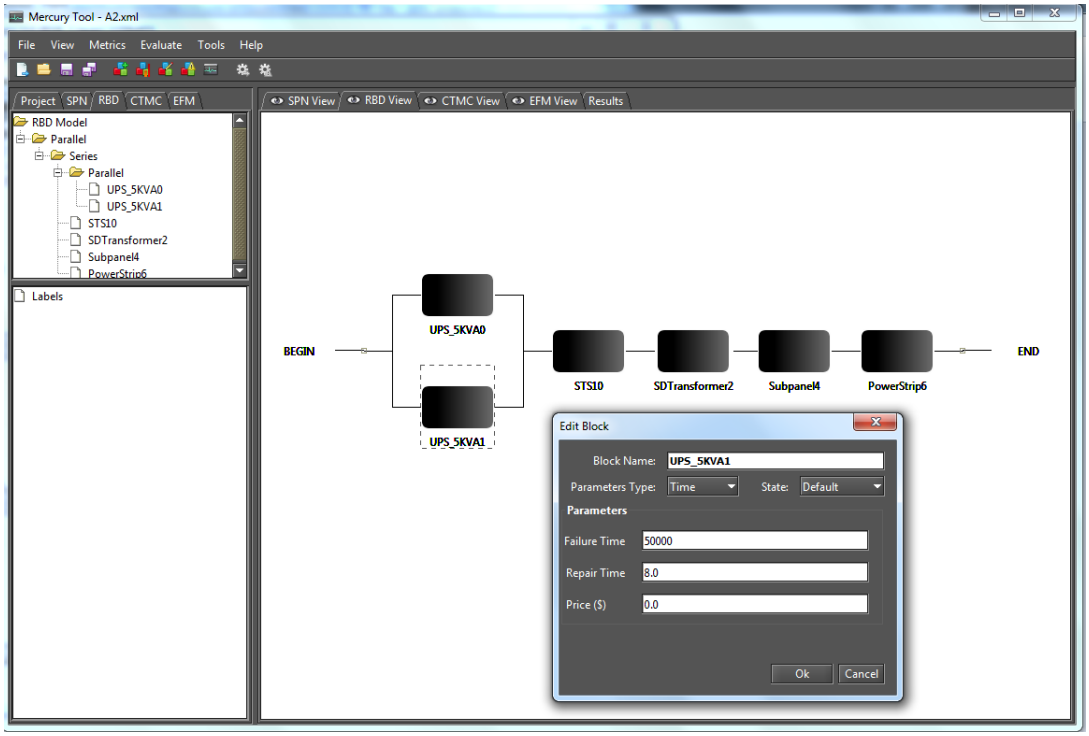


**Figure 4.7:** Mercury environment - RBD editor and evaluator

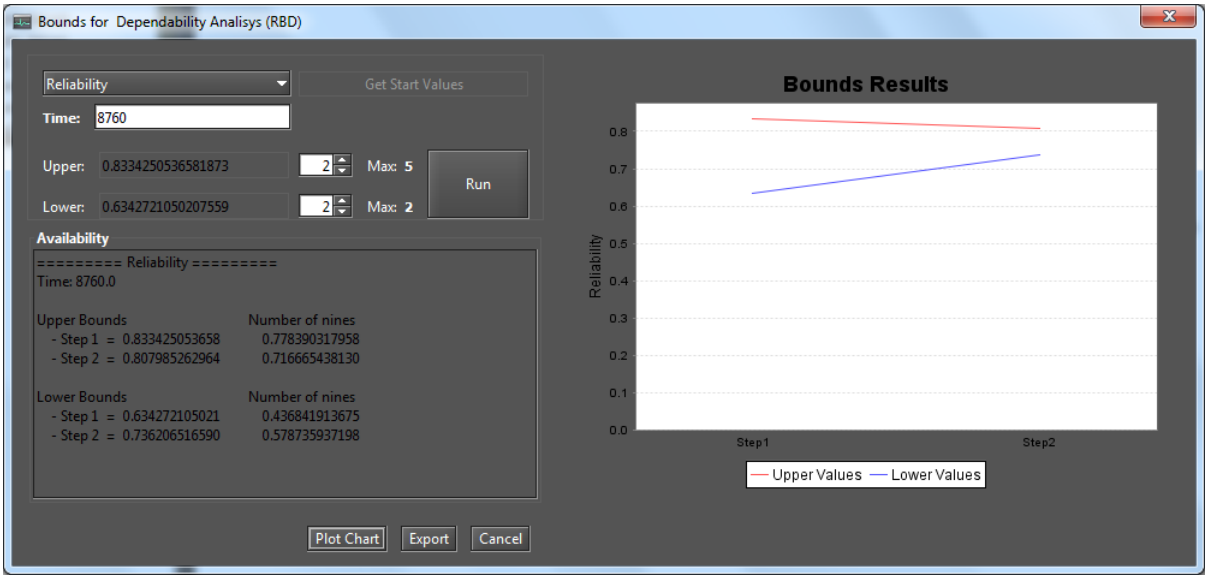


**Figure 4.8:** Mercury environment - bounds evaluation

### 4.3.4 EFM Editor and Evaluator

Figure 4.9 depicts the EFM editor and evaluator. The EFM is proposed to compute the sustainability and cost estimates of data center power and cooling infrastructures without

exceeding the power constraints of each device. The EFM is defined as a XML file to allow the model to communicate to other tools. For instance, the proposed GRASP based method (Section 2.6) is implemented as a separate module that receives as a parameter the EFM model to compute the exergy consumption and cost issues as well as the RBD or SPN models to compute the dependability metrics. Each model is executed several times with different parameters to generate the optimized versions of the models.

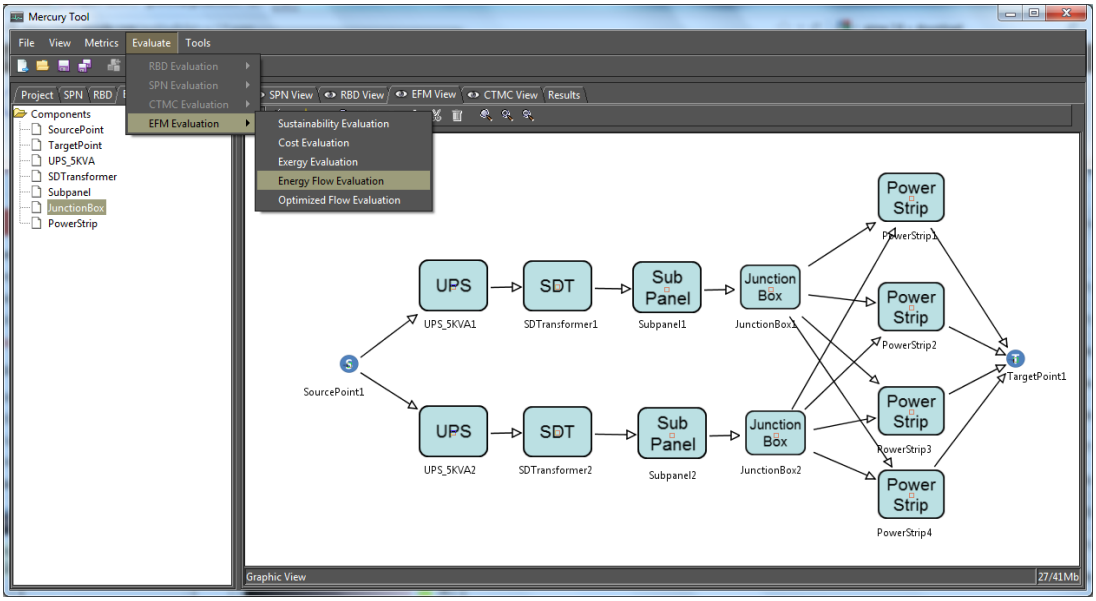


**Figure 4.9:** EFM Editor and Evaluator.

In the EFM, the demanded energy that should be provided or extracted is associated to the *TargetPoint1* node considering power models (or to the *SourcePoint1* in case of cooling system). The weights associated to each edge is adopted for representing the energy that should flow when a fork structure is present. Although not shown in the example depicted in Figure 4.9, multiple source and target nodes may be considered to represent a data center with more than one source point.

Figure 4.10 shows the evaluation results of the energy flow (Figure 4.10 (a)) and the integrated evaluation of sustainability, cost without exceeding the power constraints (Figure 4.10 (b)). In the energy flow results depicted in Figure 4.10 (a), the reader should notice that the first component exceeding the energy constraint is detected and shown in the results.

## 4.4 OPTIMIZATION MODULE

The optimization module is able to evaluate RBD, SPN, CTMC and EFM models by calling the Mercury engine. The Mercury conducts the evaluation and provides the results to the optimization module. A GRASP based algorithm and a power load distribution

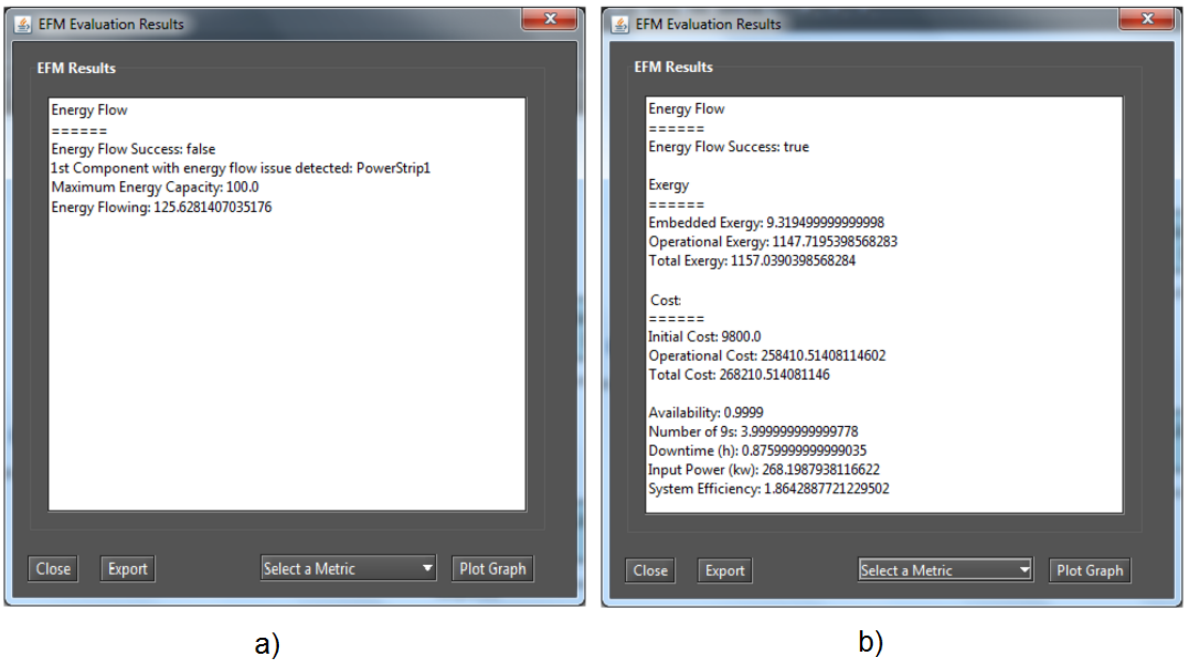


**Figure 4.10:** EFM Example Results: a) Energy Flow Result; b) All Results.

algorithm (PLDA) [57] represent two optimization techniques that were implemented in that module. The proposed GRASP based method improves the results obtained by RBD, SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center architecture. The PLDA is an algorithm that improves the power distribution among devices by changing the weights of the edges present on EFM models.

Optimization algorithms are able, for instance, to reduce the number of scenarios to be evaluated and, therefore, the time spent to achieve a solution is also reduced. However, there is no guarantee that the solution obtained through optimization algorithms provide the optimal result. Indeed, in many cases, optimization algorithms achieve results close to the optimal ones. Additionally, the necessary time to evaluate all possible scenarios may turn this strategy not recommended for complex systems. Therefore, optimization strategies represent an alternative option that is expected to achieve results close to the optimal in a reduced runtime.

Figure 4.11 shows a screenshot of the optimization tool from which users are able to select the optimization method as well as to set the respective parameters. In the proposed Grasp based optimization algorithm, some parameters are the number of elements adopted by a benchmark to create the list of candidate devices that may compose the final optimized model; the greedness degree; and the number of iterations that the optimization algorithm executes, where a higher number provides results closer to the optimal however demanding more time to be executed. Additionally, users define the

desired metrics (e.g., cost, availability and sustainability) to be optimized. In the Grasp based optimization algorithm, the optimization process consists on the evaluation of the model considering different devices from the list of candidate components. In addition, different seeds are also adopted. Figure 4.12 shows a spreadsheet that is automatically filled out with the evaluation results.

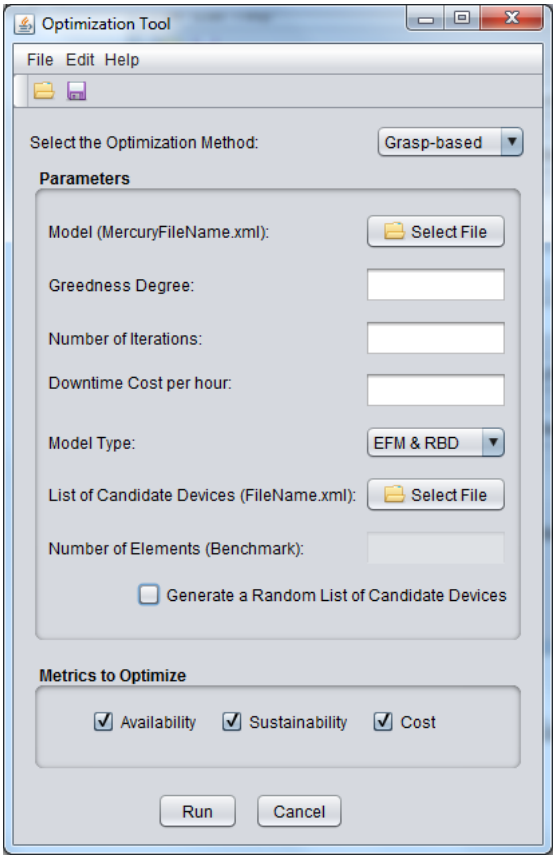


**Figure 4.11:** Screenshot of the Optimization Tool.

B1 [Name=UPS_5KVA0 Type=UPS_5KVA Id=3, Name=UPS_5KVA0 Type=UPS_5KVA Id=3, Name=SDTransformer2

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Solution | [Name=U | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI | [Name=UI |
| 2 | | | | | | | | | | | | | |
| 3 | Availability | 0.999829 | 0.999823 | 0.999847 | 0.99983 | 0.999851 | 0.999856 | 0.999859 | 0.999862 | 0.99983 | 0.999827 | 0.99986 | 0.99986 |
| 4 | 9s | 3.767917 | 3.751647 | 3.81431 | 3.770743 | 3.826025 | 3.841545 | 3.851487 | 3.860311 | 3.76887 | 3.761984 | 3.853896 | 3.854501 |
| 5 | Downtime | 1.494812 | 1.551875 | 1.343365 | 1.485118 | 1.307611 | 1.261709 | 1.233154 | 1.20835 | 1.491538 | 1.515376 | 1.226331 | 1.224625 |
| 6 | Downtime Cost | 298962.5 | 310375.1 | 268672.9 | 297023.5 | 261522.3 | 252341.8 | 246630.7 | 241670 | 298307.6 | 303075.3 | 245266.2 | 244925 |
| 7 | Aquisition Cost | 9321 | 7771 | 8887 | 8302 | 8596 | 8484 | 7954 | 8761 | 8474 | 8697 | 8134 | 8826 |
| 8 | Operational Cost | 6533.812 | 6211.58 | 7269.551 | 6336.829 | 5996.932 | 7971.679 | 6152.468 | 6689.916 | 7075.879 | 7209.021 | 7003.499 | 6844.931 |
| 9 | Total Cost | 15854.81 | 13982.58 | 16156.55 | 14638.83 | 14592.93 | 16455.68 | 14106.47 | 15450.92 | 15549.88 | 15906.02 | 15137.5 | 15670.93 |
| 10 | Exergy Consumption | 56.18075 | 45.63601 | 80.25676 | 49.73385 | 38.60677 | 103.234 | 43.69569 | 61.28447 | 73.92106 | 78.27889 | 71.5475 | 66.35797 |
| 11 | System Efficiency | 0.737269 | 0.775511 | 0.662663 | 0.760188 | 0.803291 | 0.604303 | 0.78299 | 0.720089 | 0.680789 | 0.668214 | 0.687846 | 0.70378 |
| 12 | Func Objective | 314817.3 | 324357.6 | 284829.5 | 311662.4 | 276115.2 | 268797.4 | 260737.2 | 257120.9 | 313857.5 | 318981.3 | 260403.7 | 260595.9 |
| 13 | Iteraction | 23 | 0 | 0 | 1 | 1 | 3 | 11 | 1 | 3 | 14 | 0 | 20 |
| 14 | Execution time (s) | 0.511 | 0.131 | 0.129 | 0.133 | 0.162 | 0.123 | 0.128 | 0.122 | 0.129 | 0.188 | 0.182 | 0.135 |
| 15 | | | | | | | | | | | | | |
| 16 | Execution time: 00:00:05.965 | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | |

Optimization

**Figure 4.12:** Spreadsheet results of the Grasp based optimization algorithm.

## 4.5 SUMMARY

This chapter presented the evaluation environment that is composed of ASTRO, Mercury and optimization module. ASTRO was described showing the data center power and cooling views. Next, the Mercury that supports SPN, RBD, EFM and CTMC models was presented. After that, the optimization module and its functionalities were described to allow data center designer to optimize the desired metrics such as dependability, sustainability and cost of data center architectures.

CHAPTER 5

# CASE STUDIES

This chapter presents case studies that illustrate the applicability of the proposed methodology and models with the support of the developed evaluation environment composed of ASTRO, Mercury and optimization module to evaluate data center power and cooling infrastructures. The first case study is conducted to provide insights on validation of the proposed evaluation environment. Next, a study that analyzes more complex data center power infrastructures through the proposed set of models and following the adopted methodology was conducted. This study also considers RI index. Afterwards, the case study III main goals are to present a comparative study of real-world data center architectures (from HP Labs Palo Alto, U.S. [64]) assessing dependability as well as environmental impact and operational energy cost associated to the energetic mix of U.S. and Brazil. Finally, the last two case studies are conducted to present the proposed optimization algorithm applied on simple data center power architectures as well as on a real-world infrastructure.

## 5.1 CASE STUDY I

The main goal of this case study is to present the first evaluation results related to dependability, cost and sustainability issues obtained through the proposed models following the adopted methodology. This case study also provides insights on validation demonstrating the accuracy of the evaluation environment (ASTRO and Mercury) results in comparison to TimeNET [65] and Sharpe [66] tools.

Five data center power infrastructures have been considered (see Figure 5.1) with increasing redundancy, such that each successive architecture has an additional component duplicated. For each architecture, we estimate: (i) availability, (ii) sustainability impact and (iii) costs. In the second part of this study, scenarios considering cheaper devices with lower MTTFs are analyzed to show that similar availability levels can be obtained with lower cost due to the redundancy strategies adopted.

### 5.1.1 Models

In the baseline architecture, A1, there are no redundant components. Figure 5.2 depicts the EFM, SPN and RBD models correspondent to the architecture A1. The MTTFs,

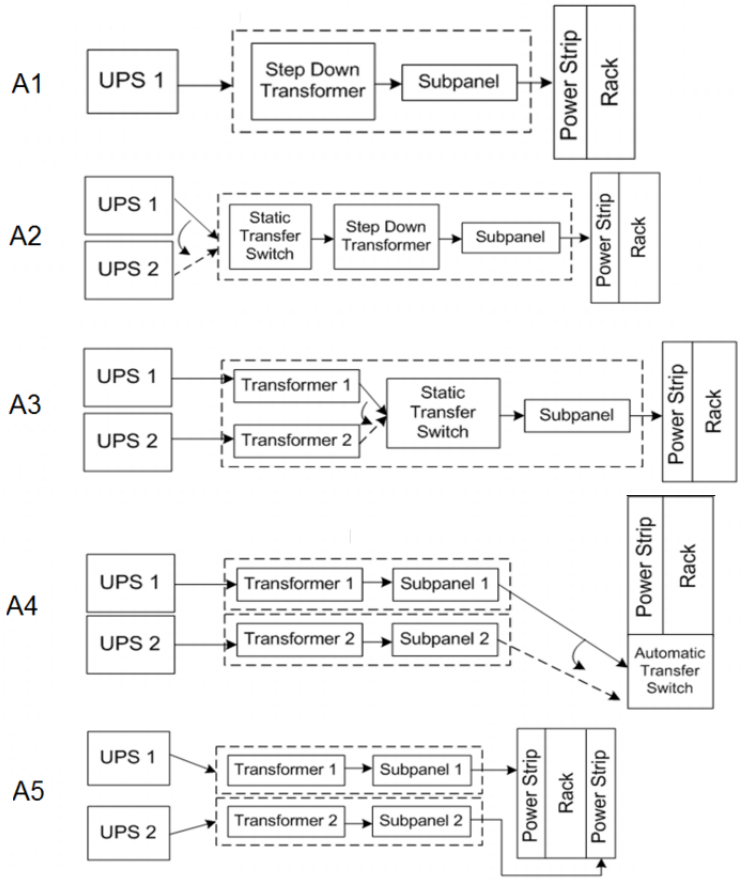


**Figure 5.1:** Data Center Power Infrastructures.

MTTRs, acquisition cost and energetic efficiency values for each equipment were taken from [67] [68] [69] and are shown in Table 5.1.

**Table 5.1:** MTTF, MTTR, acquisition cost and efficiency values for power devices

| Equipment | MTTF (hs) | MTTR (hs) | AC (USD) | Eff (%) |
|---|---|---|---|---|
| ATS | 500,000.00 | 0.33 | 800.00 | 99.5 |
| STS | 240,384.62 | 6.00 | 800.00 | 99.5 |
| Subpanel | 1,520,000.00 | 2.40 | 200.00 | 99.5 |
| Step Down Transformer | 1,412,908.33 | 156.01 | 550.00 | 98.5 |
| UPS | 250,000.00 | 8.00 | 3,600.00 | 95.3 |
| Power Strip | 11,511,175.63 | 3.80 | 200.00 | 99.5 |

Regarding the SPN model (Figure 5.2 (c)), the MTTF related to the UPS compo-

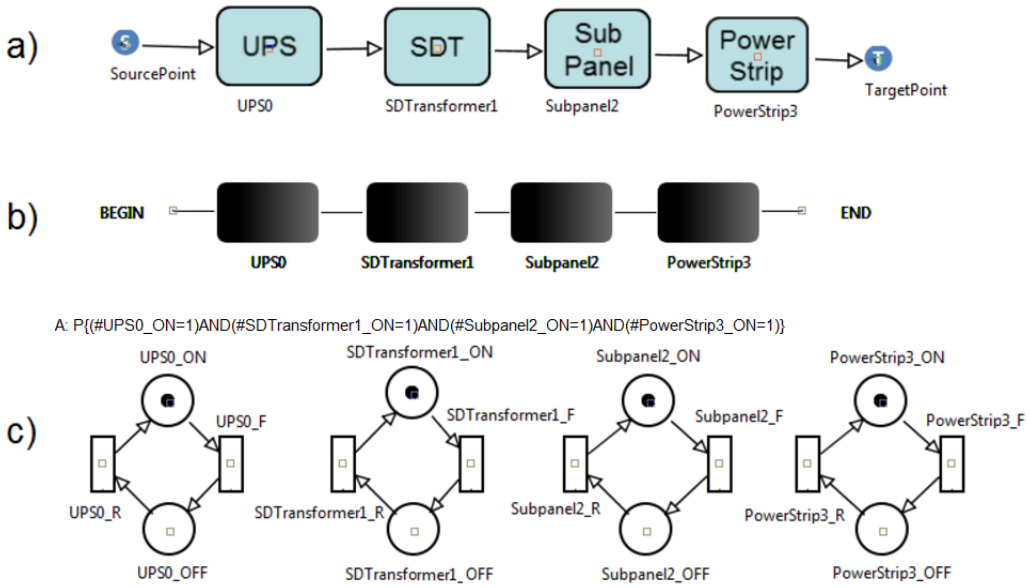


**Figure 5.2:** (a) EFM, (b) RBD and (c) SPN models.

nent is assigned to the transition UPS0_F, while the MTTR is depicted in the transition UPS0_R. Furthermore, the place UPS0_ON represents the UPS operational state, while UPS0_OFF is the failure state. The same representation is adopted for all other architecture components. The availability is computed by the probability expression P{(#UPS0_ON=1)AND(#SDTransformer1_ON=1)AND(#Subpanel2_ON=1)AND(#PowerStrip3_ON=1), where "#" represents the number of tokens present in a place.

Figure 5.2 (b) depicts the respective RBD model related to architecture A1. Note that the RBD model adopts a series composition, because there is no redundancy. It is important to stress that the evaluation conducted through both models produce equivalent results.

Figure 5.2 (a) presents the correspondent EFM model for the architecture A1. The acquisition cost and energetic efficiency of each device is present on Table 5.1. In addition, the demanded power (10kW) by the IT system is associated to the target node $T$ of the EFM model.

Architectures A2, A3, A4 and A5 have additional redundant devices, such as an UPS (A2, A3, A4, A5), a transformer (A3, A4, A5), a subpanel (A4, A5) and a rack power strip (A5). Architecture A5 corresponds to the baseline (A1) architecture considering all components redundant. In this representation, a STS is not required, since modern IT devices have dual corded systems.

### 5.1.2 Experiment I - Results

Table 5.2 presents the summary results of the evaluation of those 5 architectures previously detailed, in which $9s$ represents the availability depicted in number of nines calculated using the following equation:$-log(1 - Avail(\%)/100)$. As expected, the availability increases when more redundant components were considered. However, the availability increase from A1 to A2 is minimal due to the fact that to add the additional UPS, a STS was also added. This additional STS component represents a single point of failure and, therefore, the system availability increase was minimum. The number of nines doubles from A1 to A5, as expected, since A5 is essentially two instances of A1 connected in parallel.

**Table 5.2:** Summary results for all architectures.

| Architecture | Avail(%) (9s) | Exergy(GJ) | Cost (USD) | Effic (%) |
|---|---|---|---|---|
| A1 | 99.98556 (3.84) | 37.99 | 20038.43 | 93.3 |
| A2 | 99.98627 (3.86) | 44.28 | 24516.38 | 92.8 |
| A3 | 99.99731 (4.57) | 44.62 | 25068.09 | 92.8 |
| A4 | 99.99989 (5.99) | 45.03 | 25268.50 | 92.8 |
| A5 | 99.99999 (7.68) | 42.06 | 24590.67 | 93.3 |

In Table 5.2 *Cost* corresponds to both the retail price of the system and the operational cost as a fraction of total data center costs, *Exergy* is the operational exergy destroyed and *Effic* is the electrical system efficiency. The assumed period was 5 years.

For a better visualization, Figure 5.3 shows a comparison between the availability, cost and exergy in relation to the baseline architecture A1. The big leap from A1 to A2 in terms of cost reflects the high cost of UPS as compared to the other components. The variations in the exergy destroyed are primarily due to additional hardware, since the difference in operational power consumption is negligible. Moreover, from Figure 5.3, it can be seen that the availability increased significantly from A2 to A3 (almost 20%) and from A3 to A4 (about 35%), the sustainability impact increase is negligible. Thus, minimizing redundancy may not always lead to optimal sustainability. The sustainability impact decreased from A4 to A5, since a STS or ATS is not required in A5.

Additionally, Figure 5.3 also illustrates that both cost and sustainability impact slightly increased from A2 to A4, although these parameters do not always change together. While A2 has a lower cost, A5 has lower exergy consumption. Thus, optimizing cost will not necessarily lead to optimal sustainability. Therefore, architecture A5 corresponds to an interesting alternative, since it has the highest availability, the cost is almost identical to the architecture A2 and it has the smallest environmental impact.

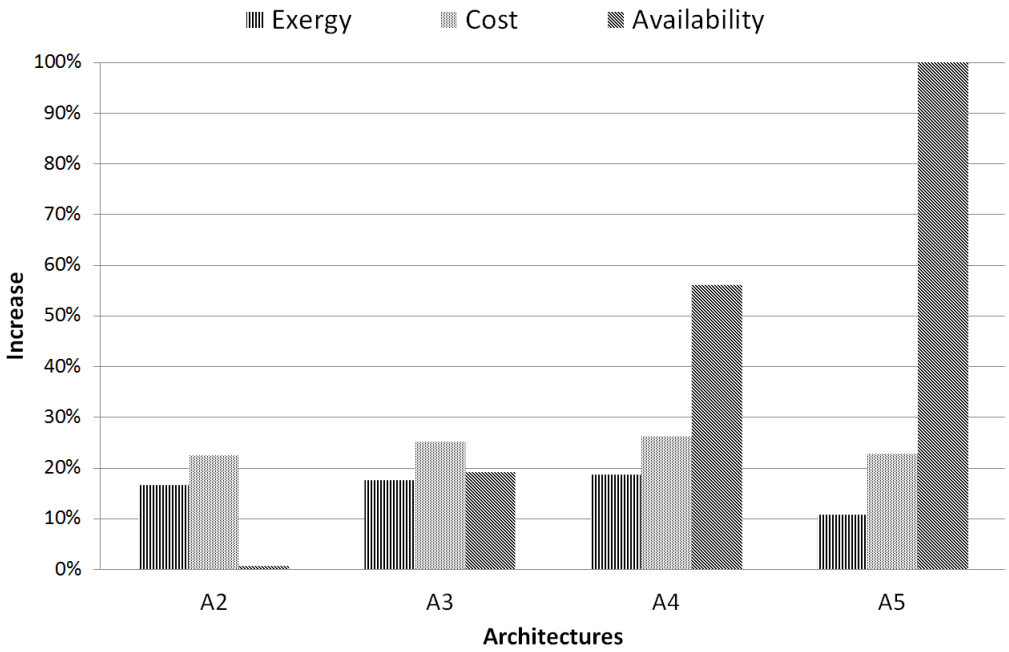


**Figure 5.3:** Comparison: Availability, Exergy Consumption and Cost.

### 5.1.3 Insights on validation

To demonstrate the accuracy of the availability results obtained by the proposed tools, this work compared the obtained values for the case study I with the respective results generated by TimeNET [65] and Sharpe tools [66]. Five different architectures (A1...A5) for a data center power infrastructure were considered, and the results are detailed in Table 5.3.

The paired-t test [70] was conducted to analyze the results between ASTRO and TimeNET as well as between ASTRO and Sharpe. Assuming a significance level of 5%, the first test resulted in the following confidence interval:[-0.0037174, 0.004621]. As 0 (zero) is enclosed by that interval, there is no evidence to reject the hypothesis of equivalence between the values generated by both tools. Similarly, the same test was applied for ASTRO/Mercury and Sharpe, obtaining $0 \in [-0.0001475, 0.000452]$ and, thus, the hypothesis of equivalence cannot be refuted.

### 5.1.4 Part II

The second part of this case study presents alternative scenarios in which cheaper devices with lower MTTFs values were considered. Architecture A5 has been adopted as the baseline, and each scenario has the same structure as well as the same amount of devices.

In the first scenario ($S1$), the devices are assumed to be 10% cheaper and the respective

**Table 5.3:** Availability values obtained with Mercury, TimeNET and Sharpe

| Architecture | Mercury | TimeNET | Sharpe |
|---|---|---|---|
| A1 | 99.98556 | 99.9892 | 99.985 |
| A2 | 99.98627 | 99.98032 | 99.98596 |
| A3 | 99.99731 | 99.99712 | 99.99715 |
| A4 | 99.99989 | 99.999904 | 99.999931 |
| A5 | 99.99999 | 99.999997 | 99.9999977 |

MTTFs 50% lower than the ones considered in the baseline. The second scenario ($S2$) adopts devices 20% cheaper than the ones in the baseline and the MTTFs are decreased by 75%.

Table 5.6 presents the results for this study, in which the reader should observe that, although considering cheaper devices with reduced MTTFs values, the availability is not considerably reduced. Indeed, comparing the cost values of these two scenarios with the part I of this study, it is possible to note that the part II results are cheaper than architecture A4, A3, A2 (see Table 5.2 from part I) and the availability is higher. Therefore, the adoption of cheaper devices represents an interesting option, in which the environmental impacts are basically equivalent to the ones obtained from architecture A5.

**Table 5.4:** Summary results for case study I - part II.

| Scenarios | Avail(%) (9s) | Exergy(GJ) | Cost (USD) |
|---|---|---|---|
| S1 | 99.9999917 (7.07) | 42.06 | 23680.67 |
| S2 | 99.9999667 (6.47) | 42.06 | 22770.67 |

## 5.2 CASE STUDY II

The main goal of this study is to analyze more complex data center power infrastructures through the proposed set of models and following the adopted methodology. This study also considers RI index that is adopted for supporting data center designers to identify which component should be replicated to increase the system reliability/availability. From a base line architecture, other architectures are proposed according to the RI index. Seven data center power infrastructures were considered as depicted in Figure 5.4. For each architecture, we estimate: (i) availability; (ii) the sustainability impact and (iii) costs.

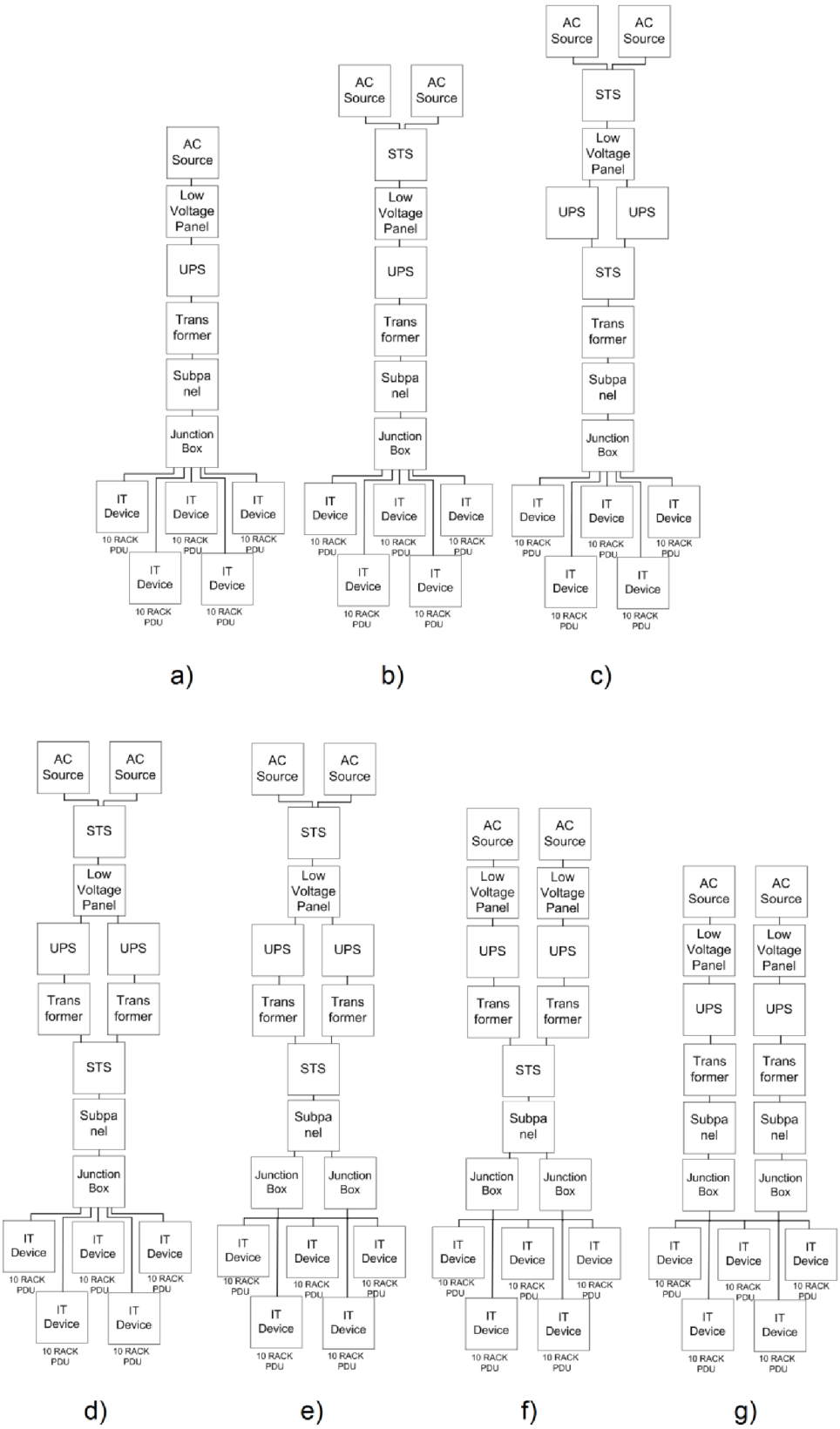


**Figure 5.4:** a) Architecture A1, b) Architecture A2, c) Architecture A3, d) Architecture A4, e) Architecture A5, f) Architecture A6, g) Architecture A7.

### 5.2.1 Architectures

From the power infrastructure A1 depicted in Figure 5.4 (a), we propose other architectures that are created according to the component reliability importance index. Originating in the utility feed (i.e., AC Source), typically, the power goes through voltage panel, uninterruptible power supply (UPS) unit, power distribution units (PDUs) (composed of a transformer and an electrical subpanel), junction boxes, and, finally, to rack PDUs (rack power distribution units).

The system is operational if the power infrastructure is able to supply energy to the IT devices. For the sake of readability, each IT device block represents 10 racks and, so, the adopted infrastructure provides electrical power to 50 IT racks. MTTFs and MTTRs values for the components were obtained from [67].

We started performing the dependability evaluation of the baseline architecture (A1). As it will be further explained, six new architectures are generated as shown in Figure 5.4. Table 5.5 presents the importance index for the components in each architecture. In that table, it is possible to note that the AC Source on architecture A1 has the highest *RI* value. Thus, architecture A2 (see Figure 5.4 (b)) is created replicating that equipment. A static transfer switch (STS) is also added for switching from a failed AC Source to the operational one.

**Table 5.5:** Reliability Importance Values of Architectures A1-A7.

| | ACS | STS | UPS | SDT | JB | LVP | SP | ITD |
|---|---|---|---|---|---|---|---|---|
| A1 | 1 | | 0.140 | 0.137 | 0.136 | 0.136 | 0.136 | 3.3E-08 |
| A2 | 1 | 0.303 | 0.302 | 0.295 | 0.294 | 0.293 | 0.293 | 7.2E-08 |
| A3 | 1 | 0.303 | 0.010 | 0.295 | 0.294 | 0.293 | 0.293 | 7.2E-08 |
| A4 | 1 | 0.303 | 0.013 | 0.013 | 0.294 | 0.293 | 0.293 | 7.2E-08 |
| A5 | 1 | 0.303 | 0.013 | 0.013 | 0.002 | 0.293 | 0.293 | 7.2E-08 |
| A6 | 1 | 0.301 | 0.140 | 0.137 | 0.002 | 0.136 | 0.292 | 7.2E-08 |
| A7 | 1 | | 0.140 | 0.137 | 0.136 | 0.136 | 0.1361 | 7.2E-08 |

**Key:** ACS-AC Source, LVP-Low Voltage Panel, STS-Static Transfer Switch, UPS-Uninterruptible Power Supply, SDT-Step Down Transformer, SP-Sub Panel, JB-Junction Box, ITD-Information Technology Device.

In architecture A2, the RI index results (see Table 5.5) shows that the redundant AC Sources still have the highest values. However, the cost to add another AC Source is very high. Besides that, the STS (the second highest value in Table 5.5) is a component that cannot be replicated separately. Therefore, architecture A3 (see Figure 5.4 (c)) corresponds to architecture A2 with redundant UPS components (i.e., the next item

from RI index list with the highest value).

Similarly, architecture A4 (see Figure 5.4 (d)) is created from architecture A3 adopting redundant transformers. Architecture A5 (Figure 5.4 (e)) is defined applying redundancy to the Junction Box of A4. Architecture A6 (see Figure 5.4 (f)) is conceived with redundancy applied to the low voltage panel. In such architecture, the STS component that switches the AC Sources was eliminated. Finally, architecture A7 (see Figure 5.4 (g)) corresponds to architecture A1 with all components replicated.

### 5.2.2 Results

Table 5.6 summarizes the results separately for each infrastructure, in which: *Architec* represents the architectures, *Avail(%) (9s)* is the availability level ($A$) with the respective number of nines (-$log$[1 - A/100]), *Exergy(GJ)* is the exergy destroyed, and *Cost(USD)* is the acquisition and operational costs in U.S. dollars. This study adopted the period of one year.

**Table 5.6:** Summary results.

| Architec | Avail(%) (9s) | Exergy(GJ) | Cost(USD) |
|---|---|---|---|
| A1 | 99.8117 (2.725) | 1996.09 | 647984.73 |
| A2 | 99.9904 (4.017) | 2088.94 | 667482.52 |
| A3 | 99.9902 (4.012) | 2178.66 | 741023.44 |
| A4 | 99.9913 (4.061) | 2178.68 | 741579.18 |
| A5 | 99.9919 (4.094) | 2178.70 | 741732.68 |
| A6 | 99.9957 (4.376) | 2089.05 | 738411.95 |
| A7 | 99.9996 (5.450) | 1999.84 | 735104.79 |

As expected, the availability increases when more redundant devices were considered in the architectures. However, in some cases the addition of a component may require other device. Therefore, the availability result of A3 is slightly smaller than A2 due the fact that A3 considers an additional UPS and a STS. On the other hand, the expressive availability increase from A6 to A7 can be explained by the absence of STS devices in A7.

As expected, the availability increases when more redundant devices were considered in the architectures. However, as previously presented in Table 5.5, STS has a significant impact in the system availability. For instance, the availability result of A3 is slightly smaller than A2 due the fact that A3 considers an additional UPS and a STS. Similarly, the expressive availability increase from A6 to A7 can be explained by the absence of STS devices in A7.

Figure 5.5 depicts a graphical comparison between the availability, cost and exergy destroyed for each power infrastructure architecture. The baseline corresponds to the architecture A1. As the reader should note, architecture A7 is an interesting option, since its cost is similar to the baseline (less than 15% more expensive) and provides the highest availability (i.e., twice the value of A1). Therefore, the exergy destroyed was smaller than the value estimated for the architectures A2 to A6.

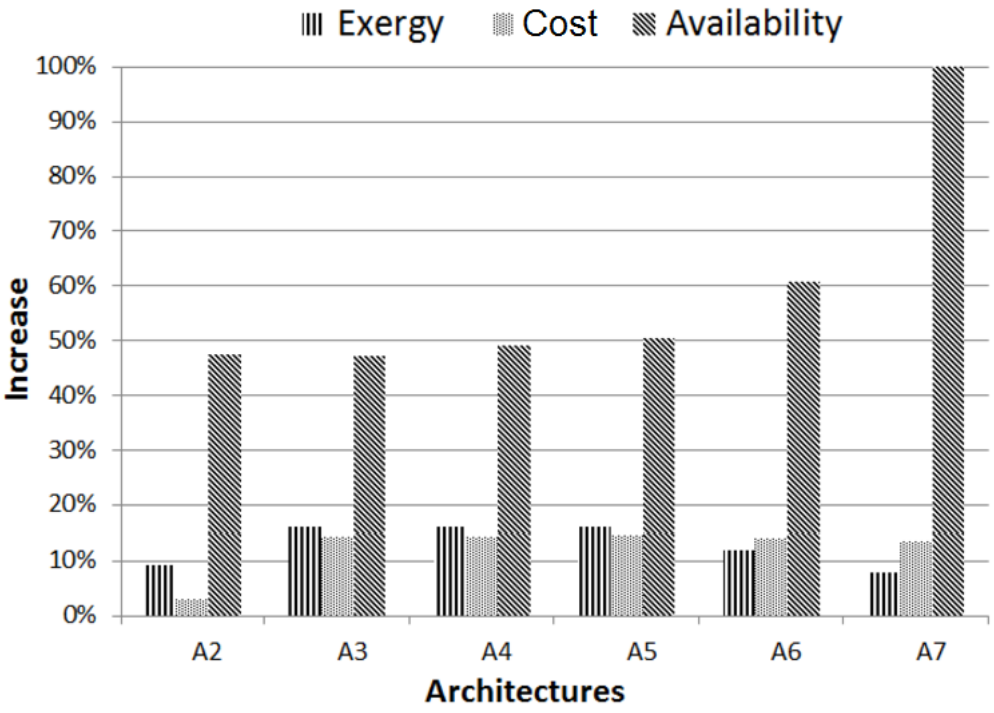


**Figure 5.5:** Comparing Exergy, Availability and Cost.

## 5.3 CASE STUDY III

The exergy destroyed is a metric useful for quantifying the efficiency of a component or architecture in relation to the energy consumption. This metric may be not enough to estimate the environmental impact of identical data center architectures located in different places around the world. This comparison may be assessed by estimating the greenhouse gas emissions such as $CO_2$.

The main goal of this case study is to present a comparative study of real-world data center architectures (from HP Labs Palo Alto, U.S. [64]) assessing dependability as well as environmental impact and operational energy cost associated to the energetic mix of U.S. and Brazil. $CO_2$ emissions (and exergy consumption) related to these energetic mix are considered as a metric to the quantification of environmental impacts. Additionally, this case study also illustrates the applicability of the RCI index, which relates the RI

index with the acquisition cost of the devices that compose the data center architectures under analysis.

Table 5.7 shows the energetic mixes adopted in Brazil [71] and U.S. [72]. Brazil electric energy is mostly generated through hydroelectric power plants (around 80%). Whereas in the U.S., coal power plants represent more than 46% of the energy used. The table also presents the $CO_2$ emissions (in $g$ per kWh of energy consumed) of the four energy sources most used in Brazil and U.S.. Additionally, Table 5.8 presents the energy price (considering the energetic mix) for end-users from Brazil (São Paulo) and U.S.(Palo Alto) [73]. Similarly, it shows the $CO_2$ emissions generated according to the energetic mix in Brazil and U.S.

**Table 5.7:** Electric energy price in Brazil and U.S. and $CO_2$ emissions.

| Energy Source | U.S.(%) | BRA(%) | Price U.S.(USD) | Price BRA(USD) | $CO_2$ Em.(g/kWh) |
|---|---|---|---|---|---|
| Coal | 46.18 | 3.675 | 0.08250 | 0.2759 | 950 |
| Natural Gas | 22.65 | 11.025 | 0.06292 | 0.2529 | 515 |
| Nuclear | 20.35 | 2.5 | 0.20735 | 0.2628 | 150 |
| Hydroelectric | 6.44 | 80.3 | 0.16159 | 0.2710 | 20 |

**Table 5.8:** Electric Energy Price and $CO_2$ Emissions

| | Energy Price (USD) | $CO_2$ Emissions (g/kWh) |
|---|---|---|
| Brazil | 0.2689 | 114.3603 |
| U.S. | 0.1102 | 614.0666 |

The following subsections detail the data center architectures as well as present the correspondent models. Besides, evaluation results are presented.

### 5.3.1 Architectures

This section presents the cooling and power architectures.

**Cooling Architectures**
From the baseline infrastructure (C1) depicted in Figure 5.6, alternative architectures are

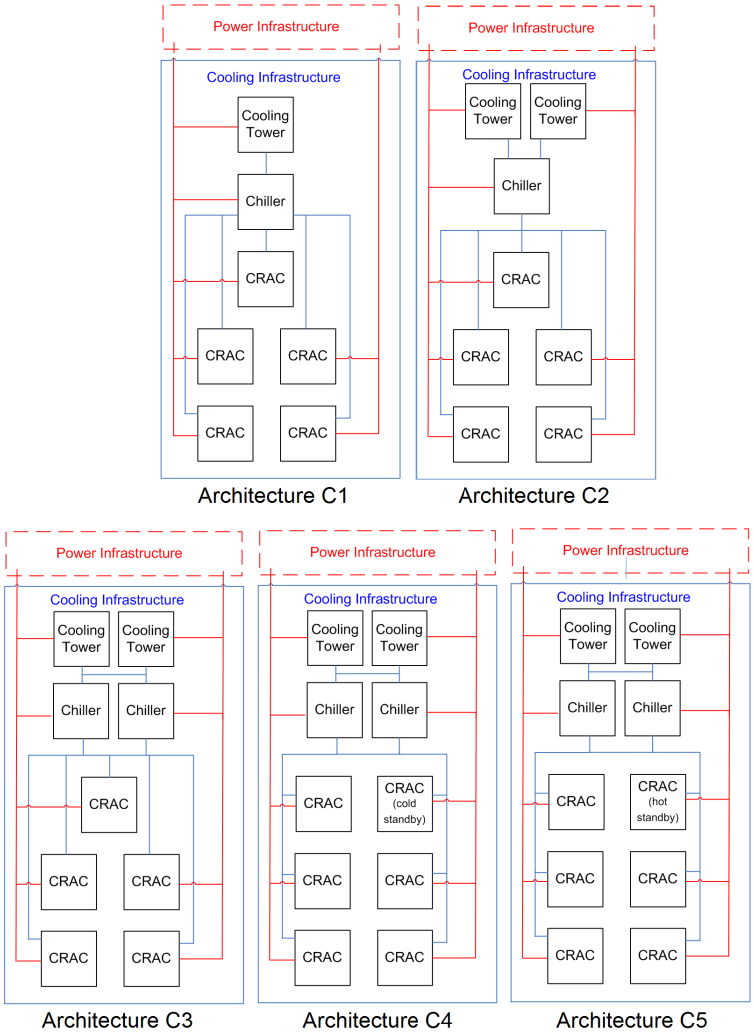


**Figure 5.6:** Cooling infrastructures.

created according to the RCI index. RCI quantifies both acquisition cost and reliability importance of each component to the entire system. In that figure, the cooling infrastructure is represented by the components inside the blue rectangle, in which the red lines represent the power connections and the blue lines delineate the heat connections (e.g. ducts).

The cooling system is responsible, in this work, for extracting 500kW of heat from the data center room. The cooling architecture C1 fails (and, thus, the system) whenever one CRAC, chiller or cooling tower fails.

From the baseline architecture C1, four new architectures are generated considering the $RCI$ index shown in Table 5.9. The reader should notice that the indexes $RI$ and $RCI$ may indicate different component to be replicated. For instance, architecture C2 is proposed by adopting a redundant cooling tower (it has the highest $RCI$ value in

the architecture C1). However, if the $RI$ is adopted, the component suggested to be replicated is the chiller (it has the highest $RI$ value in the architecture C1). Similarly, the architecture C3 is obtained from the architecture C2 by adopting a redundant chiller.

**Table 5.9:** RI and RCI Values of Architectures C1-C3.

| | RI | | | RCI | | |
|---|---|---|---|---|---|---|
| Architecture | Chiller | Cooling Tower | CRAC | Chiller | Cooling Tower | CRAC |
| C1 | 0.22 | 0.19 | 0.17 | 0.17 | 0.18 | 0.14 |
| C2 | 0.10 | 0.08 | 0.30 | 0.23 | 0.05 | 0.19 |
| C3 | 0.08 | 0.26 | 0.23 | 0.09 | 0.07 | 0.27 |

Architectures C4 and C5 (Figure 5.6) use an additional CRAC unit as a cold and hot standby component, respectively. At least one chiller, one cooling tower and five CRAC units are demanded for the cooling data center environment to be working. For each architecture, the availability, sustainability impact and costs are estimated considering one year period.

**Power Architectures**

Four architectures for the power infrastructure have been considered with different configurations, which include redundant components and/or particular arrangements. The baseline architecture (A1) is depicted in Figure 5.7, in which the power infrastructure is delineated by red dashed lines and the cooling infrastructure is represented by two rectangles outside of the red dashed lines. The power infrastructure fails (and thus, the system fails) whenever both paths depicted in Figure 5.7 are not able to provide the power demanded (500 kW) by the IT components (50 racks). The reader should assume a path as a set of redundant interconnected components inside the power infrastructure.

From the baseline architecture A1, alternative architectures are proposed based on the RI and RCI indexes shown in Table 5.10. In the architecture A1, AC sources and UPSs have the highest RI and RCI values. Therefore, it is possible to consider a diesel generator or to replicate the UPS. Architecture A2 (Figure 5.8 (a)) provides replicated UPSs. Besides considering replicated UPSs, the architectures A3(Figure 5.8 (b)) and A4 (Figure 5.8 (c)) adopt one and two electricity generators (in parallel arrangement), respectively, to support the system when both AC sources are not operational. For each architecture, availability, sustainability impact and costs are estimated over a period of one year.

### 5.3.2 Models

We divided the system into two subsystems, the cooling and power infrastructures, which are explained as follows:

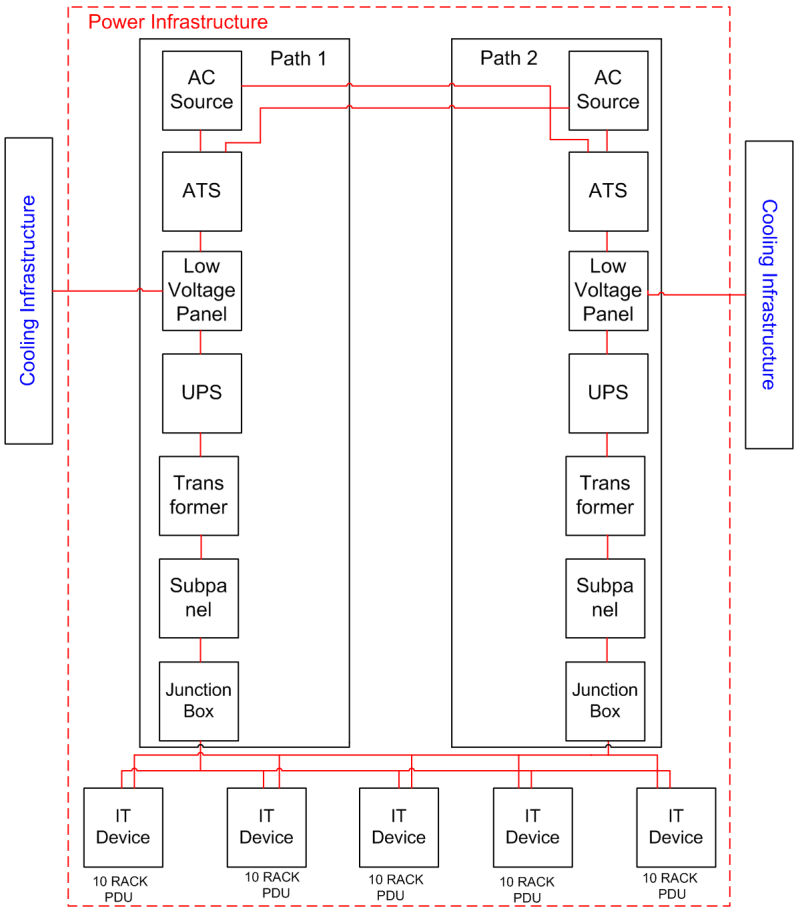


**Figure 5.7:** Power Architecture A1.

**Table 5.10:** RI and RCI Values of Architectures A1-A2.

| | RI | | | | | | RCI | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Arc | ACS | UPS | ATS | SDT | Subpanel | JBox | ACS | UPS | ATS | SDT | Subpanel | JBox |
| A1 | 0.788 | 0.063 | 0.058 | 0.054 | 0.054 | 0.052 | 0.595 | 0.062 | 0.058 | 0.054 | 0.054 | 0.052 |
| A2 | 0.835 | 0.006 | 0.041 | 0.039 | 0.039 | 0.038 | 0.732 | 0.006 | 0.041 | 0.039 | 0.039 | 0.038 |

### Cooling Models

Figure 5.9 shows the EFM for the cooling baseline architecture (C1). In this model, the demanded thermal load corresponds to 500kW and such a value is associated to the source node $S$. The evaluation of EFM provides the sustainability impact and cost issues as well as the verification of the energy flow in each component (Section 3.1). In

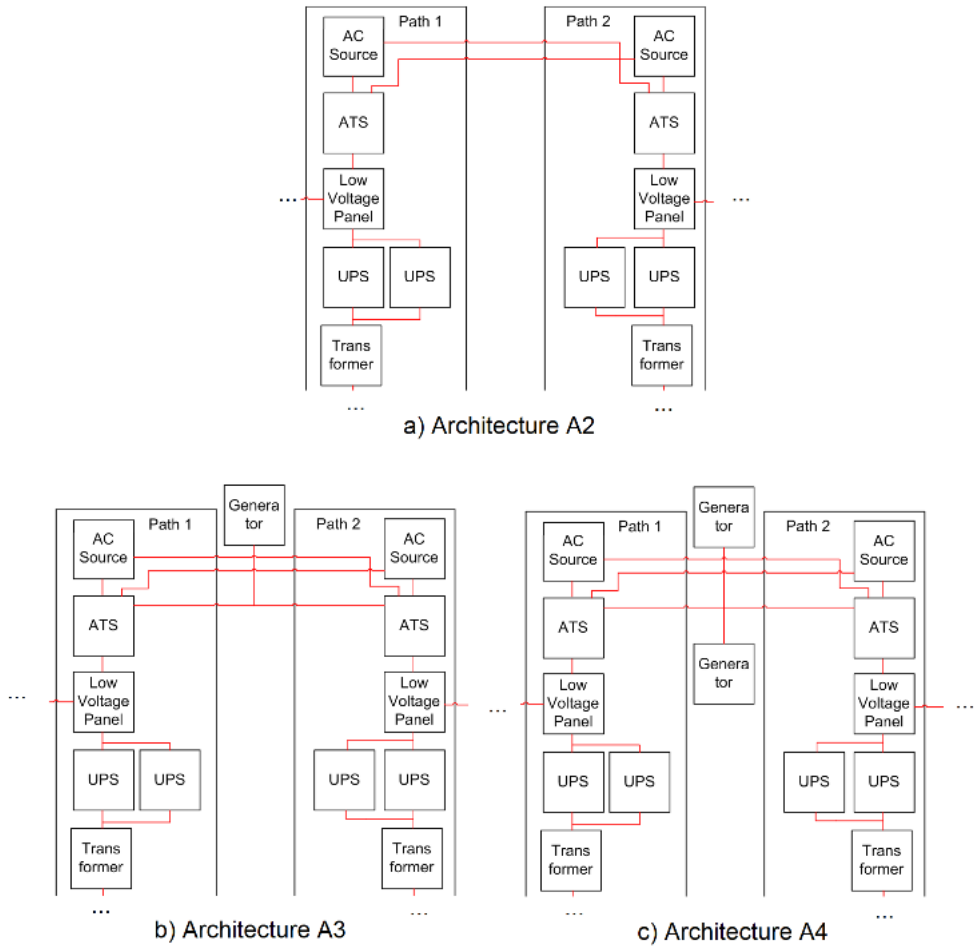


**Figure 5.8:** Power infrastructures: a) Architecture A2; b) Architecture A3; c) Architecture A4.

this system, all components must be operational. A RBD model is adopted to compute the dependability metrics of the architecture C1 in which all components are serially connected. Table 5.11 presents the adopted MTTF (obtained from [67] [74]), MTTR (based on SLA contracts [75] [76]) and acquisition cost [69] values for the cooling devices.

Similarly, other EFMs and dependability models are created to represent cooling architectures C2, C3 and C4. Figure 5.10 depicts the EFM adopted for cooling architecture C5. The difference between architectures C4 and C5 is the CRACs structural organization. In the architecture C4, the CRAC6 is only activated once one of the other CRACs has failed. In architecture C5, all six CRACs are working at the same time as a 5-out-of-6 system.

Architecture C4 corresponds to a cold standby system. Figure 5.11 shows the correspondent SPN model to that system. The availability is computed by the following probability: $P\{\wedge_{i=1}^{5}((\#\text{CRAC}i\_\text{ON}=1)\vee(\#\text{CRAC6\_ON}=1))\ \wedge(\vee_{j=1}^{2}((\#\ \text{Chiller}\ j\_\text{ON}=$

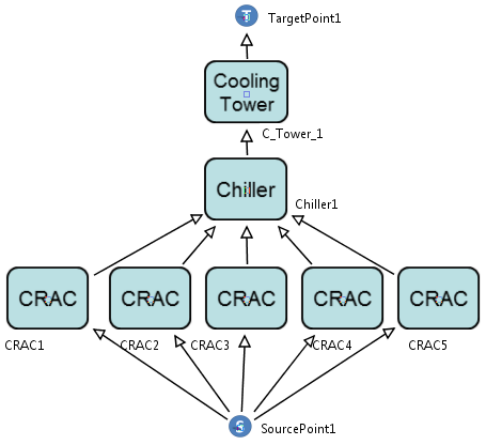


**Figure 5.9:** EFM for the cooling C1.

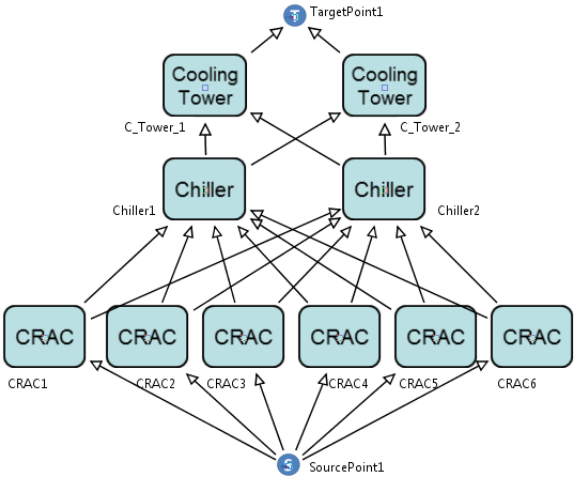


**Figure 5.10:** EFM for the cooling C4.

**Table 5.11:** MTTF, MTTR, acquisition cost (AC) values for cooling devices

| Equipment | MTTF (h) | MTTR (h) | AC (USD) |
|---|---|---|---|
| Chiller | 18,000 | 48 | 40,000.00 |
| Cooling Tower | 24,816 | 48 | 9,000.00 |
| CRAC | 37,059 | 8 | 30,000.00 |

1)∧((#C_Tower_1_ON=1)∨(#C_Tower_2_ ON=1)) ))}.

Additionally, in order to mitigate the complexity of that model, it is important to state

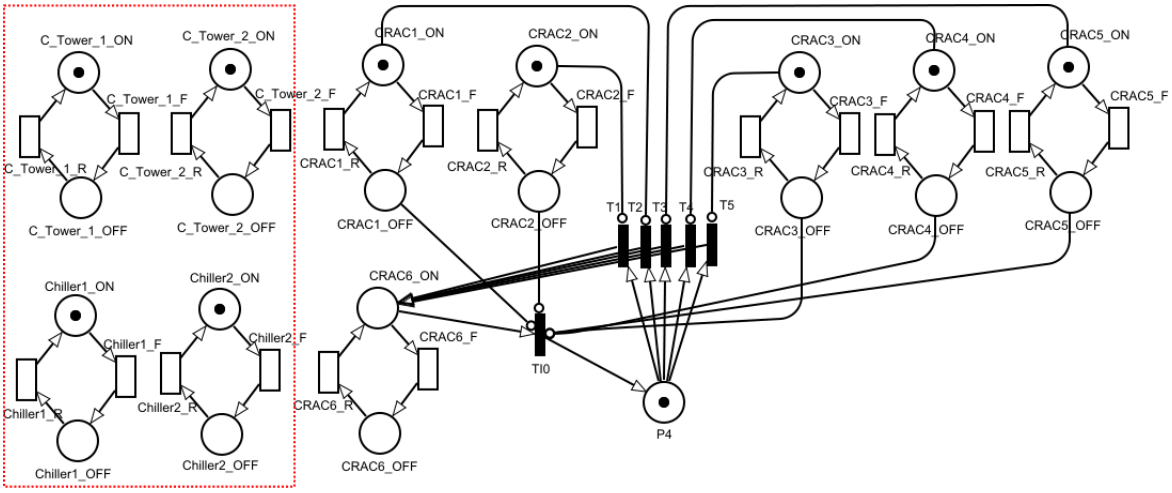


**Figure 5.11:** SPN model for the cooling architecture C4.

that an hierarchical approach is adopted. For instance, the four subnets highlighted on that figure can be evaluated separately through a RBD model considering all those four components in a serial representation. Once the results of both models are obtained, a RBD model with two blocks representing those models in a serial arrangement is created. The RBD evaluation provides the dependability results of the system.

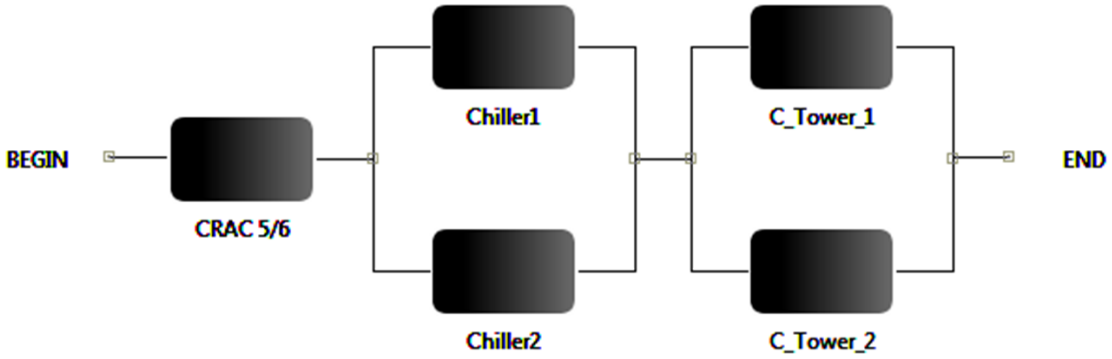


**Figure 5.12:** RBD model for the cooling architecture C5.

Figure 5.12 shows the RBD model correspondent to the cooling architecture C5, in which the block CRAC5/6 represents the 5-out-of-6 CRACs system behavior.

### Power Models

Figure 5.13 depicts the EFM adopted to obtain the sustainability impact results as well

as the cost issues for the power architecture A1. The power demanded (500kW) is associated to the target node $T$ present in the EFM model. The acquisition cost and energetic efficiency values adopted by the EFM is shown in Table 5.12. Following the adopted methodology, systems with no failure dependencies between components have been evaluated through RBD models. For instance, Figure 5.14 depicts the RBD model that represents the architecture A1. The MTTRs and MTTFs adopted in this study is also present in Table 5.12.

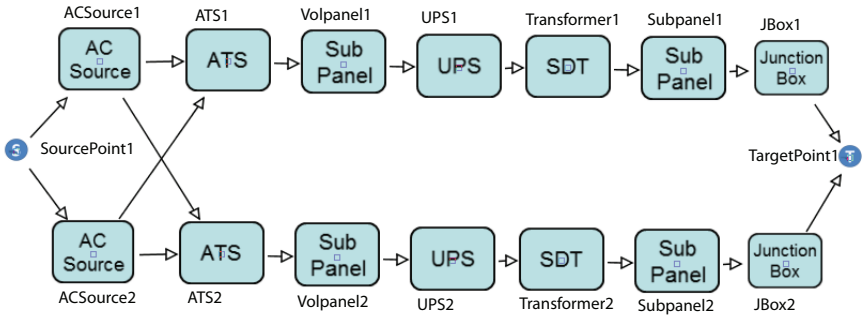


**Figure 5.13:** EFM of Architecture A1.

**Table 5.12:** MTTF, MTTR, acquisition cost and efficiency(Eff) values for power devices

| Equipment | MTTF (hs) | MTTR (hs) | AC (USD) | Eff(%) |
|---|---|---|---|---|
| AC Source | 4,380 | 8 | 15,000.00 | 95.3 |
| Generator | 2,190 | 8 | 66,000.00 | 25.0 |
| Junction box | 5,224,000 | 8 | 150.00 | 99.9 |
| STS | 48,076 | 8 | 800.00 | 99.5 |
| Subpanel | 304,000 | 8 | 200.00 | 99.9 |
| Transformer | 282,581 | 8 | 550.00 | 98.5 |
| UPS | 50,000 | 8 | 60,000.00 | 95.3 |
| Voltage Panel | 304,000 | 8 | 200.00 | 99.9 |

Although not presented here, the power architecture A2 is also modeled considering RBD to obtain the dependability metrics, and the EFM is similar to the previous one. In power architecture A3 and A4, a generator is adopted to improve the availability. In those architectures, the generator is activated when both AC sources are not operational.

Figure 5.15 shows the proposed SPN model to represent the power architecture A3 subsystem composed of one generator and two AC sources. Besides, we assume that

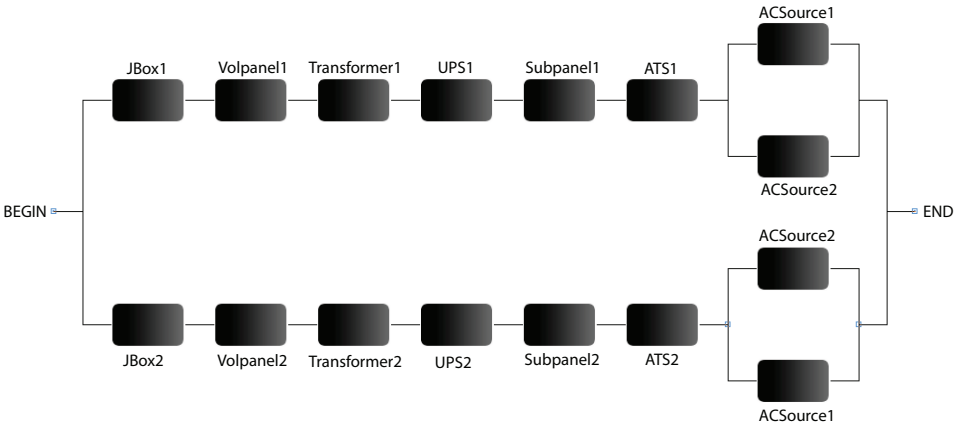


**Figure 5.14:** RBD of Architecture A1.

the UPSs' batteries support the system during the generator activation. The availability of that model is computed by the probability $P\{\#ACSource1\_ON = 1$ OR $\#ACSource2\_ON = 1$ OR $\#Generator\_ON = 1\}$. The other components of the architecture A3 are modeled using RBD (not shown here).

Once the results of both models have been obtained (RBD and SPN), a RBD model with two blocks (considering the results of both models) in a serial arrangement is created. The RBD evaluation provides the dependability results of the power architecture A3.

The adopted MTTF and MTTR values for the power devices were obtained from [67] [74] [75] [76] and are shown in Table 5.12. The results of each sub-system (power and cooling) are considered to obtain the dependability, cost and sustainability results for the whole system.

### 5.3.3 Results

Initially, we present the cooling and power results. Next, we show results for the whole system (cooling and power together).

**Cooling Results.** Table 5.13 summarizes the results for each cooling infrastructure, where *Arc* represents the architecture evaluated; *Avail(%) (9s)* is the availability level ($A$) with the respective number of nines (-$log$[1 - A/100]); *Down(h)* is the system downtime considering the period of 8760 hours (one year); *EX(GJ)* is the operational exergy destroyed in gigajoule; *ACQ(USD)* is the acquisition cost; *OP-U.S.(USD)* and *OP-BRA(USD)* correspond to the operational cost considering the energetic mix adopted in U.S. and Brazil, respectively.

The availability increases and the downtime decreases when redundant components are included. For instance, the availability of architecture C2 is around 10% higher than

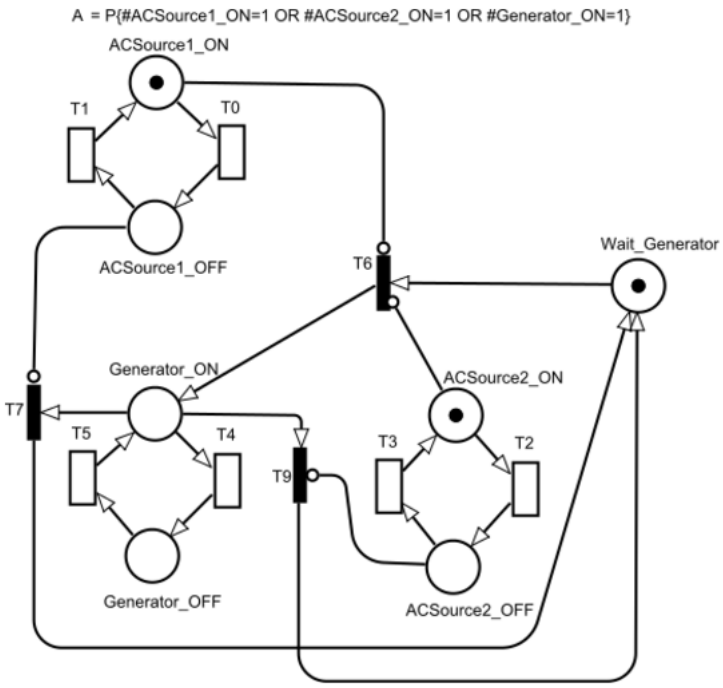


**Figure 5.15:** SPN model for the Architecture A3.

**Table 5.13:** Summary results of Cooling infrastructures.

| Arc | Avail(%) (9s) | Down(h) | EX(GJ) | ACQ(USD) | OP-U.S.(USD) | OP-BRA(USD) |
|---|---|---|---|---|---|---|
| C1 | 99.4341 (2.25) | 49.56 | 9096.52 | 199000.00 | 286158.52 | 698551.39 |
| C2 | 99.6986 (2.52) | 26.40 | 9120.72 | 208000.00 | 286919.60 | 700409.30 |
| C3 | 99.8911 (2.96) | 9.54 | 9138.32 | 248000.00 | 287473.50 | 701761.45 |
| C4 | 99.9989 (4.96) | 0.09 | 9148.19 | 278000.00 | 287783.85 | 702519.05 |
| C5 | 99.9988 (4.94) | 0.10 | 9148.18 | 278000.00 | 287783.71 | 702518.70 |

architecture C1. C4 and C5 are the architectures with the must redundant devices. Considering the availability results of those architecture C4 and C5, a small difference is obtained (4.96 and 4.94, respectively in nines (9s)). Similar exergy destroyed and cost are also obtained for that two architectures.

Figure 5.16 is a graphical comparison between the cost and $CO_2$ emissions considering the adopted energetic mix in U.S. and Brazil. Each cooling infrastructure is compared to the baseline architecture C1 with the energetic mix of U.S. As the reader should note, the Brazilian energetic mix emits around 80% less $CO_2$ in comparison to the U.S mix. However, the operational cost is more expensive in Brazil. For instance, in cooling

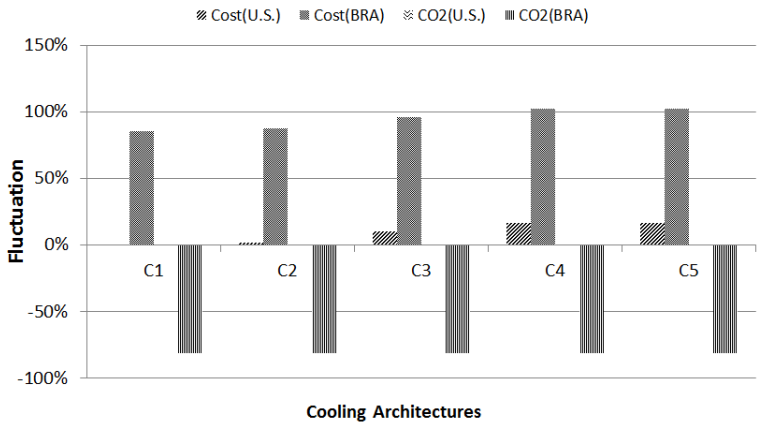


**Figure 5.16:** Cooling results: comparing Cost and $CO_2$ emissions in BRA and U.S.

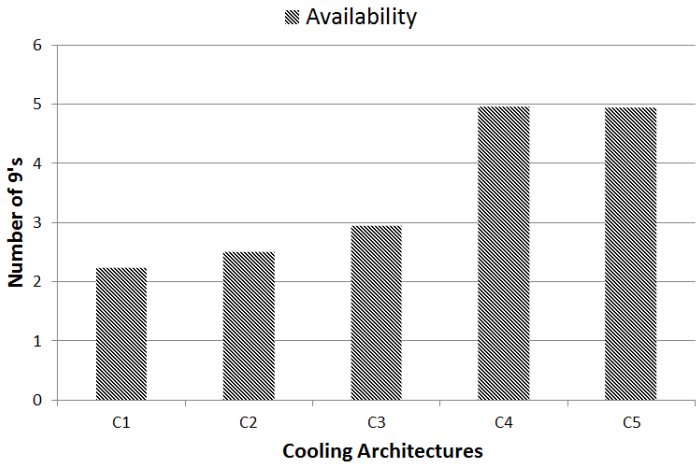


**Figure 5.17:** Cooling Availability Results.

architecture C5, the cost in Brazil is the double of the baseline. Meanwhile, in U.S., the cost is around 30% more expensive than the baseline.

Additionally, Figure 5.17 depicts the availability results of each cooling architecture. The last two graphs show that the cooling architectures C4 and C5 are interesting options due to the fact that both have the higher availabilities and the costs are not much more expensive than the other architectures. Regarding sustainability impact, the adopted energetic mix should be the Brazilian one.

**Power Results.** Table 5.14 presents a summary of results for the power infrastructure architectures. The availability improvement from considering redundant UPSs (architecture A2) is not significant. In architecture A3, the additional generator increased the availability of the power system from 5.5 to 7.5 nines as depicted in Figure 5.18. However, redundant generators (architecture A4) did not increase the availability significantly enough to justify its adoption. Additionally, the operational exergy destroyed in all power

**Table 5.14:** Summary results of Power infrastructures.

| Arc | Avail(%) (9s) | Down(h) | EX(GJ) | ACQ(USD) | OP-U.S.(USD) | OP-BRA(USD) |
|---|---|---|---|---|---|---|
| A1 | 99.9996572 (5.46) | 0.03003 | 1999.8495 | 123800.00 | 550753.55 | 1344463.43 |
| A2 | 99.9996650 (5.47) | 0.02934 | 1999.8496 | 243800.00 | 550753.78 | 1344463.99 |
| A3 | 99.9999970 (7.52) | 0.00026 | 2000.5330 | 309800.00 | 553486.07 | 1351133.88 |
| A4 | 99.9999974 (7.58) | 0.00023 | 2000.5355 | 375800.00 | 549858.38 | 1342278.20 |

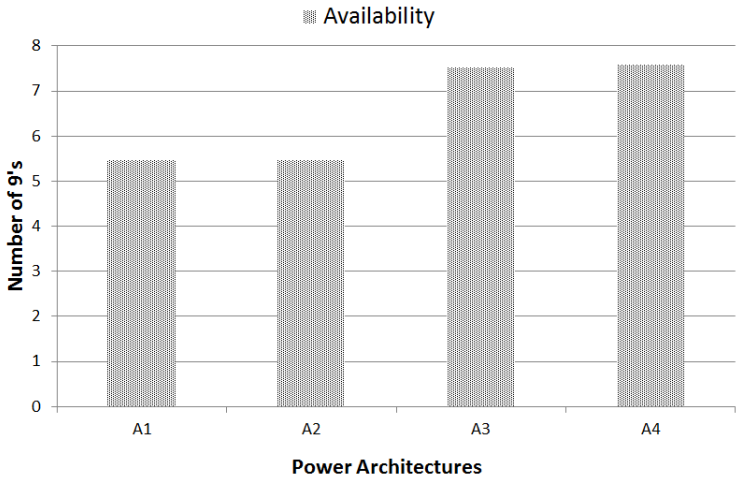


**Figure 5.18:** Power Availability Results

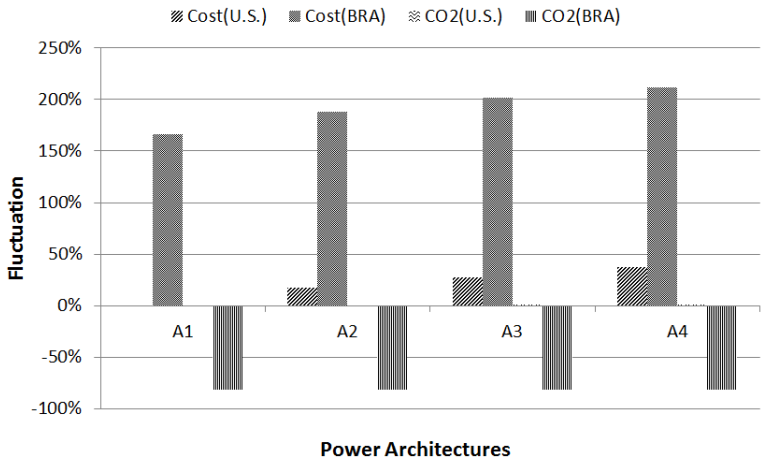


**Figure 5.19:** Power results: comparing Cost and CO2 emissions in BRA and U.S.

infrastructures are similar, meaning that the energetic efficiency is not so different between those architectures.

Figure 5.19 presents a comparison of $CO_2$ emissions as well as the respective cost differences of all power architectures in comparison to the baseline A1, adopting the U.S. energetic mix. Similarly to the results obtained in the cooling infrastructures, the resulting operational energy costs in Brazil are higher than in U.S. However, the Brazilian energetic has a lower impact on the environment. Taking into account the previous results, architectures A3 and A4 are interesting due to their high availability and their relatively small difference in operational costs compared to A1 and A2.

**System Results (Power and Cooling).** Considering the power architectures (A3 and A4) and cooling infrastructures (C4 and C5), a RBD model is adopted to combine those architectures to represent the system final model. The RBD is represented by two blocks (one for the power infrastructure and one for the cooling infrastructure) connected serially.

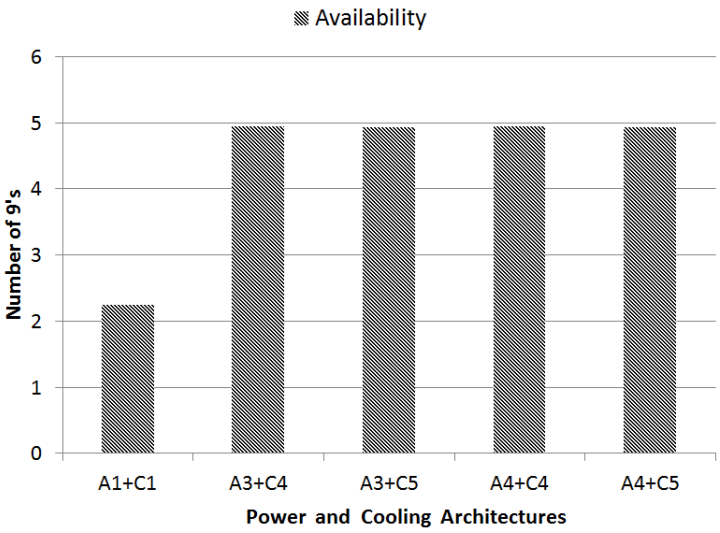


**Figure 5.20:** Power and Cooling Availability Results.

Figure 5.20 presents the power and cooling availability results combined. In that figure, $A4+C4$ means that the power architecture A4 and cooling architecture C4 are combined. Similarly, other results of combinations are also provided. There is no significant difference between the availability results obtained considering the power architectures A3 and A4 combined with the cooling infrastructures C4 and C5.

Figure 5.21 presents a comparison of the cost and $CO_2$ emissions considering the energetic mix adopted in Brazil and U.S. The adopted baseline is the power architecture A1 combined to the cooling infrastructure C1, adopting the energetic mix of U.S.

Analyzing the previous results, it is possible to notice that to increase the availability from 2.3 nines to close to 5 nines, the cost has increased around 140% considering the energetic mix of Brazil (and around 30% adopting the U.S. energetic mix) in all architectures. An interesting option is the power architecture A3 combined to the cooling

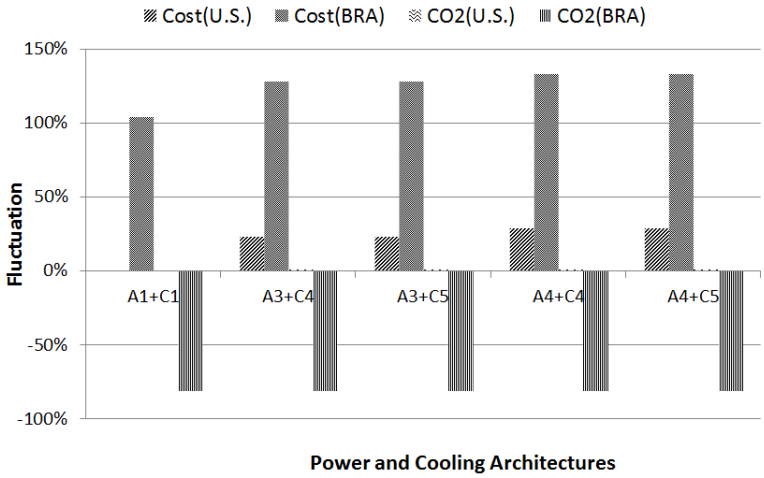


**Figure 5.21:** Power and Cooling results: comparing Cost and CO2 emissions in BRA and U.S.

infrastructure C4 due to the fact that the acquisition cost of the power architecture A3 is cheaper than A4 (only one generator is adopted). Additionally, the availability of architecture A3 is a little bit higher than A4 (4.95 against 4.93 nines).

Considering the fact that the demanded energy for each architecture analyzed (power and cooling) is established by the IT device needs, the energy consumption difference between each architecture is determined by the energetic efficiency of each architecture. Therefore, the operational costs and $CO_2$ emissions do not vary much across the different architectures.

## 5.4 CASE STUDY IV

The main goal of this study is to illustrate the proposed optimization algorithm applied on data center infrastructures. This method improves the results obtained by RBD, SPN and EFM models through the selection of the appropriate data center devices from a list of candidate components. This case study is subdivided into two parts. In part I, the list is automatically generated through a benchmark and, in part II, that list is defined by the data center designer. The evaluation of all possible combinations from the list of candidate components provides the optimal result. However, this evaluation is very time consuming. The main goal of this experiment is then to show that the adopted optimization algorithm represents an interesting option that provides results close to the optimal ones in a reduced time. Therefore, this case study compares the results obtained by the optimization algorithm to the results obtained evaluating all possible scenarios.

This case study is subdivided into two parts. The main goal of the first part is to illustrate the applicability of the proposed methodology to optimize data center power infrastructures in relation to cost (acquisition and operational costs) and dependability

issues. It is important to state that once reducing the operational cost, which is impacted by the energy consumption, the sustainability impact is also improved. In the first part of this case study, a benchmark is adopted to create a list of candidate elements through a random process that is further explained in the following sections. In the second part, the list of candidate elements is fixed (defined by the designer).

### 5.4.1 Part I

**Architectures**

Five data center power infrastructures [68] have been considered (see Figure 5.22) with increasing redundancy degree, such that each successive architecture has an additional component duplicated. For each architecture, we estimate: (i) availability, (ii) the sustainability impact and (iii) the cost.

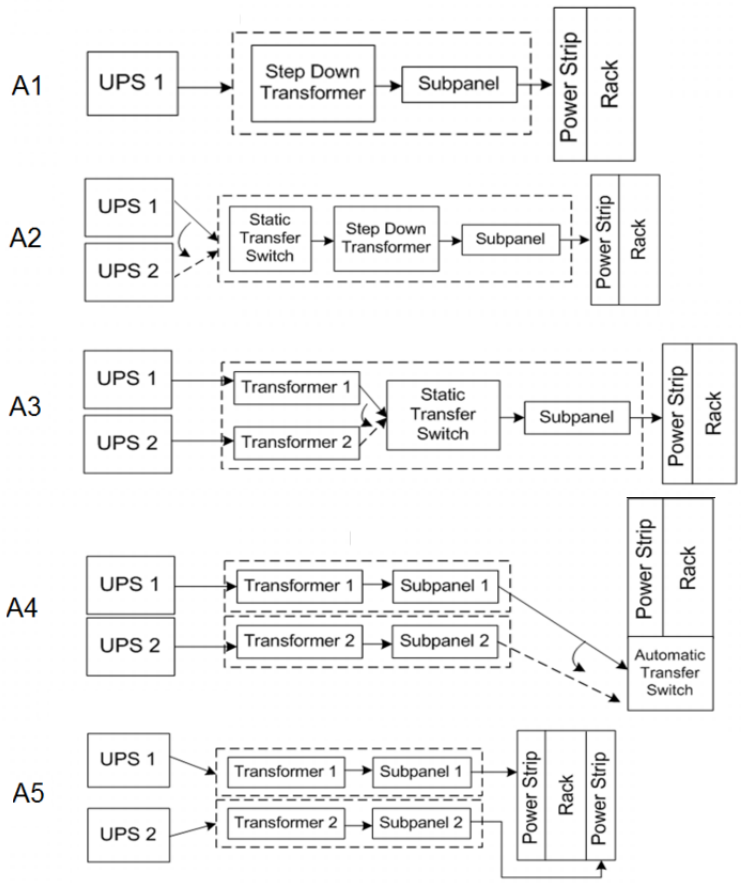


**Figure 5.22:** Data Center Power Architectures.

**Models**

Figure 5.23 (a) depicts the RBD model adopted to quantify the system dependability for the architecture A3 while respecting the system restrictions quantified by the EFM (Figure 5.23 (b)). The EFM is also adopted to estimate the sustainability impact and cost of the data center power infrastructure. The adopted MTTFs values for the power devices of the RBD model were obtained from [67] [77] [78] and are shown in Table 5.15. The MTTRs were constants in 8 hours for each component [76]. Table 5.15 also presents the adopted acquisition cost and efficiency for each device modeled through EFM. Besides, it is important to state that these power infrastructures have a power demand of 10kW (value associated to the target node of the EFM).

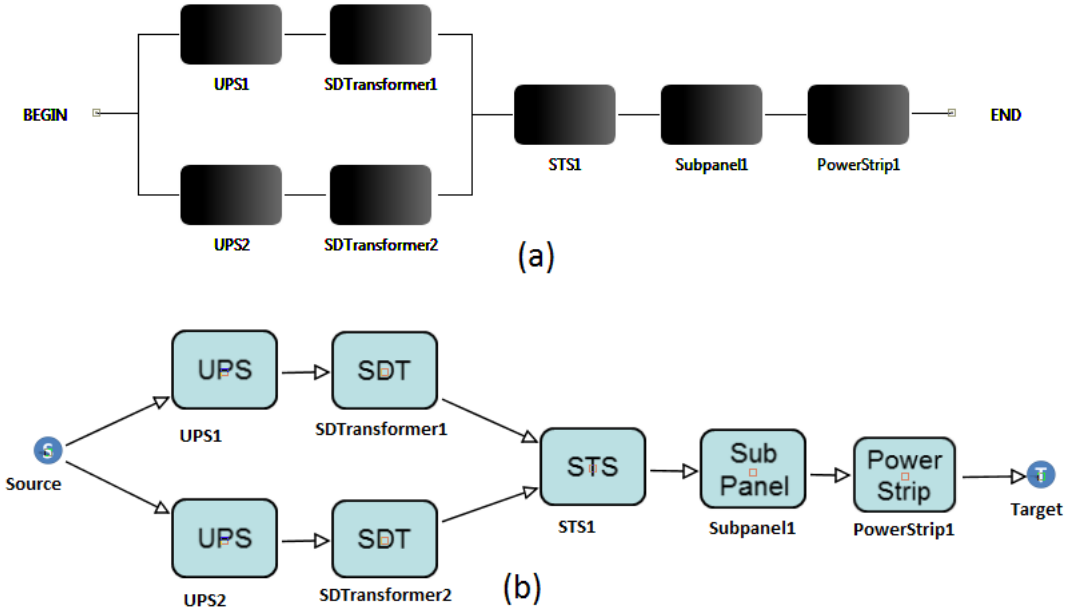


**Figure 5.23:** (a) RBD and (b) EFM models for architecture A3.

### Benchmark

A benchmark was adopted to create a list of candidate elements through a random process. This process considered a range of values for each parameter (MTTF, acquisition cost (AC) and efficiency (Eff)) on each type of component as shown in Table 5.15. In this particular case study, five elements were added to the list for each type of component. Once the list of candidate elements was created, the optimization process was started.

### Results

Table 5.16 presents a summary results that compare the evaluation of all possible combinations ($ALL$) against the optimization algorithm ($Opt.$). In the table, $Avail$ is the column with the availability results, $9s$ is the availability level ($A$) in number of nines ($-log[1 - A/100]$), $Down$ is the downtime which is represented in hours as well as the associated cost, $Acq$ is the acquisition cost, $Op$ is the operational cost, $Tot$ is the total

**Table 5.15:** Benchmark range values for the devices.

| Equipment | MTTF (hs) | AC (USD) | Eff(%) |
|---|---|---|---|
| UPS | [25000, 75000] | [11250, 18750] | [90.535, 99.9] |
| STS | [24038, 72114] | [600, 1000] | [94.525, 99.9] |
| Subpanel | [152000, 456000] | [150, 250] | [94.905, 99.9] |
| Step Down Transformer | [141290.5, 423871.5] | [412.50, 687.50] | [93.575, 99.9] |
| Power Strip | [115111.8, 345335.3] | [150, 250] | [94.525, 99.9] |

cost, *Exergy* corresponds to the results for the exergy destroyed, *Sys Eff* is the system efficiency, *Obj Func* is the objective function, which is represented by the mean of $m$ executions as well as by the smaller result obtained, and *Diff.* is the difference which is represented by the relation of the results obtained through the evaluation of all scenarios by the results obtained from the optimization process.

The optimization method was executed $m$ times considering different seeds that randomly generate the list of candidate elements. In addition, for each seed, the method was repeated $m$ times. The $m$ value has to be big enough to guarantee the confidence level of 95% and not so huge that may increase the execution time. Therefore, *m=30* was adopted. The reader should remember that the goal of this case study is to minimize the objective function and thus, the smaller value between the $m$ executions is the result of the optimization algorithm. Additionally, it is important to state that the optimized results were computed considering 0.2 as the greediness degree ($\beta$ parameter of the optimization method in Section 2.6).

Figure 5.24 depicts the objective function results achieved through the evaluation of all possible scenarios (All) and the results obtained by the optimization method (Opt. Alg.). The reader should notice that both results are quite similar. The paired-t test [70] was conducted to analyze the evaluation results. Assuming a significance level of 5%, the test resulted in the following confidence interval:[-1004.36, 7881.47]. As 0 (zero) is enclosed by that interval, there is no evidence to reject the hypothesis of equivalence between the values generated by the evaluation of all possible scenarios and the results obtained by the optimization method.

Additionally, Figure 5.25 presents a comparison between the execution time demanded for both cases, in which it was possible to observe that the time needed to conduct the evaluation of all possible combinations has increased when more complex system were analyzed. However, this behavior was not observed during the optimization algorithm results.

The main goal of this experiment is to compare the results obtained by an optimization algorithm to the results obtained by the evaluation of all possible scenarios. For instance, architecture A1 is composed of a UPS, a step down transformer, a subpanel and a power

**Table 5.16:** Comparing optimal results (*ALL*) with optimization algorithm (*Opt.*) estimates.

| Arch | | Avail (9s) (Mean) | Down (hs) (Mean) | Down (USD) (Mean) | Acq (USD) (Mean) | Op (USD) (Mean) | Tot (USD) (Mean) | Exergy (GJ) (Mean) | Sys Eff (Mean) | Obj Func (Mean) | Obj Func | Runtime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *ALL* | 0.9998 (3.73) | 1.63 | 326321.75 | 4440.85 | 6897.92 | 11338.77 | 68.0993 | 0.7061 | 337660.52 | 309658.32 | 2.3s |
| A1 | *Opt.* | 0.9998 (3.71) | 1.73 | 345356.08 | 4587.37 | 6897.79 | 11485.17 | 68.0971 | 0.7028 | 356841.25 | 309658.32 | 2.0s |
| | Diff | 1.00 (1.01) | 0.94 | 0.94 | 0.97 | 1.00 | 0.99 | 1.00 | 1.00 | 0.95 | 1.00 | 1.20 |
| | *ALL* | 0.9998111 (3.72) | 1.66 | 331007.34 | 8318.95 | 7406.41 | 15725.36 | 84.74142761 | 0.6551 | 346732.70 | 323597.83 | 39.4s |
| A2 | *Opt.* | 0.9997947 (3.69) | 1.80 | 359753.32 | 7783.50 | 7377.51 | 15161.01 | 83.79825129 | 0.6581 | 374914.34 | 327659.81 | 4.0s |
| | Diff | 1.00 (1.01) | 0.92 | 0.92 | 1.07 | 1.00 | 1.04 | 1.01 | 1.00 | 0.92 | 0.99 | 9.64 |
| | *ALL* | 0.9998289 (3.77) | 1.50 | 299701.16 | 8855.68 | 7053.88 | 15909.55 | 73.20110047 | 0.6875 | 315610.72 | 284483.23 | 3min27s |
| A3 | *Opt.* | 0.9998088 (3.72) | 1.67 | 334940.61 | 8274.06 | 7112.76 | 15386.82 | 75.13123792 | 0.6814 | 350327.43 | 293539.03 | 5.4s |
| | Diff | 1.00 (1.01) | 0.89 | 0.89 | 1.07 | 0.99 | 1.03 | 0.97 | 1.01 | 0.90 | 0.97 | 38.61 |
| | *ALL* | 0.9998462 (3.82) | 1.35 | 269424.51 | 8974.10 | 6584.39 | 15558.49 | 57.83336742 | 0.7371 | 284982.99 | 254187.48 | 18min55s |
| A4 | *Opt.* | 0.9998382 (3.80) | 1.42 | 283448.68 | 8403.39 | 6572.17 | 14975.55 | 57.43458247 | 0.7390 | 298424.23 | 257552.05 | 6.6s |
| | Diff | 1.00 (1.01) | 0.95 | 0.95 | 1.07 | 1.00 | 1.04 | 1.01 | 1.00 | 0.95 | 0.99 | 172.34 |
| | *ALL* | 0.99999994 (7.22) | 0.00056 | 112.80 | 7413.00 | 5736.13 | 13149.13 | 30.04785664 | 0.8426 | 13261.93 | 12230.32 | 19min39s |
| A5 | *Opt.* | 0.9999999 (7.20) | 0.00058 | 115.42 | 7772.58 | 5796.62 | 13569.20 | 32.0276621 | 0.8335 | 13684.62 | 12940.75 | 8.1s |
| | Diff | 1.00 (1.00) | 0.98 | 0.98 | 0.95 | 0.99 | 0.97 | 0.94 | 1.01 | 0.97 | 0.95 | 146.23 |

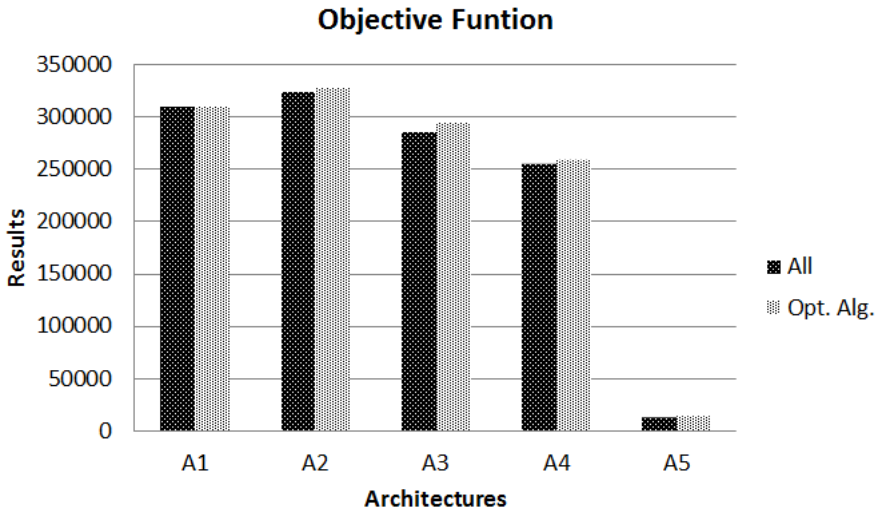


**Figure 5.24:** Objective Function.

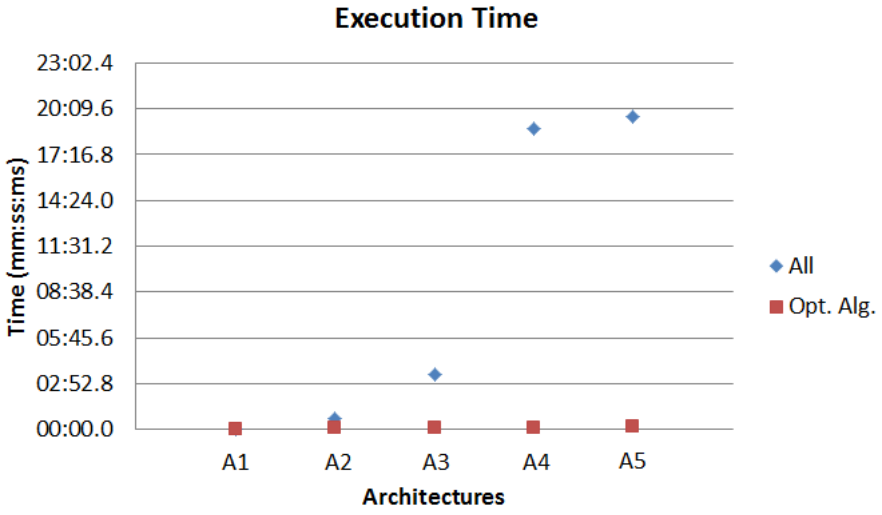


**Figure 5.25:** Execution Time.

strip. Considering that the list of candidates elements for each component is composed of five different devices (e.g., with different MTTF, efficiency and acquisition cost), there are $5^4$ possible scenarios to be evaluated for the architecture A1. In addition, the optimization method requires a reduced time to be executed in comparison to the evaluation that computes the optimal results.

Significant results were obtained with the optimization algorithm that required a reduced time to be executed in comparison to the evaluation of all scenarios. This runtime difference increases when more sophisticated architectures are modeled. For instance, to compute the optimal result of the architecture A3 more than three minutes was de-

manded, however only five seconds was needed to obtain the result taking into account the optimization algorithm. In the architecture A5, the optimal evaluation demanded 146 more time to be executed than the results achieved through the optimization algorithm.

Additionally, it is important to stress that no significant difference was observed between the results obtained adopting the optimization algorithm and the evaluation of all possible scenarios. Considering the objective function results, the higher difference was 5% achieved in architecture A5. In A2 and A4, only 1% of difference was observed. Besides, for architecture A1, the exact value was obtained for both the optimal and the optimization algorithm results.

This case study has shown that the proposed optimization technique can be adopted to obtain results close to the optimal ones without the need to execute all the scenarios available. In addition, this case study also provided sustainability, dependability and cost solutions whilst at the same time respecting the system restrictions as quantified by the EFM.

### 5.4.2 Part II

The main goal of this part is also to compare the optimal results to the ones obtained through the optimization algorithm. However, in this section a fixed list of candidate components was adopted. Table 5.17 presents this list with the MTTFs, acquisition cost (AC) and efficiency (Eff) considered for each device. The reader should notice that three different diesel generators and UPSs were adopted.

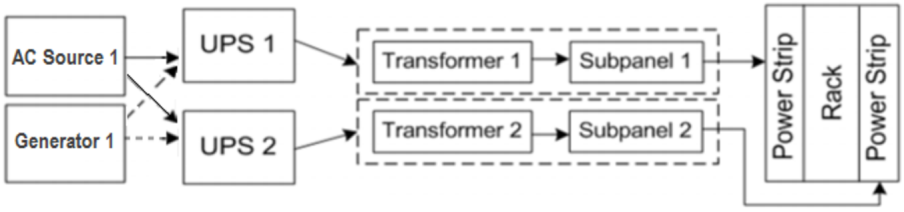


**Figure 5.26:** A6 - Data Center Power Architecture.

**Models**

Figure 5.26 depicts the data center power architecture (A6) adopted in this study. This architecture corresponds to the previous architecture A5 (Figure 5.22) with an additional diesel generator and an AC source. Assume that, in this particular case, the batteries available on the UPSs are not enough to support the system during the activation time demanded by the generator device. Besides, the generator is only activated if the AC source does not provide the demanded power. Therefore, SPN is adopted to evaluate the availability. Figure 5.27 shows the SPN availability model for the architecture A6. The availability is described by the probability expression

**Table 5.17:** MTTF, acquisition cost(AC) and efficiency(Eff) values.

| Equipment | MTTF (hs) | AC (USD) | Eff(%) |
|---|---|---|---|
| AC Source | 4380 | 15000.00 | 95.3 |
| Generator | 2500 | 6000.00 | 25.0 |
| Generator | 3500 | 7500.00 | 30.0 |
| Generator | 4000 | 9000.00 | 32.0 |
| UPS | 50000 | 15000.00 | 95.3 |
| UPS | 60000 | 16500.00 | 94.0 |
| UPS | 75000 | 16000.00 | 95.3 |
| Step Down Transformer | 282581 | 550.00 | 98.5 |
| Subpanel | 304000 | 200.00 | 99.9 |
| Power Strip | 230223.512 | 200.00 | 99.5 |

$P\{((\#ACSource0\_ON = 1)OR(\#Generator1\_ON = 1))\ AND(((\#UPS\_5kVA2\_ON = 1)AND(\#SDT4\_ON = 1)\ AND(\#Subpanel6\_ON = 1)AND(\#PowerStrip8\_\ ON = 1))OR((\#UPS\_5kVA3\_ON = 1)AND(\#SDT5\ \_ON = 1)AND(\#Subpanel7\_ON = 1)AND(\#Power\ Strip9\_ON = 1)))\}$, where $\#p$ denotes the marking of place $p$.

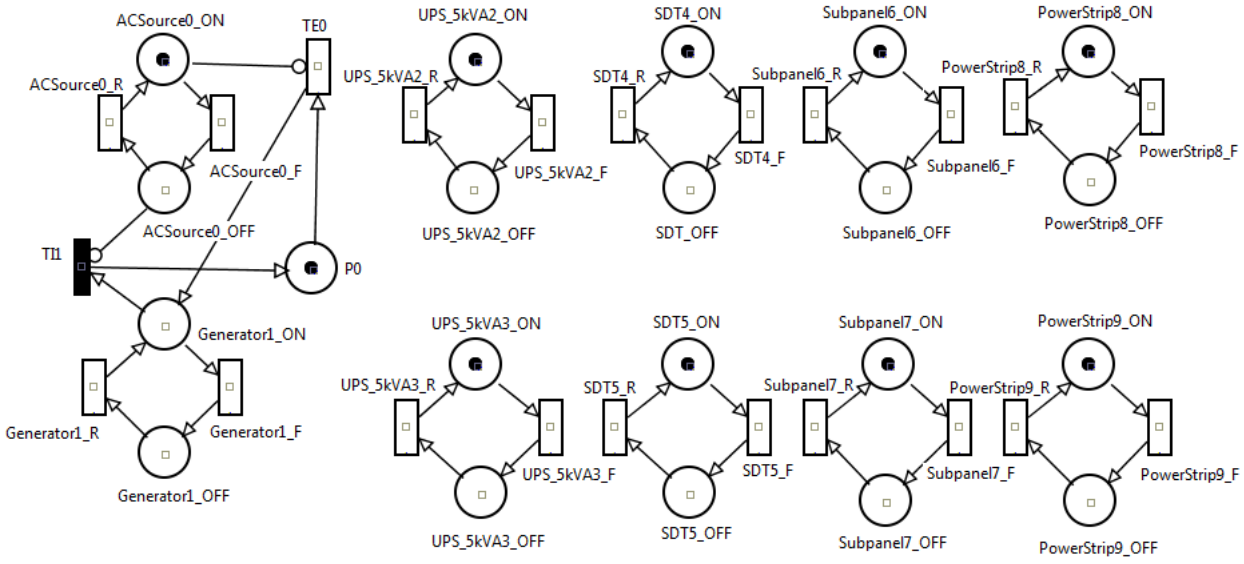


**Figure 5.27:** SPN of A6.

Similarly to the Part I, EFM is adopted to estimate the sustainability and cost issues

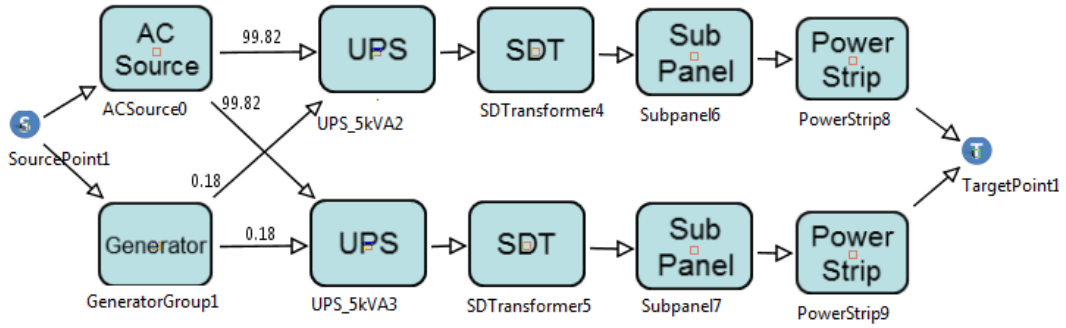


**Figure 5.28:** EFM of A6.

without overstepping the power constraints of each device. Figure 5.28 depicts the EFM for the architecture A6.

**Results**

Table 5.18 presents the summary results of the Part II of this case study. From this table, it is possible to observe that the results for the object function are quite similar in the optimization method and the evaluation of all scenarios. However, the runtime needed for executing all the scenarios was over 2.8 times higher than the one spent by the optimization technique. In addition, the reader should remember that the list of candidate components was fixed (e.g., three generators and tree UPSs) reducing the number of possible combinations. Besides, it is important to stress that in case the list of candidate components increases, the difference between the runtime spent by the optimization algorithm in comparison to the execution of all possible scenarios tends to grow in such way that may turn impossible to perform the analysis of all possible scenarios.

## 5.5 CASE STUDY V

This study focuses on conducting the previous optimization technique in a real-world data center power architecture (from HP Labs Palo Alto, U.S. [64]) which assesses the dependability as well as environmental impact and operational energy cost. In this case, the number of scenarios to be analyzed through all possible combinations is huge. Therefore, the optimization method was adopted. The following subsections present the data center power architecture as well as show the corresponding models and results.

**Architecture**

Figure 5.29 depicts the data center power architecture modeled. This architecture supplies energy for 50 racks, which are represented in this figure by only five racks (each one representing 10 racks). The power infrastructure fails (and consequently the entire system) when neither of the paths depicted in the figure are able to provide the 500kW

**Table 5.18:** Comparing optimal results (ALL) with Optimization Algorithm (Opt. Alg.) estimates.

| Arch | | Avail (9s) | Down (hs) | Down (USD) | Acq (USD) | Op (USD) | Tot (USD) | Exergy (GJ) | Sys Eff | Obj Func | Runtime |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A6 | ALL | 0.99994178 (4.2349) | 0.5100 | 101995.61 | 52900.00 | 10890.64 | 63790.64 | 47.1384 | 0.8847 | 165786.26 | 5min03s |
| | Opt. Alg. | 0.9999427 (4.2420) | 0.5017 | 100352.80 | 54400.00 | 10878.26 | 65278.26 | 44.3915 | 0.8858 | 165631.06 | 1min47s |
| | Diff | 0.9999991 (0.9983) | 1.0164 | 1.0164 | 0.9724 | 1.0011 | 0.9772 | 1.0619 | 0.9989 | 1.00093 | 2.83 |

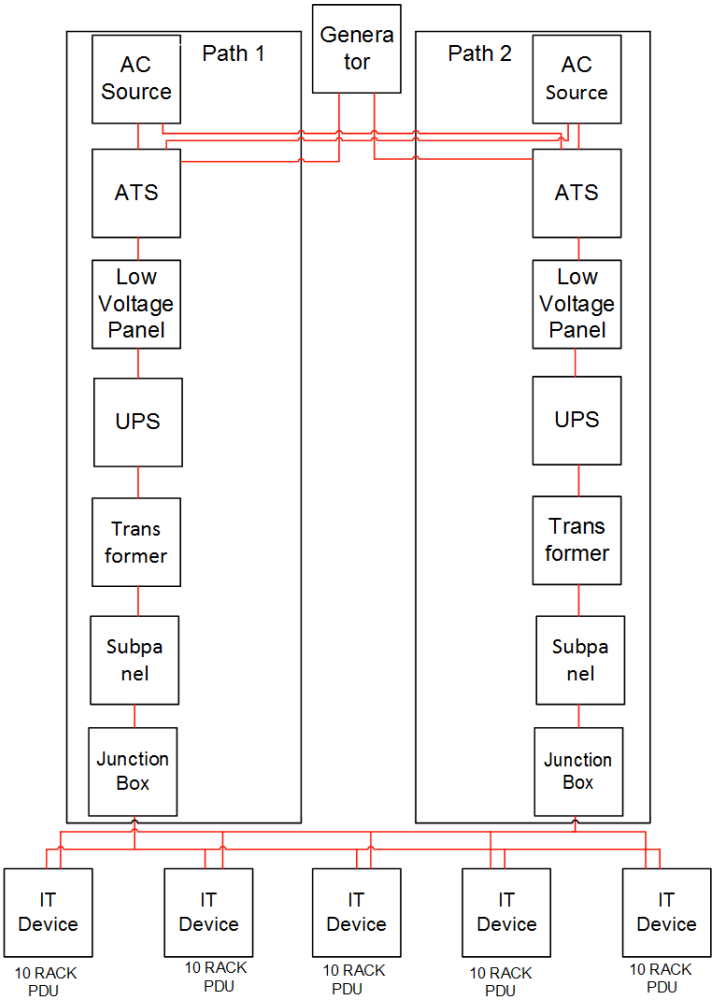


**Figure 5.29:** Data Center Power Infrastructure.

of power required by the racks of IT components. In this study a path is defined as a set of interconnected redundant components within the power infrastructure. In this architecture, the diesel generator is only activated once both AC sources are not able to provide the required power.

### Models

Figure 5.30 illustrates the EFM correspondent to the previous power architecture. According to the adopted methodology, the dependability analysis of this system should be performed through an RBD since neither active redundancies nor state-dependent interactions between the components of the system exist. For instance, Figure 5.31 shows the RBD model that represents the power system under analysis. It is important to state that in this particular system, we assumed that batteries present in the UPSs support the system during the activation time needed to start the diesel generator since both AC

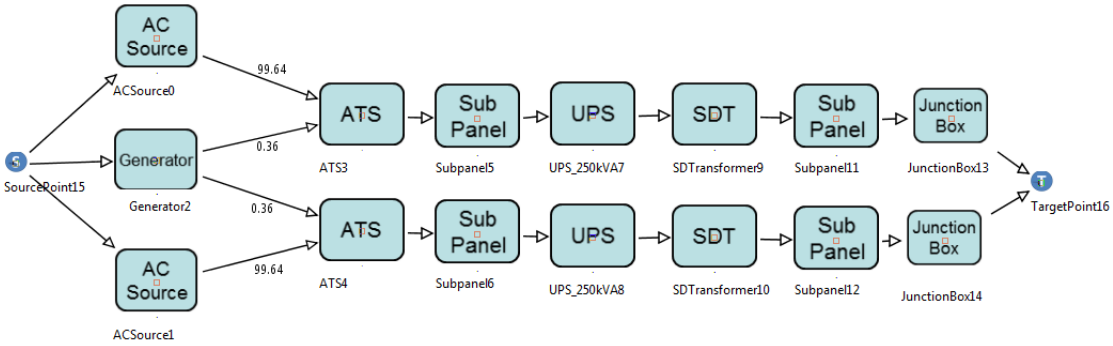


**Figure 5.30:** EFM for the power infrastructure.

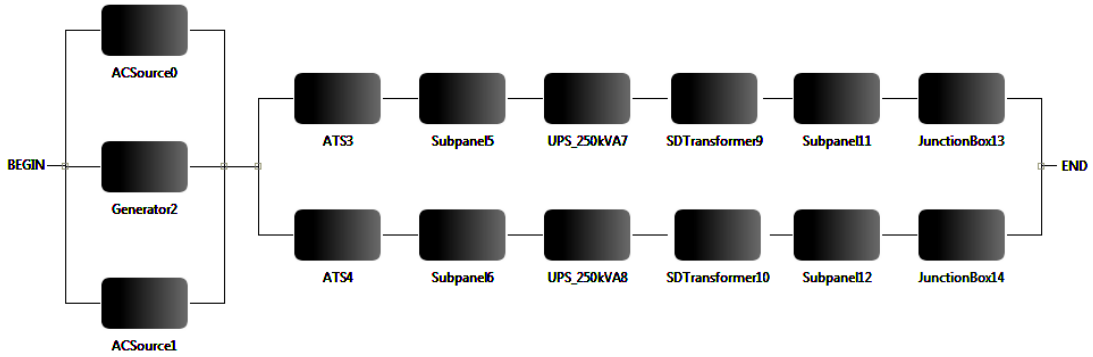


**Figure 5.31:** RBD for the power infrastructure.

sources are not properly working. Therefore, the dependability analysis of this system may be conducted by RBD as shown in Figure 5.31. The reader should remember that the list of candidate devices was created in a random process similar to the case study I (Part I). Table 5.19 shows the range of values adopted for the MTTF, acquisition cost and efficiency to the creation of the list of candidate components.

### Results

Table 5.20 is a summary results of the evaluated power infrastructure architecture. The availability, cost and exergy destroyed were 6.67 (in number of 9s), 862,820.06 USD and 6311.18 kJ for this system taking into account the components according to the algorithm that optimizes the objective function. This experiment has shown that following the adopted methodology, we are able to perform the integrated evaluation and optimization of dependability, cost and sustainability issues.

**Table 5.19:** Benchmark range values for the devices.

| Equipment | MTTF (hs) | AC (USD) | Eff(%) |
|---|---|---|---|
| AC Source | [2190, 6570] | [11250, 18750] | [90.535, 99.9] |
| Generator | [1095, 3285] | [45000, 75000] | [23.75, 26.25] |
| Junction box | [2612000, 7836000] | [112.50, 187.50] | [94.905, 99.9] |
| UPS | [25000, 75000] | [11250, 18750] | [90.535, 99.9] |
| ATS | [24038, 72114] | [600, 1000] | [94.525, 99.9] |
| Subpanel | [152000, 456000] | [150, 250] | [94.905, 99.9] |
| Step Down Transformer | [141290.5, 423871.5] | [412.50, 687.50] | [93.575, 99.9] |
| Voltage Panel | [152000, 456000] | [150, 250] | [94.905, 99.9] |

## 5.6 SUMMARY

This chapter presented the case studies adopted to illustrate the applicability of the proposed set of models and the methodology for performing the integrated evaluation of dependability, sustainability and cost issues on data center power and cooling infrastructures. For instance, real-world data center architectures from HP Labs Palo Alto were analyzed. In addition, an analysis that assessed dependability as well as environmental impact and operational energy cost associated to the energetic mix of U.S. and Brazil were conducted. Finally, studies were presented to show that the proposed optimization algorithm applied on data center architectures has provided interesting results close to the optimal ones.

**Table 5.20:** Summary Results.

| | Results |
|---|---|
| Availability (%) | 0.999999781 |
| 9s | 6.67 |
| Downtime (hs) | 0.0019 |
| Downtime Cost (USD) | 384.17 |
| Acquisition Cost (USD) | 211279.68 |
| Operational Cost (USD) | 651540.38 |
| Total Cost (USD) | 862820.06 |
| Exergy Destroyed | 6311.18 |
| System Efficiency (%) | 0.74 |
| Func Objective | 863204.23 |

CHAPTER 6

# RELATED WORK

A reasonable definition of sustainability and sustainable development is provided by the World Commission on Environment and Development (WCED) [79] which states "... development that meets the needs of the present without compromising the ability of the future generations to meet their needs and aspirations".

In the last few years, some research has been done on reliability analysis of data center systems and a subset has also considered sustainability impact and cost issues as well as some optimization techniques. The following sections presents those researches.

## 6.1 DEPENDABILITY

Reliability (which encompasses both the durability of the data and its availability for access) corresponds to the primary property that data center users desire [12]. Regarding data center power architectures, voltage interruption or transient voltage may require a complete system restart or components repair. In both cases, the system mean time to repair (MTTR) as well as the system reliability is impacted. Additionally, redundancies on components to increase system reliability are costly. A design engineer needs to avoid over-sizing electrical equipment, since it may waste not only power consumption but also investment.

Wiboonrat [80] proposes an approach based on RBD to analyze the dependability of data center power distribution topologies. Besides, Wiboonrat [81] extended that approach by proposing a model for data center design considering risk assessment and adopting a method for identifying single point of failure in data center power infrastructures. Although both papers estimated the dependability of data center power architectures, nothing is comment regarding cost and environmental impact issues.

Robidoux [82] proposes the Dynamic RBD (DRBD) model, an extension to RBD, which supports reliability analysis of systems with dependence relationships. The additional blocks (in relation to RBD) to model dependence, turned the DRBD model complex. The DRBD model is automatically converted to a coloured Petri net (CPN) model in order to perform behavior properties analysis which may certify the correctness of the model [83]. Different from this work, this thesis considers SPN models for modeling systems with dependence relationships. Therefore, we believe that instead of proposing complex models, an interesting alternative would be to model the system directly using

CPN or any other formalism (e.g., SPN) able to perform dependability analysis as well as to model dependencies between components.

Wei [84] presents a hierarchical method to model and analyze a virtual data center (VDC). The approach combines the advances of both RBD and General SPN (GSPN) for quantifying availability and reliability. Data center power and cooling architectures are not the focus of their research and the proposed models are specific to modeling VDC.

An approach for reliability evaluation and risk analysis of dynamic process systems using stochastic Petri nets is proposed in [85]. The approach consists of connecting interrelated nets and evaluating them adopting Monte Carlo simulation technique. As a case study, that approach evaluates a process control system, in which the system behavior is modeled and, next, qualitative/quantitative results are deduced. This work focuses in dependability evaluation of complex system with dependencies, however, the associated cost and sustainability impact are not mentioned.

Vilkomir [86] estimates the availability of fault-tolerant computer systems by classifying processor failures into several types. A segregated failure model is proposed and compared to their previously correspondent Markov chain model [87] considering a cluster system as a case study. Adopting the proposed failure model, the influence of each failure type on the system availability may be computed. This work is not focused in estimating sustainability issues integrated with availability and cost of data center architectures.

## 6.2 SUSTAINABILITY

Gmach [88] proposes an approach to manage data center's energy supply. The approach estimates the power usage in a data center based on the average CPU utilization across all servers. The paper does not provide any integrated comparison between the cost, sustainability impact and availability of data center architectures.

In [89], the authors propose a platform for the evaluation of smart data centers taking into account cooling, power and IT components. A coefficient of performance comprising those components is proposed to measure the overall efficiency of energy flow. The paper is not focused in presenting dependability and cost results.

Patterson [90] evaluates the impact of ambient temperature on energy efficiency. The conducted analysis indicates the existence of an optimum temperature for data center operation that depends on several factors. Herold [91] describes opportunities for energy integration in the context of combined cooling, heating and power systems. However, both works are not focused in an integrated approach considering cost, sustainability and dependability issues.

The practice of monitoring, measuring, and reporting environmental impact represents a challenge for many industries. Authors in [10] illustrates the importance of the Power Usage Efficiency (PUE), which corresponds to the ratio of total facility power by IT Equipment power. This work focuses only sustainability issues.

Chang [92] proposes a method for estimating the exergy consumption during the raw material extraction, manufacturing, operational, transport and disposal phases. Many assumptions are used to take into account the entire device lifecycle, which may introduce systematic errors.

## 6.3 COST

Urgaonkar [93] proposes an approach that focuses in energy storage devices to reduce the average time of the electricity bill related to data centers. The main idea is to avoid using the energy from the company provider during the day peaks of energy use (which corresponds the time that the energy is more expensive). During that time, the energy stored through UPS is adopted. The author focused only in the cost, nothing was commented regarding availability or sustainability issues.

Bianchini [94] [95] presents an optimization-based framework that considers multi-data center services, service-level agreements (SLAs) as well as energy consumption in order to manage the cost and carbon foot print of those data centers. The enormous electricity consumptions in US are translate into large carbon footprints, since most of the electricity are produced by burning coal, a carbon intensive energy production approach. The approach is focused in cost and environmental impacts of data centers, nothing is considered regarding dependability.

## 6.4 OPTIMIZATION

Studies have been conducted in the last few years which attempt to evaluate dependability and cost of data center infrastructures whilst others present optimization techniques.

In [96], an optimization-based framework, named REWIRE, is proposed to optimize data center network architectures by the adoption of an algorithm that improves the network bandwidth and minimize the end-to-end latency while respecting the user defined constraints. The work does not focus on the power and cooling infrastructures.

Wang [97] proposes an optimization algorithm, named CARPO, to optimize the energy consumption of data center networks. The algorithm focuses on reducing the number of activated switches of data center networks through the dynamically consolidation of the network traffic. The goal is to reduce the data flows into a small set of links and, then, shuts down unused network devices for energy savings. In order to accomplish this, the data flow is modeled as an optimal flow assignment problem in which the traffic constraints must be satisfied while the energy consumption should be minimized. This work focuses on the IT network without mentioning the other data center architectures (e.g., power and cooling).

Tham and Ang [98] adopt Continuous-Time Markov-Chains (CTMC) models to compute the data center cluster availability. Additionally, an approach based on Bellman

optimality equation [99] and greedy-based algorithms is considered to improve the system availability by determining the optimal secondary machine policy used. The approach starts considering an arbitrary assignment policy. Then, the Markov Decision Process (MDP) [100] graph (generated from the CTMC) with the Bellman optimally equation are adopted for solving the state values of the current policy. To optimize the system availability, other policies are generated by the greedy-based algorithms and evaluated in the optimization approach. The authors have not the focus in analyzing data center power and cooling architectures.

**Table 6.1:** Summary of the related works.

| Related Work | Dependability | Sustainability | Cost | Optimization |
|---|---|---|---|---|
| [80], [81] | RBD | - | - | - |
| [82], [83] | RBD, CPN | - | - | - |
| [84] | RBD, SPN | - | - | - |
| [85] | SPN | - | - | - |
| [86] | MC | - | - | - |
| [88] | - | Power usage | - | - |
| [89] | - | Energy flow | - | - |
| [90], [91] | - | Energy | - | - |
| [10] | - | PUE | - | - |
| [92] | - | Exergy | - | - |
| [93] | - | - | Yes | - |
| [94], [95] | - | $CO_2$ | Yes | Yes |
| [96] | - | - | - | Yes |
| [97] | - | Energy | - | Yes |
| [98] | CTMC | - | - | Yes |
| This Work | RBD, SPN | Energy, Exergy, $CO_2$ | Yes | Yes |

## 6.5 CONCLUDING REMARK

Table 6.1 presents a summary of the previous work main goals in comparison to the work conducted in this thesis. The compared proprieties are dependability, sustainability and

cost of data center systems. In dependability column of the table, we show the formalism (e.g., Markov Chain (MC), SPN, RBD and CTMC) adopted for computing those metrics for each particular related work. Similarly, the sustainability column presents the metrics (e.g., energy, exergy, $CO_2$) that were considered for estimating the environmental impact on each work. In addition, some works also have performed studies to optimize those issues of data center infrastructures.

Most related works focus in estimating only dependability [80][81][82][83][84][85][86], sustainability [88][89][90][10][92] or cost [93] of data center architectures. Other works have performed efforts to compute and optimize dependability [98] or sustainability [97]. Moreover, works also have focused in an integrated optimization techniques that improve sustainability and cost issues [94][95] of data centers.

The mentioned works, in a way or another, aim at solving or dealing with aspects and issues related to dependability, energy consumption or optimization. Nevertheless, none of them proposes an integrated strategy to sponsor the design, by supporting modeling, estimation and optimization of dependability, cost and sustainability issues for data center infrastructures.

CHAPTER 7

# CONCLUSION

With increased dependence on Internet services (e.g., cloud computing), data center availability has become a more serious concern. For companies that heavily depend on Internet for their operations, service outages can be very expensive, easily running into millions of dollars per hour. A widely used design principle in fault-tolerance is to introduce redundancy to enhance availability. However, since redundancy leads to additional use of resources and energy, it is expected to have a negative impact on sustainability and associated cost.

At present stage, data center designers do not have many mechanisms to support the integrated sustainability, cost and dependability evaluation of data center infrastructures. This work aims at reducing this gap by proposing models (supported by the developed environment ASTRO/Mercury) that can be adopted to estimate those issues before implementing data center architectures. The adopted methodology considers an integrated approach based on dependability, cost and energy evaluation.

This work presented a comparative study of real-world data center architectures assessing dependability as well as environmental impact and operational energy cost associated to the energetic mix of U.S. and Brazil. To conduct those case study experiments, ASTRO/Mercury environment has been adopted to support the integrated evaluation of sustainability, dependability and cost issues. Additionally, an optimization method is proposed for optimizing the dependability (RBD and SPN) and EFM models adopted for the integrated evaluation of sustainability, dependability and cost issues on data center power and cooling systems.

The following sections describe the main contributions of this thesis and proposes possible extensions as future directions.

## 7.1 CONTRIBUTIONS

This thesis presented a set of formal models for analyzing data center infrastructures considering dependability, sustainability and cost issues. Besides, the adopted methodology proposes that the system should be evaluated piecewisely to allow the composition of simpler models representing a data center infrastructure appropriately. Nevertheless, the reader should note that the models as well as implementations on the evaluation environment (ASTRO, Mercury and optimization module) represent contributions of this

thesis. For a better visualization, the contributions are detailed below according to each activity:

- **Modeling:** This thesis presented a set of formal models to allow the integrated evaluation of dependability, sustainability and cost issues on data center power and cooling infrastructures. The EFM is proposed to estimate sustainability impact in terms of operational exergy consumption as well as to compute the associated costs while respecting the power constraints of each device. Additionally, dependability models such as RBD and SPN are adopted to evaluate dependability issues.

- **Methodology:** The conceived methodology represents a contribution which proposes that the system under analysis may be evaluated piecewisely. Moreover, the methodology also takes the advantage of both RBD and SPN formalism. For instance, to compute dependability metrics (e.g., availability, reliability and downtime) RBD or SPN may be adopted. RBD considers closed-form equations, the results are exact and usually obtained faster than using SPN simulation. However, to represent complex systems, particularly those based on dynamic redundancy methods, RBD models experience drawbacks concerning the thorough handling of failures and repairing dependencies. In this case, SPN models are more indicate as previously mentioned in this thesis.

- **Algorithms:** In this thesis, algorithms that traverse the EFM are proposed to perform the power verifications as well as to compute data centers costs (acquisition and operational) and sustainability impacts. Additionally, an optimization GRASP-based algorithm was proposed for improving the results obtained by RBD, SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center architecture.

- **Evaluation environment:** This thesis extends the Mercury environment besides considering RBD, SPN and CTMC formalisms, to consider the EFM model. It is important to state that the views supported by the ASTRO tool can be converted to the fundamental models (SPN, EFM, CTMC and RBD) allowing non-specialized users to create models as well as obtain the sustainability, dependability and cost estimates as previously explained (Chapter 3). Therefore, the gap that data center designers has to perform the integrated evaluation of sustainability, dependability and cost is reduced. In addition, an optimization module was developed for allowing the implementation of algorithms (e.g., GRASP-based algorithm and PLDA) that improves the fundamental models.

- **Data center architectures:** From base line data center power and cooling architectures, other architectures are proposed focusing in the availability/reliability increase and trying to reduce the sustainability impact as well as the associated costs. In order to accomplish this, RI and RCI are adopted to identify the components that most impact the availability and reliability of the system. Additionally,

optimization techniques have been adopted to optimize the results obtained through the model evaluation.

## 7.2 FUTURE WORKS

Although this thesis tackles some issues regarding sustainability, dependability and cost of data center architectures, there are many possibilities to improve and extend the current work. The following items summarize some possibilities:

- In this work, the sustainability impact has been estimated considering the exergy consumption during the operational phase of the data center. Although in some works [101] [102] [103] [64] we already had performed studies considering embedded (see Appendix A) and operational exergy consumption, this thesis focuses in the operational exergy due to the fact that the estimative of the amount of energy consumed during the embedded phase (e.g., material extraction and device manufacturing) seems to be obscure. A possible extension is to consider the sustainability impact during the whole cycle (Life-cycle assessment - LCA) of the equipment. Therefore, a study related how to obtain those energy consumption estimative during the embedded phase must be performed.

- The exergy consumption is known as an interesting option to measure how efficient is the energy consumption on each device. However, this work can be extended through the adoption of other metrics such as the Green Computing ones (e.g., Global-warming potential (GWP), Power usage effectiveness (PUE), Data center infrastructure efficiency (DCIE), Data Centre Efficiency (DCE), Data Center Performance Efficiency (DCPE) and Corporate Average Data center Efficiency (CADE)).

- Data center dependability values may be impacted by some factors, named hardened computing, such as the temperature and humidity variation in the data center room. There can be no doubt that electrical devices suffer degradations (e.g., availability and reliability reductions) due to the rise on the temperature and/or humidity. Therefore, the relation between data center cooling infrastructure and the IT system can be analyzed. For instance, models that relates such relation can be proposed to estimate the impact of the temperature variation on the availability of IT system. In addition, the relation can be studied to be computed statically and/or dynamically. By dynamic, the reader should understand that the relation between the cooling infrastructure and the IT system can be monitored for simulating the impact of the temperature on the availability of the IT devices on site.

- This thesis may be also extended to consider maintenance policies [104] as well as different service level agreements (SLAs). For instance, the presented operational cost of this thesis does not consider any cost associated to maintenance policies as well as fines associated in case the contracted maintenance company does not

provide the required availability. Therefore, a possible future direction is to extend the proposed models in this work to couple with maintenance policies. In addition, the equation adopted in this work to compute the operational cost may include the associated cost of maintenance policies, which improves the operational cost.

- To improve ASTRO's data center views in order to let them be closer to the engineering representation of the power and cooling data center architectures. Additionally, the conversion process from ASTRO's views to the fundamental models (EFM, SPN, RBD and CTMC) supported by the Mercury can also be improved. For instance, to consider the k-out-of-n functionality, in which the data center designer will not be an expertise to deal directly with the RBD and SPN formalism to implement that type of system. In addition, the tool may also be extended to consider other formalisms besides RBD, SPN, CTMC and EFM (e.g., fault tree).

- To consider other optimization methods (e.g., Ford Fulkerson algorithm [105] and multi-objective optimization methods[106]) to optimize the evaluation results obtained by the dependability and EFM models. A Ford Fulkerson based algorithm may be proposed for dynamically improve the energy flow through the EFM model by optimizing the weights on the edges. This can be performed in cooperation to dependability models, in which the current state of the model could change the actual state of the EFM. It is important to state that once changed the weights on the edges and considering devices and paths with different energetic efficiency, different energy consumption results will be obtained and so, an optimal solution may be found. Multi-objective optimization analysis can be conducted to improve the integrated evaluation of dependability, sustainability and cost issues presented in this work. Additionally, the results obtained through multi-objective optimization strategies can be compared against the optimization method proposed in this thesis.

- Although power and cooling data center architectures may account for 50% of the energy consumption of data center system, a natural extension of this work is to consider the mentioned methodology to be applied for IT infrastructures. In addition, performance metrics can be studied to try to relate them with the integrated evaluation of dependability, cost and sustainability issues.

## 7.3 SUMMARY

This work proposed a set of formal models for the integrated evaluation of sustainability impact, dependability and cost values on data center power and cooling architectures. These models have the support of the developed evaluation environment which is composed of ASTRO, Mercury and the Optimization module. In addition, the adopted methodology that takes into account the advantages of both RBD and SPN formalism to compute the dependability metrics, and the EFM to estimate the cost and sustainability impacts whilst respecting the power constraints of each device.

Optimization studies were also conducted in this work through the proposed algorithm that is based on GRASP. This algorithm improves the results obtained by RBD, SPN and EFM models through the selection of the appropriate devices from a list of candidate components. This list corresponds to a set of alternative components that may compose the data center architecture. The adopted optimization method is an alternative approach to the evaluation of all possible combinations from the devices of the candidate list. Interesting goals were achieved in which the optimization technique has provided results close to the optimal in a reduced time. Despite the results presented, research on data center has other open issues, which lay down several possibilities for further development of new techniques.

# BIBLIOGRAPHY


[1] U. E. I. Administration-EIA, "Sustainability review - 2010," tech. rep., BP, 2011.

[2] A. W. Hodfes and M. Rahmani, "Fuel sources and carbon dioxide emissions by electric power plants in the united states," tech. rep., University of Florida, 2009.

[3] B. Weihl, E. Teetzel, J. Clidaras, C. Malone, J. Kava, and M. Ryan, "Sustainable data centers," *XRDS*, vol. 17, pp. 8–12, June 2011.

[4] "Microsoft - creating a greener data center." http://www.microsoft.com/presspass/features/2009/apr09/04-02Greendatacenters.mspx, 2009.

[5] R. Harmon and H. Demirkan, "The next wave of sustainable it," *IT Professional*, vol. 13, no. 1, pp. 19–25, 2011.

[6] I. Dincer and M. A. Rosen, *EXERGY: Energy, Environment and Sustainable Development*. Elsevier Science, 2007.

[7] "Technology and policy recommendations and gols for: Reducing carbon dioxide emissions in the energy sector," tech. rep., ASME General Position Statement on, 2009.

[8] K. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. Wiley Interscience Publication, 2nd ed., 2002.

[9] W. Kuo and M. J. Zuo, *Optimal Reliability Modeling - Principles and Applications*. Wiley, 2003.

[10] R. Ellis, D. Perre, A. Latreche, J. Hearnden, L. Gajic, E. Boonstra, and A. Hoxtell, "The green grid energy policy research for data centres: France, germany, the netherlands and the united kingdom," in *The Green Grid: White Paper 25* (V. Smith, ed.), 2009.

[11] E. N. Power, "Energy logic: Reducing data center energy consumption by creating savings that cascade across systems," in *A White Paper from the Experts in Business-Critical Continuity*, 2009.

[12] P. Banerjee, C. Bash, R. Friedrich, P. Goldsack, B. A. Huberman, J. Manley, C. Patel, P. Ranganathan, and A. Veitch., "Everything as a service: Powering the new information economy," *IEEE Computer*, pp. 36–43, March 2011.

[13] A. Venkatraman, "Global census shows datacentre power demand grew 63 percent in 2012." http://www.computerweekly.com/news/2240164589/Datacentre-power-demand-grew-63-in-2012-Global-datacentre-census, 2012.

[14] D. Alger, *Build the best data center facility for your business.* Networking Technology Series, Cisco, 2005.

[15] M. Arregoces and M. Portolani, *Data center fundamentals.* Fundamentals Series, Cisco, 2003.

[16] L. Barroso and et al, "Web search for a planet: The google cluster architecture," 2009.

[17] C. Belady, "In the data center, power and cooling costs more than the it equipment it supports," *Electronics Cooling Magazine*, February 2007.

[18] X. Fan and et al, "Power provisioning for a warehouse-sized computer," 2007.

[19] D. Patterson, "A simple way to estimate the cost of downtime," 2002.

[20] A. J. S. Chandrakant D. Patel, "Cost model for planning, development and operation of a data center, internet systems and storage laboratory.," in *Report of HP Laboratories, Palo Alto*, pp. 1–36, 2005.

[21] A. Avizienis, J. Laprie, and B. Randell, "Fundamental Concepts of Dependability," *Technical Report Series-University of Newcastle upon Tyne Computing Science*, 2001.

[22] J. F. Meyer and W. H. Sanders, "Specification and construction of performability models," 1993.

[23] C. Ebeling, *An introduction to reliability and maintainability engineering.* Waveland Press, 1997.

[24] A. Desrochers and R. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis.* IEEE Press, 1995.

[25] G. Bolch, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications.* John Wiley & Sons, Inc, 2nd ed., 2006.

[26] C. Thormark, "A low energy building in a life cycle–its embodied energy, energy need for operation and recycling potential," *Building and environment*, vol. 37, no. 4, pp. 429–435, 2002.

[27] L. Yang, R. Zmeureanu, and H. Rivard, "Comparison of environmental impacts of two residential heating systems," *Building and Environment*, vol. 43, no. 6, pp. 1072–1081, 2008.

[28] B. De Meester, J. Dewulf, S. Verbeke, A. Janssens, and H. Van Langenhove, "Exergetic life-cycle assessment (ELCA) for resource consumption evaluation in the built environment," *Building and Environment*, vol. 44, no. 1, pp. 11–17, 2009.

[29] U. S. E. P. Agency, "Overview of greenhouse gases," tech. rep., USA, 2011.

[30] C. A. Petri, *Kommunikation mit Automaten*. Darmstad University, Germany: PhD Dissertation, 1962.

[31] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[32] P. Maciel, R. Lins, and P. Cunha, "Introdução às Redes de Petri e Aplicações," *X Escola de Computação, Campinas, SP*, 1996.

[33] P. Merlin and D. J. Faber, "Recoverability of communication protocols: Implicatons of a theoretical study," *IEEE Transactions on Communications*, vol. 24, pp. 1036–1043, Sept. 1976.

[34] J. Noe and G. Nutt, "Macro E-Nets for Representation of Parallel Systems," *IEEE Transactions on Computers*, vol. 31, no. 9, pp. 718–727, 1973.

[35] W. van der Aalst, K. van Hee, and H. Reijers, "Analysis of discrete-time stochastic petri nets," *Statistica Neerlandica*, vol. 54, no. 2, pp. 237–255, 2000.

[36] W. Reisig, *Petri nets: an introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[37] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.

[38] R. German, *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[39] M. Xie, Y. Dai, and K. Poh, *Computing systems reliability: models and analysis*. Springer Us, 2004.

[40] R. A. Sahner, K. S. Trivedi, and A. Puliafito, *Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package*. Norwell, MA, USA: Kluwer Academic Publishers, 1996.

[41] S. Distefano, M. Scarpa, and A. Puliafito, "Modeling distributed computing system reliability with DRBD," in *25th IEEE Symposium on Reliable Distributed Systems, 2006. SRDS'06*, pp. 106–118, 2006.

[42] L. Leemis, *Reliability: Probabilistic Models and Statistical Methods*. Ascended Ideas, 2009.

[43] R. Sarker and C. Newton, *Optimization modelling: a practical approach*. CRC, 2008.

[44] S. S. Rao, *Engineering Optimization - Theory and Practice*. John Wiley and Sons, 2009.

[45] M. R. R. Martí, V. Campos and A. Duarte, "Multi-objective grasp with path-relinking," *AT&T Labs Research Technical Report*, pp. 1–21, 2011.

[46] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[47] M. Resende, "Computing approximate solutions of the maximum covering problem with grasp," *Journal of Heuristics*, vol. 4, no. 2, pp. 161–177, 1998.

[48] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.

[49] M. J. Moran and H. N. Shapiro, *Fundamentals of Engineering Thermodynamics*. John Wiley Sons, Inc., 5 ed., 2006.

[50] Y. A. Cengel and M. A. Boles, *Thermodynamics an Engineering Approach*. 5 ed., 2003.

[51] G. Ciardo, A. Blakemore, J. Chimento, PhilipF., J. Muppala, and K. Trivedi, "Automated generation and analysis of markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models* (C. Meyer and R. Plemmons, eds.), vol. 48 of *The IMA Volumes in Mathematics and its Applications*, pp. 145–191, Springer New York, 1993.

[52] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, pp. 541–580, April 1989.

[53] G. Ciardo, J. Muppala, and K. Trivedi, "SPNP: stochastic Petri net package," *Proceedings of the Third International Workshop on Petri Nets and Performance Models. PNPM89.*, pp. 142–151, 1989.

[54] M. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 2, 1998.

[55] M. Souza, L. Ochi, and N. Maculan, "A grasp-tabu search algorithm for solving school timetabling problems," *Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, Boston*, pp. 659–672, 2003.

[56] M. Resende, "Greedy randomized adaptive search procedures (grasp)," tech. rep., AT&T Labs Research Technical Report, 1998.

[57] J. a. Ferreira, G. Callou, and P. Maciel, "A power load distribution algorithm to optimize data center electrical flow," *Energies*, vol. 6, no. 7, pp. 3422–3443, 2013.

[58] B. Silva, P. Maciel, E. Tavares, C. Araujo, G. Callou, E. Sousa, N. Rosa, M. Marwah, R. Sharma, A. Shah, T. Christian, and J. Pires, "Astro: A tool for dependability evaluation of data center infrastructures," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 783 –790, oct. 2010.

[59] W. Chen, *The Electrical Engineering Handbook*. ACADEMIC PRESS, 2005.

[60] P. Maciel, K. Trivedi, R. Matias, and D. Kim, *Dependability Modeling*, vol. 1. Pennsylvania: IGI Global, 2011.

[61] D. C. F. Ron Larson, *Elementary Linear Algebra*. Cengage Learning, 2008.

[62] G. Balbo and G. Chiola, "Stochastic petri net simulation," in *WSC '89: Proceedings of the 21st conference on Winter simulation*, (New York, NY, USA), pp. 266–276, ACM, 1989.

[63] C. A. Chung, *Simulation Modeling Handbook: A Practical Approach*. CRC Press, Inc., 2003.

[64] M. Marwah, P. Maciel, A. Shah, R. Sharma, T. Christian, V. Almeida, C. Araújo, E. Souza, G. Callou, B. Silva, S. Galdino, and J. Pires, "Quantifying the sustainability impact of data center availability," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, pp. 64–68, March 2010.

[65] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "Timenet: a toolkit for evaluating non-markovian stochastic petri nets," *Performance Evaluation*, vol. 24, no. 1-2, pp. 69 – 87.

[66] A. R. Sahner and S. K. Trivedi, "Reliability modeling using sharpe," tech. rep., Durham, NC, USA, 1986.

[67] *IEEE Gold Book 473, Design of Reliable Industrial and Commercial Power Systems*. 2007.

[68] A. V., "Comparing availability of various rack power redundancy configurations," 2003.

[69] Hewlett-Packard, "Hp power advisor tool." http://h18004.www1.hp.com/products/solutions/power, 2013.

[70] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, vol. 1. Wiley, 1991.

[71] S. G. Schwieger, *2010 Brazil Energy Handbook*. Las Vegas, Nev (USA): PSI Media Inc, 2010.

[72] U. E. I. A. (EIA), "Monthly energy review - october 2011," tech. rep., U.S. Department of Energy, 2011.

[73] "U.s. energy information administration." http://http://www.eia.gov/, 2012.

[74] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you?," in *Proceedings of the 5th USENIX conference on File and Storage Technologies*, FAST '07, (Berkeley, CA, USA), USENIX Association, 2007.

[75] L. Zhou and W. Grover, "A theory for setting the "safety margin" on availability guarantees in an sla," in *Design of Reliable Communication Networks, 2005. (DRCN 2005). Proceedings.5th International Workshop on*, p. 7 pp., oct. 2005.

[76] "Service level agreement for data center services." http://www.earthlinkbusiness.com/ _static/_files/_pdfs/legal/DataCenterService SLA.pdf, 2012.

[77] B. Schroeder and G. Gibson, "Understanding failures in petascale computers," in *Journal of Physics: Conference Series*, vol. 78, p. 012022, IOP Publishing, 2007.

[78] Z. Zhou, S. Cheng, and B. Hua, "Supply chain optimization of continuous process industries with sustainability considerations," *Computers & Chemical Engineering*, vol. 24, no. 2-7, pp. 1151–1158, 2000.

[79] *World Commission on Environmental and Development (WCED), Our Common Future*. Oxford University Press, 1987.

[80] M. Wiboonrat, "An empirical study on data center system failure diagnosis," in *Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, (Washington, DC, USA), pp. 103–108, IEEE Computer Society, 2008.

[81] M. Wiboonrat, "Risk anatomy of data center power distribution systems," in *Proceedings of IEEE International Conference on Sustainable Energy Technologies - ICSET*, pp. 674 – 679, 2008.

[82] H. Xu, L. Xing, and R. Robidoux, "Drbd: Dynamic reliability block diagrams for system reliability modeling," *International Journal of Computers and Applications*, 2008.

[83] R. Robidoux, H. Xu, S. Member, L. Xing, S. Member, and M. Zhou, "Automated modeling of dynamic reliability block diagrams using colored petri nets," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 40, pp. 337–351, March 2010.

[84] B. Wei, C. Lin, and X. Kong, "Dependability modeling and analysis for the virtual data center of cloud computing," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pp. 784 –789, sept. 2011.

[85] Y. Dutuit, E. Châtelet, J. Signoret, and P. Thomas, "Dependability modelling and evaluation by using stochastic petri nets: application to two test cases," *Reliability Engineering & System Safety*, vol. 55, no. 2, pp. 117–124, 1997.

[86] S. A. Vilkomir, D. L. Parnas, V. B. Mendiratta, and E. Murphy, "Segregated failures model for availability evaluation of fault-tolerant systems," in *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, (Darlinghurst, Australia, Australia), pp. 55–61, Australian Computer Society, Inc., 2006.

[87] S. A. Vilkomir, D. L. Parnas, V. B. Mendiratta, and E. Murphy, "Availability evaluation of hardware/software systems with several recovery procedures," in *Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01*, COMPSAC '05, (Washington, DC, USA), pp. 473–478, IEEE Computer Society, 2005.

[88] D. Gmach, Y. Chen, A. Shah, J. Rolia, and C. Bash, "Profiling sustainability of data centers," in *Sustainable Systems and Technology (ISSST), 2010 IEEE International Symposium on*, pp. 1–6, IEEE, 2010.

[89] R. K. Sharma and et al, "On building next generation data centers: energy flow in the information technology stack," in *Compute '08*, 2008.

[90] M. Patterson, "The effect of data center temperature on energy efficiency," in *ITHERM'08*, 2008.

[91] K. Herold and R. Radermacher, "Integrated power and cooling systems for data centers," in *The Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems - ITHERM*, pp. 808 – 811, 2002.

[92] J. Chang, J. Meza, P. Ranganathan, C. Bash, and A. Shah, "Green server design: beyond operational energy to sustainability," in *HotPower*, 2010.

[93] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," *SIGMETRICS Perform. Eval. Rev.*, vol. 39, pp. 181–192, June 2011.

[94] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Cost- and energy-aware load distribution across data centers," in *Proceedings of Hotpower*, (New York, NY, USA), ACM, 2009.

[95] K. Le, O. Bilgir, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Managing the cost, energy consumption, and carbon footprint of internet services," in *SIGMETRICS '10: Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, (New York, NY, USA), pp. 357–358, ACM, 2010.

[96] A. Curtis, T. Carpenter, M. Elsheikh, A. Lopez-Ortiz, and S. Keshav, "Rewire: An optimization-based framework for unstructured data center network design," in *INFOCOM, 2012 Proceedings IEEE*, pp. 1116 –1124, march 2012.

[97] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "Carpo: Correlation-aware power optimization in data center networks," in *INFOCOM, 2012 Proceedings IEEE*, pp. 1125 –1133, march 2012.

[98] C.-W. Ang and C.-K. Tham, "Analysis and optimization of service availability in a ha cluster with load-dependent machine availability," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, pp. 1307 –1319, sept. 2007.

[99] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations (Systems & Control: Foundations & Applications)*. Birkhäuser Boston, 1 ed., Dec. 1997.

[100] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 1994.

[101] G. Callou, E. Sousa, P. Maciel, E. Tavares, B. Silva, J. Figueiredo, C. Araujo, F. Magnani, and F. Neves, "A formal approach to the quantification of sustainability and dependability metrics on data center infrastructures.," in *Proceedings of the 2011 Symposium on Theory of Modeling and Simulation (TMS/DEVS)*, (Boston, MA, USA), pp. 1–8, ACM, 2011.

[102] G. Callou, P. Maciel, F. Magnani, J. Figueiredo, E. Sousa, E. Tavares, B. Silva, F. Neves, and C. Araujo, "Estimating sustainability impact, total cost of ownership and dependability metrics on data center infrastructures," in *Sustainable Systems and Technology (ISSST), 2011 IEEE International Symposium on*, pp. 1 –6, may 2011.

[103] G. Callou, P. Maciel, F. Magnani, E. Tavares, E. Sousa, B. Silva, J. Figueiredo, C. Araujo, and F. Neves, "Sustainability and dependability evaluation on data center architectures," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, (Anchorage, Alaska, USA), pp. 1–6, IEEE New York, 2011.

[104] G. Callou, E. Sousa, P. Maciel, E. Tavares, C. Araujo, B. Silva, N. Rosa, M. Marwah, R. Sharma, A. Shah, T. Christian, J. P. Pires, and F. Magnani, "Sustainability and dependability evaluation on data center architectures," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, (Istanbul, USA), pp. 526–533, IEEE New York, 2010.

[105] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd ed., 2001.

[106] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Evolutionary Multi-Criterion*

*Optimization* (S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, eds.), vol. 4403 of *Lecture Notes in Computer Science*, pp. 742–756, Springer Berlin Heidelberg, 2007.

[107] "Economic input-output life cycle assessment (eio-lca)." http://www.eiolca.net, 2009.

APPENDIX A

# EXERGY LIFE CYCLE ASSESSMENT

In our previous work [103] [101] [102], we have computed the exergy consumption taking into account embedded and operational exergies. The embedded exergy is related to product design decisions which may include material extraction, manufacturing as well as end-of-life; and the operational exergy corresponds to the decisions during product use. Embedded exergy depends on the required energy to perform the raw material extraction, the manufacturing process as well as the transportation phase. Embedded energy consumption estimates are based on a LCA approach fed by [107]. To perform the sustainability evaluation, a sustainability model that may represent the data center power and cooling infrastructures is build considering those embedded energy estimates. The evaluation of that model is conducted to obtain the embedded and operational exergy consumptions. Section 3.1 presented how to compute the operational exergy consumption. Due to the difficulty to take into account other exergy sources, which is not on the method proper but on the data assessment, we adopted to consider the embedded exergy consumption only in the appendix.

## A.1 SUSTAINABILITY MODEL

Environmental impact may be computed in terms of the thermodynamic metric of exergy (also called usable available energy). The following paragraphs presents the equations adopted to compute the embedded exergy consumption.

### A.1.1 Embedded Exergy

The embedded exergy, which involves impacts related to product design decisions, is obtained as follows:

$$Ex_{emb} = E_{man} \times [\eta_{man} + (1 - \eta_{man}) \times (1 - C_{reuse})] \tag{A.1}$$

where $E_{man}$ is the energy required for manufacturing all equipments adopted in the infrastructure (see Equation A.2); $\eta_{man}$ is the manufacturing efficiency (2nd law of thermodynamics); and $C_{reuse}$ is the exergy reused in other processes.

$$E_{man} = \sum_{i=1}^{n} E_{eq_i} \tag{A.2}$$

In the particular case of electrical energy, all energy can be theoretically converted into work, thus, the variable $E_{man}$ in Equation A.2 means the total exergy made accessible during the manufacturing phase. A fraction of $E_{man}$ is consumed. The complementary fraction (1-$E_{man}$) can be reused in other processes ($C_{reuse}$) or destroyed (1-$C_{reuse}$).

In this work, the energy required for manufacturing each equipment ($E_{eq_i}$) has been obtained from [107].

### A.1.2 Lifetime Exergy

The sum of the embedded and operational exergies is the lifetime exergy (LTE) consumption.

$$LTE = Ex_{emb} + Ex_{op} \tag{A.3}$$