Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

Uma Estratégia baseada em Algoritmos Genéticos para Otimizar Arquiteturas de Data Centers

Márcio Sérgio Soares Austregésilo

Recife

Abril de 2018

Márcio Austregésilo

Uma estratégia baseada em Algoritmos Genéticos para otimizar arquiteturas de Data Centers

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Abril de 2018

À,
Meus pais, minha família, meus amigos de toda a vida

Agradecimentos

Agradeço primeiramente a Deus e ao Mestre Jesus, pois sem eles eu nada seria. Agradeço imensamente aos meus pais, que me proporcionaram todas as condições possíveis à minha realidade para eu chegar até aqui. Agradeço ao meu orientador, o Prof. Dr. Gustavo Callou, por ter se mostrado sempre solícito, paciente e, principalmente, por nunca ter desistido de mim. Agradeço aos colegas pelo companheirismo e ajuda com os estudos em grupo, dividindo a carga desta trajetória tão difícil. Agradeço ao Prof. Dr. Marcos Nunes, do Departamento de Educação Física da UFRPE, que, durante sua gestão como meu diretor, me deu todo apoio necessário para que eu pudesse avançar neste mestrado. Agradeço à Prof. Dr. Maria Cecília Tenório, atual gestora do Departamento de Educação Física da UFRPE e minha atual diretora, por todo apoio neste fim de jornada nesta pós-graduação. Agradeço a todos os meus amigos por estarem sempre ao meu lado me dando forças pra seguir adiante.

Resumo

É crescente, na sociedade, a preocupação com a sustentabilidade e os impactos ambientais causados pelo consumo e geração de energia. Existe uma grande pressão mundial para que empresas adotem práticas sustentáveis, não só pela economia financeira decorrente da redução do consumo de energia, como pela conscientização do esgotamento dos recursos naturais para gerações futuras. Considerando o avanço da tecnologia, a sociedade vem necessitando de mais serviços interligados à Internet e, consequentemente, de infraestrutura para esses serviços, fator que impacta de maneira significativa no consumo de energia elétrica.

Nos últimos anos, o crescimento da tecnologia vem demandando uma maior confiabilidade, acessibilidade, colaboração, disponibilidade e redução de custos dos *data centers*, devido a fatores como redes sociais, computação nas nuvens e comércio eletrônico. Esses sistemas necessitam de toda uma infraestrutura com mecanismos de redundância para funcionar com alta disponibilidade, fato que implica num grande consumo de energia elétrica impactando na sustentabilidade e custo operacional.

Este trabalho propõe a utilização de algoritmos genéticos multiobjetivos para otimizar custo, impacto ambiental e disponibilidade da infraestrutura de energia elétrica desses sistemas. O objetivo é maximizar a disponibilidade e minimizar o custo total e a exergia operacional (utilizada para estimar o impacto ambiental). Para se computar tais métricas são utilizados o Modelo de Fluxo de Energia (EFM), o Diagrama de Bloco de Confiabilidade (RBD) e a Rede de Petri Estocástica (SPN). Dois estudos de caso são conduzidos: (i) leva em consideração 5 arquiteturas típicas de data centers para mostrar a aplicabilidade e validação da estratégia proposta; (ii) utiliza a estratégia de otimização em quatro arquiteturas classificadas pela norma ANSI/TIA-942 (TIER I à TIER IV). Em ambos estudos

de caso, observou-se uma melhora significativa nos resultados que ficaram bem próximos ao ótimo. Vale ressaltar que o tempo utilizado para se obter as respostas utilizando a abordagem com algoritmo genético foi significativamente inferior (6.763.260 vezes) se comparado a uma estratégia que combina todas as possibilidades para obter o resultado ótimo.

Abstract

A growing concern in society is related about sustainability and the environmental impacts caused by energy consumption and generation. In deed, there is great global pressure for companies to adopt sustainable practices, not only because of the financial savings from reducing energy consumption, but also due to the awareness of the depletion of natural resources for future generations. Considering the advancements of technology, the society needs more interconnected services to the Internet and, consequently, the infrastructure demanded to support for these services also contributes with a significant impact on the consumption of electricity.

In recent years, due to factors such as social networking, cloud computing, and e-commerce, data center has been growing in importance and, so, its reliability, collaboration, availability, and cost reduction represent elements under studies. The data center systems require an entire infrastructure with redundancy mechanisms to operate with high availability, a fact that may have a negative impact on both the electrical energy consumption and operational cost.

This work proposes a method based on multiobjective genetic algorithms to optimize cost, environmental impact and availability of electrical energy infrastructures of data centers. The main goal is to maximize the availability, minimize the total cost and the operational exergy (used to estimate the environmental impact). To compute such metrics, models were proposed using the Energy Flow Model (EFM), Reliability Block Diagram (RBD) and Stochastic Petri Net (SPN). Two case studies are conducted to show the applicability of the proposed strategy: (i) takes into account 5 typical data center architectures that were optimized to conduct the validation process of the proposed strategy; (ii) uses the

optimization strategy in four architectures classified by ANSI / TIA-942 (TIER I to TIER IV). In both case studies, significant improvements were achieved in the results, which were very close to the optimum. It is worth mentioning that the time used to obtain the results using the genetic algorithm approach was significantly lower (6,763,260 times), in comparison with the strategy which combines all the possible combinations to obtain the optimal result.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A evolução da tecnologia da informação (TI) vem transformando a sociedade atual através de, por exemplo, redes sociais, comércio eletrônico e *Internet-banking*. Essa evolução impacta desde o relacionamento pessoal até a interação profissional colaborativa entre empresas e pessoas. Devido a esses fatos, a demanda por sistemas com alta confiabilidade, acessibilidade, colaboração, disponibilidade e baixos custos, dentre outros, culminam no uso de um recente paradigma: a computação em nuvem.

A computação em nuvem é definida por algumas características essenciais: auto-atendimento sob demanda, acesso amplo à rede, agrupamento de recursos, elasticidade e serviço sob medida [1]. De forma mais específica, a computação em nuvem pode ser definida como um modelo que permite acesso conveniente, ubíquo e sob demanda, de uma rede a um conjunto compartilhado de recursos computacionais configuráveis, como por exemplo, redes de computadores, servidores, armazenamento, aplicativos e serviços [2]. O termo computação em nuvem é denominado pelo fato de que o usuário não necessita determinar o local físico específico e a disposição dos equipamentos que hospedam os recursos que são, em última análise, autorizados para o uso.

O crescimento no número de processos pelo uso da computação em nuvem vêm demandando maiores investimentos e atenção por parte dos gestores e analistas na infraestrutura dos *data centers* que provêm suporte a tais sistemas. Com o aumento dessa infraestrutura, para prover suporte à tolerância a falhas, por exemplo, são adicionados equipamentos o que

acaba por aumentar o custo e o consumo de energia.

Outro fator requisitado nesses sistemas computacionais é o de alta disponibilidade, sendo assim a infraestrutura elétrica do *data center* deve ter altas taxas de disponibilidade. A redundância é um princípio de projeto amplamente adotado em tolerância a falhas para aumentar a disponibilidade de sistemas [3].

Este trabalho propõe a otimização de arquiteturas elétricas de *data centers*, visando maximizar a disponibilidade e minimizar o impacto ambiental e o custo total. O impacto ambiental é parametrizado através da exergia destruída, que é a energia dissipada pelos componentes da arquitetura. Sendo assim, esse trabalho faz uso de uma técnica de otimização multi-objetiva de maneira a equilibrar essas métricas levando em consideração a relação entre elas. Um algoritmo genético multi-objetivo é proposto para realizar a otimização das arquiteturas elétricas de *data center*. Esse algoritmo desenvolvido se comunica com a ferramenta Mercury [4] para fazer a otimização dos modelos propostos. Ao ajustar os parâmetros dos modelos de disponibilidade nos Diagramas de Bloco de Confiabilidade (RBD) e redes de Petri Estocásticas (SPN), bem como nos Modelos de Fluxo de Energia (EFM).

1.1 Motivação e Justificativa

A sustentabilidade tem recebido cada vez mais atenção pela comunidade científica, devido à preocupação em atender às necessidades atuais de energia sem comprometer, por exemplo, recursos não renováveis para as gerações futuras[5]. De modo geral, existe uma maior preocupação com o que a natureza tem a oferecer do que com os danos causados a ela. Essa cultura, enfatizada desde o início da era industrial, vem trazendo consequências graves aos recursos naturais, tornando-os cada vez mais escassos ao longo dos anos [6].

A poluição, a degradação ambiental e as mudanças climáticas são exemplos de preocupações, com o impacto ambiental, da comunidade científica e da indústria. Por exemplo, as emissões de CO_2 deverem aumentar em torno de 70% até 2020 [7], devido ao uso de gás natural, carvão, petróleo e outras fontes de energia que contribuem para o aquecimento global.

As causas e o impacto das mudanças climáticas globais estão sujeitas a uma diversidade de opiniões, mas atualmente é impossível negar que as mudanças climáticas estejam acontecendo. Da mesma forma, torna-se cada vez mais difícil para as organizações ignorarem as significativas pressões ambientais que elas enfrentam sem que medidas diversas sejam tomadas [6]. Em pesquisa recente realizada pelas Nações Unidas quanto aos assuntos dominantes no futuro, o desenvolvimento sustentável aparece como a principal preocupação[8].

A difusão da computação em nuvem e o crescimento da utilização de computadores, *tablets*, *smartfones* e dispositivos interligados à Internet, vem aumentando a demanda de infraestrutura para manter os serviços relacionados a computação em nuvem. Os *data centers* estão tendo, cada vez mais, uma maior necessidade de prover taxas altas de disponibilidade em suas infraestruturas para manter esses sistemas que dão suporte à computação em nuvem. Uma maior disponibilidade implica em maior redundância de equipamentos, resultando em aumento do custo e do consumo de energia.

De acordo com um estudo da Consultoria Gartner [9], os equipamentos de informática são responsáveis por 2% das emissões de CO_2 em todo o mundo, o que corresponde à quantidade emitida por todos os aviões existentes. Os *data centers* consomem cerca de 2% da geração de energia dos Estados Unidos [10] e, portanto, também contribuem significativamente para as emissões globais de carbono.

Um *data center* não é apenas composto por equipamentos de TI, mas também pelas infraestruturas de energia e refrigeração que incorrem em considerável consumo de energia. O aumento do custo de energia e a crescente atenção global dada em sustentabilidade tornam, além da disponibilidade, o gerenciamento do consumo de energia por *data centers* crítico [11].

Como a infraestrutura de TI é responsável por 2% das emissões globais de carbono [7], qualquer pequena porcentagem de redução no consumo de energia dos *data centers* terá um considerável impacto econômico e ambiental.

1.2 objetivos

Essa dissertação propõe uma estratégia baseada num algoritmo genético multi-objetivo que otimize as arquiteturas elétricas de *data center* de maneira a aumentar a disponibilidade, reduzir o custo e o impacto ambiental. A estratégia adotada inicia com o entendimento da arquitetura elétrica de um *data center* , seguida pela criação de submodelos que resultem nos modelos RBD, SPN e EFM da referida arquitetura. Será utilizado o ambiente Mercury [4] para a concepção e combinação destes modelos. Em seguida, o algoritmo genético proposto se comunicará com o Mercury para ajustar modelos RBD, SPN e EFM a partir de uma base dados de entrada contendo vários equipamentos (com diferentes parâmetros, por exemplo, MTTF, custo e eficiência energética) pertencentes a infraestrutura elétrica de *data center*. Estes parâmetros servirão de entrada para a função *fitness* do algoritmo genético multi-objetivo que, ao final de seu ciclo, retornará um conjunto de soluções ótimas (ou próximas da ótima).

De forma mais específica os objetivos deste trabalho são:

- Propor uma metodologia que auxilie no entendimento e otimização de arquitetura elétrica;
- Construir modelos RBD, SPN e EFM da arquitetura para se poder estimar a disponibilidade, o custo e o consumo de energia de *data centers*;
- Desenvolver um algoritmo genético multiobjetivo que se comunique com o ambiente Mercury utilizando os modelos RBD, SPN e EFM para computar as métricas de interesse (disponibilidade, custo e consumo de energia, por exemplo), e por fim realizar a otimização;

Dois estudos de caso serão apresentados para validar a metodologia proposta neste trabalho. O primeiro será sobre a aplicação do algoritmo genético multi-objetivo em cinco arquiteturas básicas de *data center*. Os resultados desta otimização serão comparados com a otimização por força bruta que analisa todas as possibilidades e retorna a solução ótima. O comparação destas duas heurísticas de otimização comprovará a eficiência e eficácia do algoritmo genético proposto neste trabalho.

O segundo estudo caso será a aplicação da metodologia proposta para otimizar as arquiteturas elétricas que se adéquem a classificação *TIER* [12] proposta pela *Telecommunications Industry Association* e certificada pelo *Uptime Institute Professional Services* [13].

1.3 Estrutura da dissertação

O restante dessa dissertação está organizada como segue. O Capítulo 2 apresenta conceitos teóricos para a fundamentação dos conteúdos deste trabalho. O Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 explana a metodologia aplicada na utilização do Algoritmo Genético para realizar a otimização das arquiteturas de *data centers*. O Capítulo 5 ilustra dois estudos de caso, para validar a metodologia, e apresenta a análise dos resultados obtidos. Por fim, O Capítulo 6 conclui a dissertação e descreve direcionamentos para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão explanados os conceitos e conhecimentos necessários para um melhor entendimento do trabalho proposto.

2.1 Data Center

Os *Data centers* são ambientes conhecidos por possuírem um papel crítico, ou seja, são ambientes que abrigam equipamentos responsáveis pelo processamento e armazenamento de informações imprescindíveis para a continuidade de negócios nas mais diversas organizações, sejam elas empresas, instituições de ensino, indústrias, órgãos governamentais, hospitais, hotéis, entre outros [14].

Data centers podem ser adotados como *host* para sites com a finalidade de processar transações comerciais, para proteger dados (por exemplo, registros financeiros, histórico médico de pacientes, gerenciar e-mails). Além disso, as organizações estão procurando por departamentos de TI com o objetivo de receber suporte em muitas fontes de negócios, como produtividade e receita. Portanto, os *data centers* devem ter altos níveis de disponibilidade para suportar novos requisitos de tecnologia [11]. O problema é que os *data centers* são um ambiente muito dinâmico: equipamentos vão se tornando obsoletos e substituídos; novos equipamentos, que demandam novas interfaces e novos tipos de ligações, são introduzidos;

novos clientes e novas configurações surgem a cada dia [15].

Os *data centers* desempenham uma parte importante de nossas vidas, dada a grande quantidade de informações digitais que eles armazenam. De acordo com os padrões, as melhores práticas e os requisitos dos usuários, a infraestrutura do *data centers* deve atender a requisitos técnicos rigorosos para garantir a confiabilidade, a disponibilidade e a segurança, pois eles têm um impacto direto no custo e na eficiência. Uma infraestrutura com alta confiabilidade e disponibilidade deve ter redundância do sistema, e por esse motivo, será mais caro e provavelmente irá consumir mais energia elétrica [16].

O *data center* compreende os seguintes sistemas [14]:

- sala de computadores (*computer room*);
- ar-condicionado e controle ambiental;
- distribuição elétrica e UPS (*Uninterruptable Power Supply*);
- automação do edifício;
- detecção de surpresa de incêndio;
- segurança e controle;
- espaços de suporte, entre outros.

Para construir uma sala de *data center*, muitas infraestruturas devem ser analisadas como por exemplo: energia, refrigeração, conectividade, espaço, proteções (por exemplo, contra fogo) e monitoramento de temperatura. Além disso, os projetistas dos *data centers* devem se concentrar no aumento da produtividade e evitar o tempo de inatividade. As estratégias de projeto são importantes para atingir essas necessidades. As infraestruturas dos *data centers* devem ser compostas por [11]:

- fontes de energia de reserva, que são responsáveis por suportar a carga elétrica do *data center* quando a energia primária falhar;
- rede redundante para manter o sistema funcionando se um dispositivo de rede falhar;

- conexões de dados redundantes para fornecer caminhos alternativos ao dispositivo de desejo.

2.1.1 Redundância

A redundância nos *data centers* é adotada em módulos, componentes e sistemas com a propósito de reduzir o *downtime* (tempo de parada) devido a falhas técnicas e humanas, ou manutenção preventiva/corretiva [14].

2.1.2 Classificação

A principal referência de classificação de *data centers* é a norma ANSI/TIA-942 [12], proposta pela *Telecommunications Industry Association*, que classifica os *data centers* quanto a sua redundância e disponibilidade. Essa norma estabelece algumas definições de redundância para auxiliar a classificação, são elas:

- **N**: o sistema atende aos requisitos básicos e não possui redundância;
- **N+1**: A redundância fornece uma unidade, módulo, caminho ou sistema adicional, além do mínimo necessário para satisfazer o requisito de base. A falha ou manutenção de qualquer unidade, módulo ou caminho não irá interromper as operações. (ex.: Energia Elétrica da concessionária + *Nobreak*);
- **N+2**: A redundância fornece duas unidades, módulos, caminhos ou sistemas adicionais, além do mínimo necessário para satisfazer o requisito de base. A falha ou a manutenção de duas únicas unidades, módulos ou caminhos não interromperão as operações. (ex.: Energia Elétrica concessionária + *Nobreak* + Gerador);
- **2N**: fornece duas unidades completas, módulos, caminhos ou sistemas para todos os componentes requeridos para um sistema base. Falha ou manutenção de uma unidade inteira, módulo, caminho ou sistema não irá interromper as operações (ex.: Energia Elétrica concessionária A + Energia Elétrica concessionária B);

- **2(N+1)**: fornece duas unidades completas (N+1), módulos, caminhos ou sistemas. Mesmo em caso de falha ou manutenção de uma unidade, módulo, caminho ou sistema, as operações não são interrompidas. (ex.: Energia Elétrica concessionária A + Energia Elétrica concessionária B; *Nobreak* A+ *Nobreak* B; Gerador A + Gerador B).

Essa alssificação prevê vários níveis de confiabilidade em *data centers*, classificando-os de acordo com o tempo em que o ambiente se encontra operacional. Essa norma possui uma abordagem que consiste em definir uma série de elementos funcionais que se conectam hierarquicamente. *Data centers* de menor porte e confiabilidade possuem um conjunto menor de elementos funcionais, mas os padrões de construção permanecem sempre os mesmos [15].

Por fim a norma ANSI/TIA-942 [12] estabelece quatro níveis de classificação dos *Data centers* de acordo com a sua infraestrutura e confiabilidade [17]:

- **TIER I (Básico)**: está sujeito às interrupções em suas atividades, tanto as planejadas quanto as não. Possui redundância N. Tal *data center* pode possuir quadro de distribuição de energia e refrigeração, *nobreak*, gerador (motor). Se o *data center* desse nível possuir um *nobreak* ou gerador, estes serão normalmente sistemas de módulo único, o que ocasionará pontos de falhas. Nesses casos, para se realizar a manutenção preventiva ou reparação dos equipamentos, os sistemas são desligados;

- **TIER II (componentes redundantes)**: é um pouco mais suscetível que o TIER I às interrupções planejadas ou não. Possui *nobreak* e geradores na estrutura de redundância N+1, porém em um único segmento. É preciso haver o desligamento dos sistemas de energia quando se faz necessária a manutenção;

- **TIER III (paralelamente sustentável)**: possui equipamentos de refrigeração e alimentação de energia redundantes, porém com apenas um equipamento de cada segmento ligado, possibilitando atividades planejadas sem interromper a operação. O TIER III ainda está sujeito às falhas de operação e de componentes;

- **TIER IV (tolerante a falhas)**: possui equipamentos de refrigeração e alimentação de energia redundantes e ativos proporcionando tolerância às falhas. No *data center* TIER IV, equipamentos que não são construídos com múltiplas entradas de energia

devem utilizar uma chave de transferência automática para que não haja nenhum tipo de interrupção. O TIER IV suporta atividades planejadas de manutenção sem interrupção, pois possui redundância 2(N+1).

O órgão responsável certificar e classificar os *data centers* de acordo com esta norma é o *Uptime Institute Professional Services* [13]. A tabela 2.1 apresenta um resumo da classificação TIER em função de sua redundância, tempo de inatividade e disponibilidade [14].

Classificação	Redundância	Paradas por ano	Indisp. (h)	Disp.
TIER I	N	2 x 12h para manutenção	28,8h	99,67%
TIER II	N+1	3 x 2h para manutenção	22h	99,75%
TIER III	N+2 (min. N+1)	-	1,6h	99,98%
TIER IV	2(N+1) ou 2N	-	0,8h	99,99%

Tabela 2.1: Requisitos de disponibilidade por classificação TIER de *data centers*.

2.1.3 Infraestrutura

Os *data centers* são compostos por três tipos básicos de infraestrutura como mostra a Figura 2.1: infraestrutura de TI, infraestrutura de refrigeração e infraestrutura elétrica.

A infraestrutura de TI consiste em três componentes principais: servidores, redes e dispositivos de armazenamento [18]. Os dispositivos de armazenamento geralmente são conectados através de uma rede de área de armazenamento (SAN). Os servidores também podem se conectar a sistemas de arquivos remotos via armazenamento em rede (NAS) através da *Ethernet* [4]. A infraestrutura de rede consiste nos hardwares físicos adotados para transmitir dados eletronicamente, como *switches*, roteadores, *bridges* e *gateways*. Os serviços de software geralmente são organizados em uma arquitetura de várias camadas com níveis separados para servidores web, aplicativos e servidores de banco de dados [19].

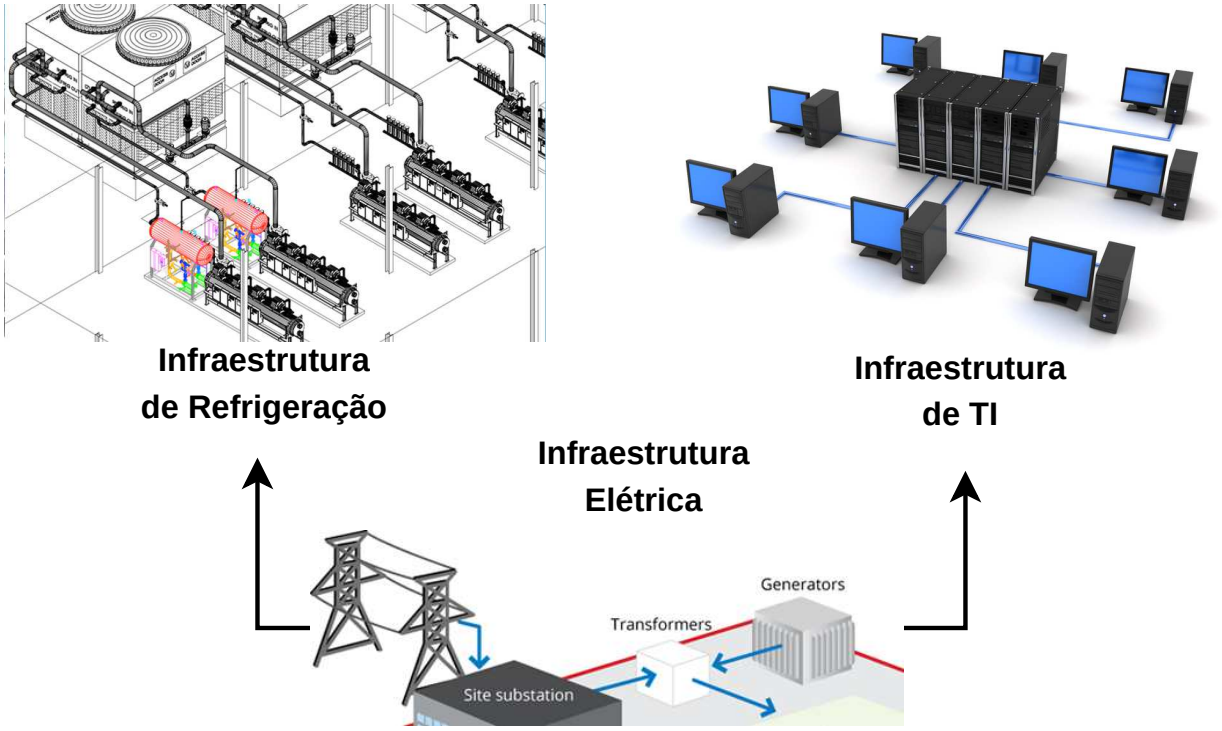

Figura 2.1: Arquitetura de *Data Center*

A infraestrutura de refrigeração compreende unidades de climatização de sala de computadores (CRAC), refrigeradores (*Chillers*) e torres de resfriamento [18]. O sistema de resfriamento representa cerca de 18% do consumo de energia dos *data centers* [20, 21].

A infraestrutura elétrica, foco desse trabalho, é responsável pelo fornecimento ininterrupto de energia elétrica condicionada à tensão e a frequência corretas para ambas as infraestruturas de TI e de refrigeração. A energia elétrica vem das concessionárias de energia e, normalmente, passa pelos transformadores, chaves de transferência, fontes de alimentação ininterrupta (UPS), unidades de distribuição de energia (PDUs) e, por fim, pelas réguas de alimentação do *rack* [22]. Devido à necessidade do aumento da tolerância a falhas, se faz necessário a redundância ou replicação dos componentes afim de aumentar a disponibilidade, incrementando, assim, o consumo de energia. Geradores ou outras fontes de energia locais podem ser utilizados para casos de interrupções mais longas ou para proporcionar uma fração importante da demanda de potência total numa base regular [4]. É na arquitetura da infraestrutura de energia elétrica que este trabalho foca com o objetivo de otimizar a

disponibilidade, reduzindo o custo e o impacto ambiental.

2.2 Dependabilidade

A dependabilidade [23, 24] de um sistema pode ser entendida como a capacidade de fornecer um conjunto de serviços de forma confiável. Sendo assim, a dependabilidade pode indicar a qualidade de um serviço fornecido por um sistema e a confiança depositada nesse serviço [25]. Uma falha é definida como a falha de um componente, subsistema ou sistema que interage com o sistema em questão [26]. Um erro é definido como um estado que pode levar a ocorrência de falha e um defeito representa o desvio do funcionamento correto de um sistema [27].

Alguns dos atributos principais ligados à dependabilidade [28] são: confiabilidade, disponibilidade, segurança de funcionamento, segurança, mantenabilidade. Um resumo dos principais atributos é mostrado a seguir [29].

- **Confiabilidade (*reliability*)**: probabilidade de o sistema fornecer um conjunto de serviços, sem ou na presença de falhas, durante um intervalo de tempo e de acordo com as condições estabelecidas, mesmo que haja componentes defeituosos [30];
- **Disponibilidade (*availability*)**: probabilidade de que o sistema seja operacional levando em consideração a alternância de períodos de funcionamento e reparo [31];
- **Segurança de funcionamento (*safety*)** : probabilidade do sistema estar operacional e executar suas funções corretamente ou interromper suas funções de maneira a não provocar dano a outro sistema que dependa dele;
- **Segurança (*security*)**: proteção contra falhas maliciosas, visando privacidade, autenticidade e integridade dos dados;
- **Mantenabilidade (*Maintainability*)**: é a probabilidade de que um sistema seja reparado em um determinado período de tempo [27].

Para esta dissertação o atributo foco da dependabilidade, utilizado na otimização, será a disponibilidade que será avaliada através de modelos de redes de petri estocásticas (SPN) e diagramas de bloco de confiabilidade (RBD).

2.2.1 Disponibilidade

A disponibilidade é a probabilidade do sistema se manter em funcionamento levando em consideração a ocorrência de falhas e reparo de dispositivos que compõe tal sistema [26]. A definição de disponibilidade pode ser expressada pela relação entre tempo ativo e a soma do tempo ativo (*uptime*) com o tempo inativo (*downtime*). *Uptime* é o período de tempo em que o sistema está operacional, *downtime* é o período de tempo em que o sistema não está operacional devido a ocorrência de um defeito ou atividade de reparo, e o *uptime* + *downtime* é o período de tempo de observação do sistema [27]. A disponibilidade pode ser representada pela Equação 2.1 [32, 31, 33].

$$Disp = \frac{uptime}{uptime + downtime} \tag{2.1}$$

Sempre que não for possível obter o tempo ativo e o tempo inativo, a disponibilidade pode ser avaliada em função das falhas e reparações do sistema. Considerando que o sistema está no estado estacionário, o MTBF (Tempo Médio Entre Falhas) e o MTTF (Tempo Médio Para Falha) podem ser usados como sinônimos e são medidos pelo tempo médio entre a ocorrência de falhas. Da mesma forma, o MTTR (Tempo Médio Para Reparo) é o tempo médio gasto para reparar uma falha [34]. Portanto, a disponibilidade pode ser computada pela Equação 2.2.

$$Disp = \frac{MTTF}{MTTF + MTTR} \tag{2.2}$$

A disponibilidade assume diferentes níveis nos sistemas computacionais e aplicações e, portanto, pode ser classificada conforme esses níveis [27]. A *U.S. Federal Aviation Administration's National Airspace System's Reliability Handbook* classifica os sistemas computaci-

onais e aplicações conforme os seus níveis de criticidade [35]. Os níveis atribuídos a esses sistemas computacionais e aplicações são [35]:

- críticos-seguros: quando a disponibilidade necessária for 99,99999%;
- críticos: quando a disponibilidade necessária for 99,999%;
- essenciais: quando a disponibilidade necessária for 99,9%;
- rotineiros: quando a disponibilidade necessária for 99%.

A disponibilidade é uma medida crítica devido ao fato de que o tempo de inatividade (*downtime*) geralmente afeta a produção das empresas bem como seus lucros. A Tabela 2.2 apresenta diferentes níveis de disponibilidade em número de noves ($-\log(1 - disponibilidade(\%)/100)$) e seu tempo de inatividade (*downtime*) considerando um período de um ano [11].

Disponibilidade (9s)	Percentual	Tempo de Inatividade
Dois noves	99	3 dias, 15 horas, 40 minutos
Três noves	99,9	8 horas, 46 minutos
Quatro noves	99,99	52 minutos, 35 segundos
Cinco noves	99,999	5 minutos, 15 segundos
Seis noves	99,9999	32 segundos

Tabela 2.2: Tempo de inatividade (*downtime*) correspondente à disponibilidades diferentes no período de um ano.

2.3 Exergia e Sustentabilidade

A Segunda Lei da Termodinâmica [36] afirma que há sempre uma perda da qualidade ao se converter um tipo de energia em outro. Ligada a este conceito, a exergia quantifica a energia que pode ser convertida em trabalho útil [36, 37]. Sendo assim exergia pode ser

usada para estimar a eficiência da conversão de energia de um sistema. A exergia é calculada como o produto de energia por um fator de qualidade de acordo com a Equação 2.3.

$$Exergia = Energia \times F \tag{2.3}$$

Em geral, a análise de exergia, no contexto da sustentabilidade, procura maximizar a eficiência da utilização de energia ou avaliar a degradação dos recursos naturais [38]. Este trabalho mede a exergia destruída em uma arquitetura elétrica de computação em nuvem. Quanto menor o valor desta métrica, menor o seu impacto em termos de sustentabilidade [39].

2.4 Redes de Petri

As redes de Petri (PN) [40] são uma família de formalismos muito adequados para modelar vários tipos de sistemas, uma vez que a simultaneidade, a sincronização, os mecanismos de comunicação, bem como os atrasos determinísticos e probabilísticos são naturalmente representados. A aplicabilidade das Redes de Petri como ferramenta para estudo de sistemas é importante por permitir representação matemática, análise dos modelos e também por fornecer informações úteis sobre a estrutura e o comportamento dinâmico dos sistemas modelados [27].

Diversas variantes do modelo de Rede de Petri clássico têm sido desenvolvidas ao longo do tempo, tais como redes temporizadas [41], estocásticas [42], alto-nível [43] e orientadas a objetos [44]. Isso é devido à necessidade de suprir as diferentes áreas de aplicação, além de prover facilidades de comunicação e transferência de métodos e ferramentas de uma área para outra [29].

As redes de Petri são compostas pelos elementos a seguir [40, 45]:

- **Lugares**: tem como principal função o armazenamento de *tokens* os quais são removidos e adicionados à medida que as transições são disparadas. Representam os

elementos passivos da rede. Graficamente, os lugares são representados por círculos (ver Figura 2.2-a).

- **Transições**: são os elementos ativos da rede, em outras palavras, as ações realizadas pelo sistema. Uma transição só estará habilitada para o disparo se todas as suas pré-condições estiverem atendidas. Caso uma pré-condição não seja satisfeita, a transição estará desabilitada para o disparo. Uma vez habilitada a disparar, a transição remove uma determinada quantidade de *tokens* dos lugares de entrada e coloca em outros (nos lugares de saída), gerando assim as pós-condições (ver Figura 2.4). Graficamente, são representadas por traços ou barras (ver Figura 2.2-b).

- **Arcos**: representam o fluxo dos *tokens* pela rede de Petri(ver Figura2.2-c). Graficamente, as transições e os lugares podem ser conectados por múltiplos arcos (arcos multi-valorados) que podem ser compactados em um único arco rotulado com um peso igual a quantidade dos múltiplos arcos da referida conexão(ver Figura 2.3).

- ***Tokens***: representam o estado em que o sistema se encontra em determinado momento (ver Figura 2.2-d).

Figura 2.2: Elementos da rede de Petri.

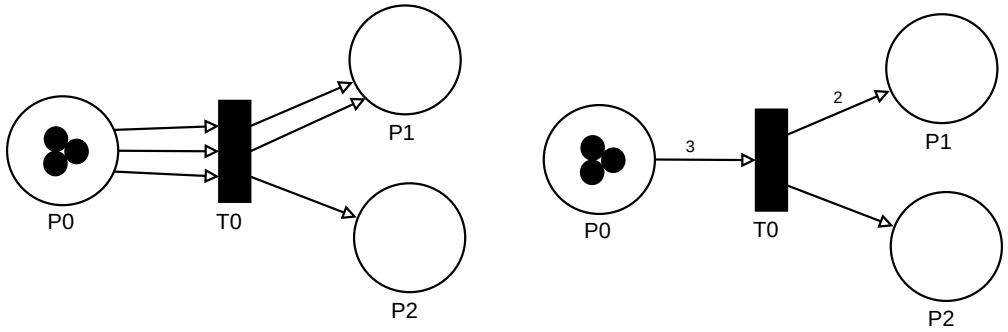

Figura 2.3: Arco multi-valorado.

A representação formal de um modelo em rede de Petri (PN) é a 5-tupla $PN = \{P, T, F, W, M_0\}$ [40], onde:

- $P = \{p_1, p_2, ..., p_m\}$ é o conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é o conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;
- $W : F \rightarrow \{1, 2, 3, ...\}$ é a função de atribuição de peso aos arcos;
- $M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ é a marcação inicial.

O exemplo da Figura 2.4 representa a modelagem do funcionamento de uma lanterna em rede de Petri. Nesse modelo, os lugares representam o estado em que pode se encontrar o funcionamento da lanterna (*Ligado* ou *Desligado*) e as transições representam as ações que alteram o estado da lanterna (*Ligar* ou *Desligar*). Um *token* marca o estado do modelo no lugar correspondente à situação atual da lanterna como descrito na Figura 2.4-a. Considerando que *Ligado* é o estado atual do modelo, transição *Desligar* fica habilitada para disparar. Uma vez disparada esta transição, o modelo passa do estado *Ligado* (ver Figura 2.4-a) para o estado *Desligado* (ver Figura 2.4-b).

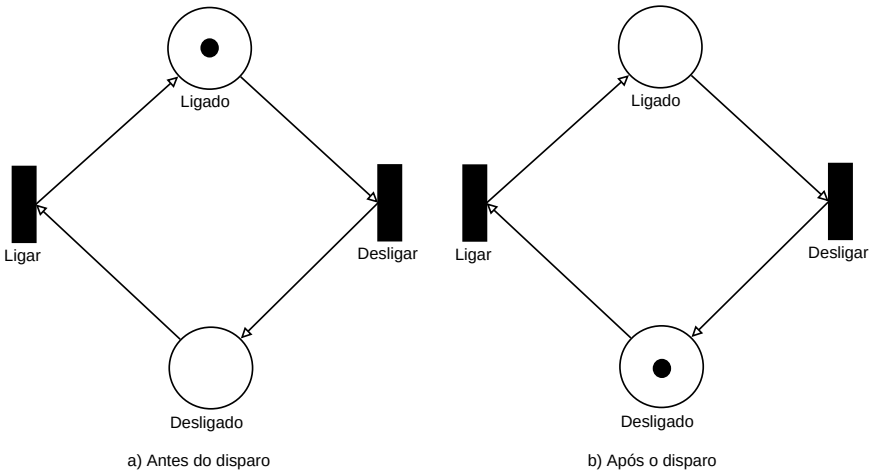

Figura 2.4: Exemplo do funcionamento de uma lanterna modelado em rede de Petri .

2.4.1 Redes de Petri Estocásticas

Este trabalho adota uma extensão particular das redes de Petri, as redes de Petri Estocásticas (SPN) [46], que permitem a associação de tempo às transições, e o respectivo espaço de estados pode ser convertido em uma cadeia de Markov de tempo contínuo (CTMC) [47]. Uma rede de Petri estocástica adiciona o tempo ao formalismo das redes de Petri, com a diferença de que os tempos associados às transições temporizadas são distribuídos exponencialmente, enquanto o tempo associado às transições imediatas é zero [27]. Além disso, SPN permite a adoção de técnicas de simulação para a obtenção de métricas de confiabilidade como uma alternativa para a geração da cadeia de Markov [29].

Os elementos de uma SPN (ver Figura 2.5) são os mesmos de uma PN: arcos, lugares, transições e *tokens* com a adição de mais dois elementos: transição temporizada (ver Figura 2.5-e) e o arco inibidor (ver Figura 2.5-f). O arco inibidor é um tipo de arco especial que possui um pequeno círculo branco em uma das extremidades em vez de uma seta, e, geralmente, é usado para desativar transições se houver *tokens* (ou ativar quando não houver) presentes em um lugar [48]. Níveis diferentes de prioridade podem ser atribuídos às transições imediatas. Além disso, as transições imediatas tem maior prioridade de disparo que as transições temporizadas[27]. As prioridades podem solucionar ocorrências de confusão [49]. As probabilidades de disparo associadas às transições imediatas podem solucionar situações de conflito[50, 49].

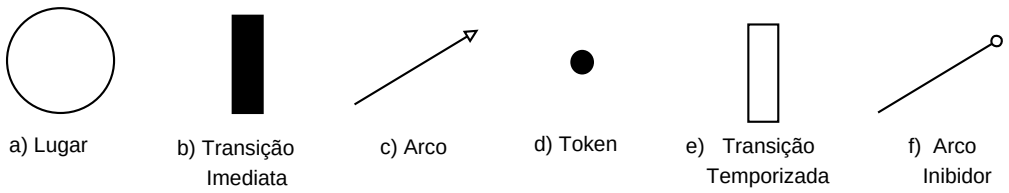

Figura 2.5: Elementos de uma Rede de Petri Estocástica.

A representação formal de um modelo em rede de Petri estocástica (SPN) é a 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$ [27, 45], onde:

- $P = \{p_1, p_2, ..., p_m\}$ é o conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é o conjunto finito de transições temporizadas, onde $P \cup T = \emptyset$;

- $I \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações);
- $\Pi \in \mathbb{N}^m$ é um vetor que associa o nível de prioridade a cada transição;
- $G \in (\mathbb{N}^n \to \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada a marcação do lugar à cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, W, Markdep, Policy, Concurrency)^m$ compreende o conjunto de atribuições para as transições, onde:
  - $Dist \in \mathbb{N}^m \to \mathcal{F}$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, sendo que o domínio de $\mathcal{F}$ é $[0, 1)$ (a distribuição pode ser dependente de marcação);
  - $W \in \mathbb{N}^m \to \mathbb{R}^+$ é a função peso, que associa um peso (w_t) às transições imediatas e uma taxa λ_t às transições temporizadas, onde:
    $$W(t) = \begin{cases} w_t \geq 0, & \text{se } t \text{ é uma transição imediata;} \\ \lambda_t > 0, & \text{caso contrário.} \end{cases}$$
  - $Markdep \in \{constant, enabdep\}$ define se a distribuição de probabilidade associada ao tempo de uma transição é constante (*constant*) ou dependente de marcação (*enabdep*);
  - $Policy \in \{prd, prs\}$ é a política de preempção adotada pela transição (*prd - preemptive repeat different* corresponde à *resampling* e *prs -preemptive resume* possui significado idêntico à *age memory policy*);
  - $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica do servidor único e is significa a semântica do servidor infinito (mesmo sentido dos modelos de fila).

Um exemplo de modelo em SPN para a obtenção da disponibilidade, é o modelo *Cold Standby*, que consiste num sistema redundante de espera composto por um módulo de reposição não ativo é para ser ativado quando o módulo ativo principal falhar. Para que o

módulo reserva entre em funcionamento, é necessário um período entre a ativação. Por esse motivo, a redundância *cold standby* é denominada passiva [29]. A Figura 2.6 representa o modelo SPN deste sistema modelado a partir de um conjunto composto por um módulo principal (PRI) e um módulo de *backup* (BKP), que inclui quatro lugares: *PRI_ON*, *PRI_OFF*, *BKP_ON*, *BKP_OFF*. Esses lugares representam os estados operacionais e os estados de falhas dos principais módulos e peças, respectivamente [51]. O módulo de backup é inicialmente desativado, e somente após a falha do módulo principal é que o módulo de backup é acionado. Quando o módulo principal falha, a transição *TACT* é acionada para ativar o módulo de backup [51].

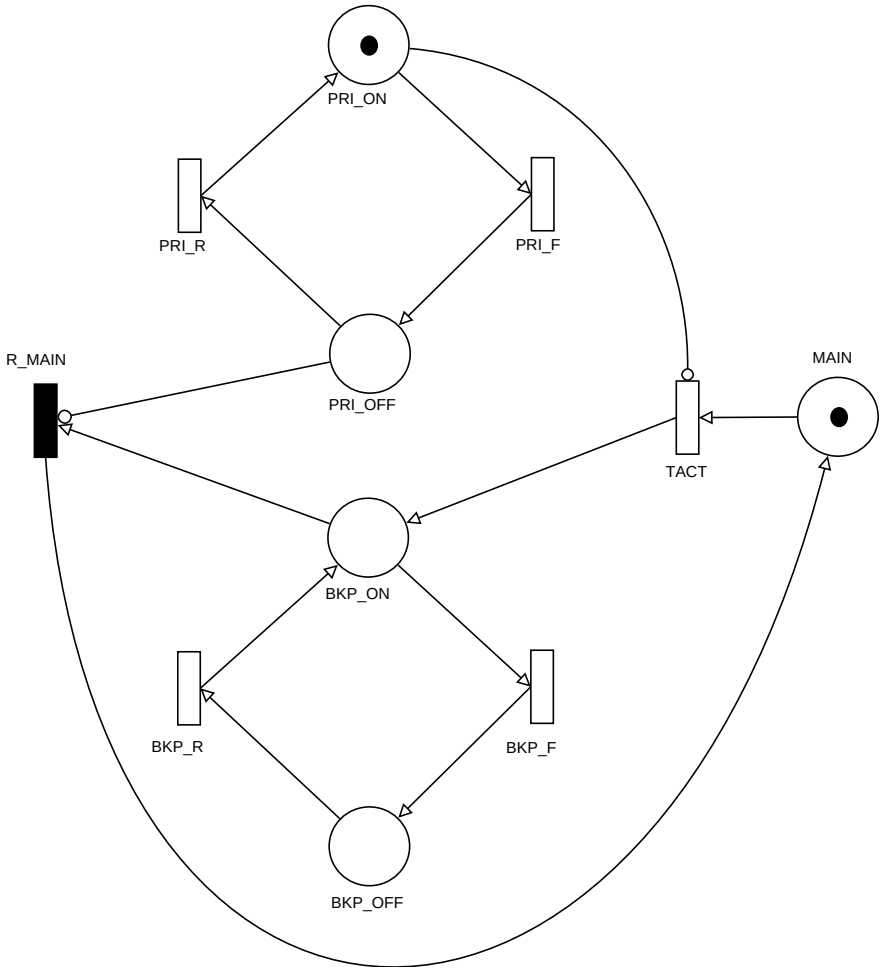

Figura 2.6: Cold Standby Model.

O lugar *PRI_ON* representa o estado de funcionamento do módulo principal. Para que o este módulo falhe, a transição *PRIS_F* tem que ser disparada. Uma vez disparada um *token*

é removido do lugar *PRI_ON* e colocado no *PRI_OFF*. Na ausência de *tokens* em *PRI_ON* a transição *TACT* é habilitada. Com o disparo desta transição o módulo reserva (BKP) é ativado removendo um *token* do lugar *MAIN* e o colocando em *BKP_ON*.

Para que o módulo principal esteja desativado, um *token* está no lugar *PRI_OFF* habilitando a transição *PRI_R* para reparar o módulo principal. Disparada esta transição, o *token* é deslocado de *PRI_OFF* para *PRI_ON*. Observando que o módulo reserva está ativado com um *token* em *BKP_ON* a transição instantânea *R_MAIN* é automaticamente disparada removendo o *token* de *BKP_ON* para *MAIN* desativando o módulo reserva.

2.5 Diagrama de Blocos de Confiabilidade (RBD)

O diagrama de bloco de confiabilidade (RBD) [30] é um modelo combinatório inicialmente proposto como uma técnica para o cálculo da confiabilidade de um sistema usando diagramas de blocos intuitivos. Essa técnica também foi estendida para calcular outras métricas de confiabilidade, tais como a disponibilidade e capacidade de manutenção [31]. Os diagramas de blocos de confiabilidade fornecem uma descrição gráfica dos componentes e conectores do sistema, que podem ser adotados para determinar o estado geral do sistema, dado o estado de seus componentes [11].

A estrutura do RBD estabelece a interação lógica entre os componentes que definem quais combinações de elementos falhos e ativos podem sustentar a operação do sistema. Mais especificamente, o sistema é representado por subsistemas ou componentes conectados de acordo com sua função ou relação de confiabilidade [32]. Na disposição em série (ver Figura 2.7), o sistema somete funcionará se todos os seus componentes estiverem ativos. Para um arranjo com n componentes, a confiabilidade (P_s) é obtida através da Equação 2.4.

$$P_s = \prod_{i=1}^{n} P_i \tag{2.4}$$

onde P_i representa a confiabilidade $R_i\,(t)$ (disponibilidade instantânea ($A_i(t)$) ou disponibilidade de estado estacionário (A_i)) do bloco *bi*.

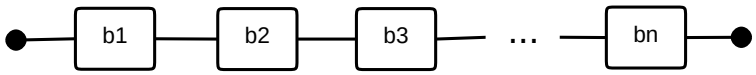

Figura 2.7: Diagrama de Blocos de Confiabilidade de um sistema em série.

Na composição em paralelo (ver Figura 2.8), o sistema funcionará se um de seus componentes estiver funcionando. Para uma arranjo com n componentes, a confiabilidade (P_p) é obtida através da Equação 2.5.

$$P_p = 1 - \prod_{i=1}^{n}(1 - P_i) \tag{2.5}$$

onde P_i representa a confiabilidade $R_i\,(t)$ (disponibilidade instantânea ($A_i(t)$) ou disponibilidade de estado estacionário (A_i)) do *bi*.

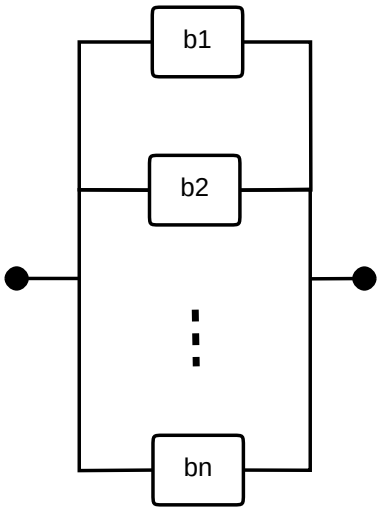

Figura 2.8: Diagrama de Blocos de Confiabilidade de um sistema em paralelo.

Nos modelos RBD, é possível representar um componente físico no modo operacional por um bloco. Se houver pelo menos um caminho conectando pontos de entrada e saída, o sistema está funcionando corretamente [31, 52].

2.6 Modelo de Fluxo de Energia (EFM)

O Modelo de Fluxo de Energia (EFM) tem o objetivo de representar o fluxo de energia elétrica entre os componentes do sistema, considerando a eficiência e a capacidade máxima que cada componente pode fornecer ou extrair [4]. O EFM é representado por um grafo acíclico dirigido em que os componentes são modelados como vértices e as respectivas ligações correspondem às arestas [39].

A Figura 2.9 ilustra um exemplo de um EFM. A intensidade do fluxo elétrico entre os componentes é indicada pelos pesos nas arestas. A representação gráfica dos modelos EFM tem por peso padrão um (1) nas arestas. No modelo, retângulos representam o tipo de dispositivo, e as etiquetas correspondem ao nome de cada item. Neste trabalho, o EFM é utilizado para calcular a energia total essencial para fornecer a potência necessária para o ambiente de TI (representado por *TargetPoint1* na Figura 2.9) e, também, para computar o custo de aquisição e operacional do ambiente sob análise.

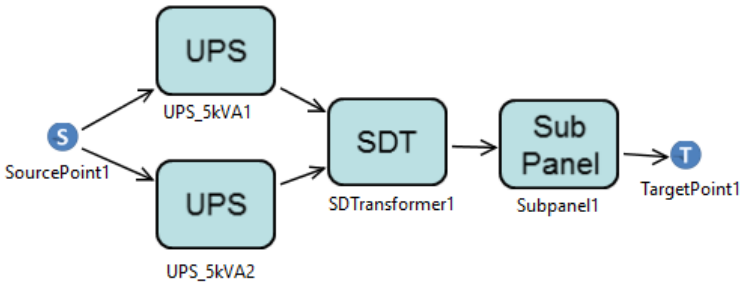

Figura 2.9: Um exemplo de EFM.

No sistema em avaliação, os componentes necessários podem não ser capazes de satisfazer a demanda do sistema por energia elétrica mesmo estando dispostos corretamente [11]. A Figura 2.10-a mostra um exemplo de uma infraestrutura elétrica. Supondo que a potência exigida pelo sistema de TI do *data center* é de 30 kW (valor associado ao *target* T) e a capacidade de potência máxima (Figura 2.10-b) dos nós internos (componentes), *UPSs* e *power strips*, é de 20 kW para cada. A Figura 2.10-c representa um possível fluxo de energia, no qual cada UPS fornece potência de 15 kW, que é transferida para os *Subpanels*(15 kW). No exemplo da Figura 2.10-d, apenas um *Subpanel* é considerado. Este sistema suporta apenas 20 kW de potência exigida (*target* T). Portanto, o sistema não é capaz de lidar com

a potência exigida[22].

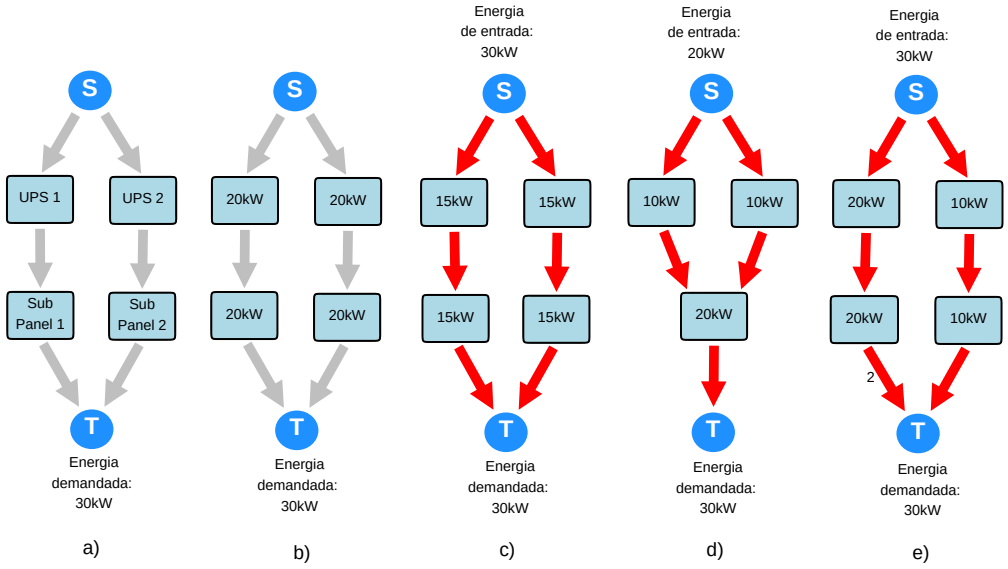

Figura 2.10: (a) Exemplo de infraestrutura de carga; (b) Capacidade máxima de energia; (c) Fluxo de energia realizado com sucesso; (d) Fluxo de energia falho; e (e) exemplo de peso nas arestas.

A Figura 2.10-e representa um sistema com dois *Subpanels*. O *Subpanel1* é capaz de fornecer duas vezes mais energia que o *Subpanel2* de vido ao peso associado a aresta do grafo. Além disso, os algoritmos são propostos para calcular o impacto de custo e sustentabilidade através do EFM[22].

2.6.1 Calculando a exergia operacional

O consumo de exergia operacional pode ser entendido como a energia dissipada por cada item de equipamento que não pode ser, teoricamente, convertido em trabalho útil. A seguinte equação representa a exergia operacional do sistema [11].

$$Ex_{op} = \sum_{i=1}^{n} Ex_{op_i} \times T \times (Disp + \alpha(1 - Disp)) \tag{2.6}$$

onde Ex_{op_i} é a exergia operacional de cada dispositivo, T é o período de análise, $Disp$ é a

disponibilidade do sistema e α é o fator que representa a quantidade de energia que continua a ser consumida depois que um componente falha.

2.6.2 Calculando o custo

O custo é a soma do custo de aquisição (CA) com o custo operacional (CO). O custo de aquisição é calculado pelo somatório dos preços de varejo (P_a) dos equipamentos que compõem a arquitetura conforme a Equação 2.7 [11].

$$CA = \sum_{i=1}^{n} P_a \tag{2.7}$$

Já o custo operacional, para este trabalho, considera apenas o custo referente ao consumo de energia do sistema de acordo com a Equação 2.8 [22]. Outros fatores, no entanto, também podem ser incluídos .

$$CO = E_{cons} \times T \times P_{energia} \times (Disp + \alpha(1 - Disp)), \tag{2.8}$$

onde E_{cons} é a energia elétrica consumida pelo sistema, T é o período assumido; $P_{energia}$ corresponde ao preço da energia, $Disp$ corresponde a disponibilidade do sistema e α é o fator que representa a quantidade de energia que continua a ser consumida depois que um componente falha.

2.7 Otimização

Otimização é o ato de encontrar a melhor solução nas circunstâncias dadas [53]. As técnicas de otimização possuem uma amplo conjunto de aplicações considerando que a maioria das empresas está engajada na resolução de problemas de otimização. Podem ser modelados como problemas de otimização, muitos dos problemas práticos e teóricos em engenharia, economia e planejamento. Por exemplo, empresas de telecomunicações interessadas

em otimizar o custo e a qualidade do serviço em seu projeto das redes de comunicação; para empresas do mercado financeiro tem o objetivo de obter a melhor estratégia de alocação de ativos para otimizar o lucro e o risco; empresas de distribuição visa de otimizar a alocação das entregas (rotas) de forma a minimizar a distância percorrida pelos veículos [54].

2.7.1 Otimização multi-objetivo

Grande parte dos problemas reais necessitam de uma otimização simultânea de objetivos distintos. Enquanto que uma solução ótima é facilmente encontrada na otimização mono-objetivo, isto não ocorre para a otimização de diversos objetivos [55]. É interessante ressaltar que o problema pode estar passivo de restrições e que os objetivos podem ser conflitantes entre si, fazendo com que o conceito de otimalidade definido em otimização mono-objetivo não possa ser utilizado [56, 57].

O conceito de otimalidade, em otimização multi-objetivo, baseia-se na noção introduzida por Francis Ysidro Edgeworth em 1881 [58] e na generalização feita por Vilfredo Pareto em 1896 [59] [60]. O resultado de uma otimização multi-objetivo é geralmente um conjunto de soluções caracterizado pelo comprometimento entre os diferentes objetivos que se deseja fazer a otimização. No universo de busca, alguma solução melhor que elas quando considerados todos os objetivos simultaneamente. Portanto, o principal alvo da otimização multi-objetivo consiste em obter as soluções Pareto-ótimas e, em consequência, conhecer o conjunto das combinações possíveis entre os objetivos [55].

A ideia de solução também difere da otimização mono-objetivo. Em um problema multi-objetivo de minimização, a solução é formada por um conjunto de soluções que apresentam uma combinação entre os objetivos. Um conjunto de soluções é um conjunto Pareto-ótimo se, para cada solução, não exista nenhuma outra solução factível capaz de reduzir o valor de um dos parâmetros do problema sem que, ao mesmo tempo, aumente pelo menos um dos demais parâmetros do problema [60].

A otimização multi-objetivo possui alguns conceitos, os principais são [55]:

- **Função objetivo**: também conhecida como função *fitness*, é uma equação matemática

que representa que melhora se deseja em uma solução;

- **Parâmetros**: são as variáveis da função objetivo, sendo ajustadas durante o processo de otimização com o objetivo de obter as soluções ótimas;
- **Espaço de busca**: corresponde ao espaço ou domínio (delimitado ou não) que contém os valores dos parâmetros. A dimensão do domínio é definida pela quantidade de parâmetros envolvidos nas soluções (por exemplo, se as soluções do conjunto são formadas por três parâmetros, o domínio é tridimensional);
- **Espaço de objetivos**: conjunto imagem do espaço de busca definido pelos valores possíveis das funções objetivo;
- **Restrições**: especificações do problema que delimitam o espaço de busca;
- **Domínio realizável**: região do espaço de busca na qual as restrições são respeitadas. É denominado também como espaço admissível, viável ou factível;
- **Domínio não-viável**: região do espaço de busca na qual as restrições são violadas.

Este trabalho foca em uma abordagem multi-objetiva baseada em algoritmos genéticos.

2.8 Algoritmos Genéticos

Os algoritmos genéticos (AG) são algoritmos de busca baseados na lógica da seleção e genética natural. Embora randomizados, os algoritmos genéticos não são uma caminhada aleatória simples. De maneira eficiente, eles exploram informações históricas para especular sobre novos pontos de busca com desempenho melhorado [61].

Algoritmos genéticos foram desenvolvidos por John Holland [62], seus colegas e seus alunos na Universidade de Michigan. Os objetivos da pesquisa foram: (i) abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais e (ii) projetar software de sistemas artificiais que possui mecanismos de sistemas naturais significativos [61].

Os algoritmos genéticos foram propostos com objetivo de aplicar a teoria da evolução das espécies de Darwin na computação. Esses algoritmos utilizam conceitos da evolução

biológica como genes, cromossomos, cruzamento, mutação e seleção, procurando aprofundar o conhecimento em processos de adaptação em sistemas naturais e, baseado neles, desenvolver sistemas artificiais (simulações computacionais) que mantenham a mesma lógica dos mecanismos originais [63]. A lógica funciona de forma semelhante ao processo de cruzamento biológico de cromossomos que compartilham informação genética para criar um novo indivíduo. No final, chegam a obter um indivíduo de alta adaptação, que no jargão de matemática não significa uma solução ótima, mas sim a "melhor solução" encontrada [61].

Os algoritmos genéticos se diferenciam dos algoritmos tradicionais de otimização por serem [64]:

- estocásticos, duas execuções distintas podem resultar em soluções distintas;
- de busca múltipla, dos quais é possível obter várias soluções;
- de convergência pouco sensível à população inicial, não existem restrições no espaço de busca;
- algoritmos intrinsecamente paralelos capazes de operar simultaneamente com várias soluções, o que assegura geralmente um funcionamento mais rápido nas execuções;
- algoritmos que resultam poucas soluções falsas.

Com o surgimento da genética, do avanço da tecnologia, da microbiologia, dentre outros estudos, e juntamente com o Darwinismo, surgiu o neo-darwinismo. Para este, os preceitos básicos do processo de evolução das espécies são [65]:

- Indivíduos de mesmas ou diferentes espécies disputam continuamente por limitados recursos presentes no meio ambiente;
- Dentre os vários concorrentes presentes em um determinado meio, alguns, por conta de suas características específicas, possuem uma melhor chance (maior probabilidade) de sobrevivência. Tais indivíduos são ditos mais adaptados ao ambiente;
- Indivíduos mais adaptados possuem uma maior probabilidade de sobrevivência, e consequente reprodução;

- Visto que, no processo de reprodução, um grande número de características dos pais são repassadas aos filhos, indivíduos que se reproduzem mais tendem a propagar mais significativamente suas características nas gerações subsequentes;
- Logo, ao longo do processo de evolução, características mais desejáveis tendem a se propagar na espécie, aumentando assim o grau de adaptação desta como um todo;
- O processo de reprodução não ocorre sem falha - durante a replicação e transmissão dos genes aos novos indivíduos criados, o fenômeno conhecido como mutação pode ocorrer. Este fenômeno é geralmente prejudicial ao indivíduo, mas em alguns casos pode incorporar a ele uma característica desejável não contida no conjunto de genes dos seus pais. Desta forma a natureza adquire a capacidade de explorar um número maior de combinações e possibilidades.

Essa ideologia dos algoritmos genéticos pode ser replicada na construção de algoritmos computacionais com o objetivo de se otimizar a solução, da forma mais viável possível, para um determinado problema. Essa otimização pode ocorrer, por exemplo, através da evolução de populações de soluções modeladas como cromossomos artificiais [66]. Gera-se uma população inicial aleatória de soluções para determinado problema com características próprias (genes), denominadas indivíduos codificados por cromossomos [67]. Existe um valor de aptidão associado a cada indivíduo atribuído por uma função objetivo. Quanto melhor a solução que o indivíduo representa, maior é a sua aptidão e suas chances de sobreviver e se reproduzir [68]. Durante a evolução, várias gerações sucessivas são criadas e, em cada uma delas, alguns indivíduos são selecionados de acordo com a aptidão de suas características, escolhendo os melhores baseado nas que seriam as mais viáveis soluções [67]. Estes indivíduos darão origem à próxima geração que irá apresentar novas características adquiridas através do cruzamento de seus genes.

A analogia entre Algoritmos Genéticos e o sistema natural é representada através da Tabela 2.3 [66].

Natureza	Algoritmos Genéticos
Cromossomo	Palavra Binária, Vetor, etc
Gene	Característica do problema
Alelo	Característica, Valor
Loco	Posição da palavra, Vetor
Genótipo	Estrutura
Fenótipo	Estrutura subjetiva do problema
Indivíduo	Solução
Geração	Ciclo

Tabela 2.3: Analogia entre Algoritmos Genéticos e o sistema natural.

Para se otimizar ao máximo um determinado problema, todo o processo, ilustrado na Figura 2.11, deve ser repetido por várias gerações (ciclos) com o objetivo de se chegar a uma solução ótima. O fluxograma do algoritmo genético pode ser representado pelas etapas descritas na Figura 2.11.

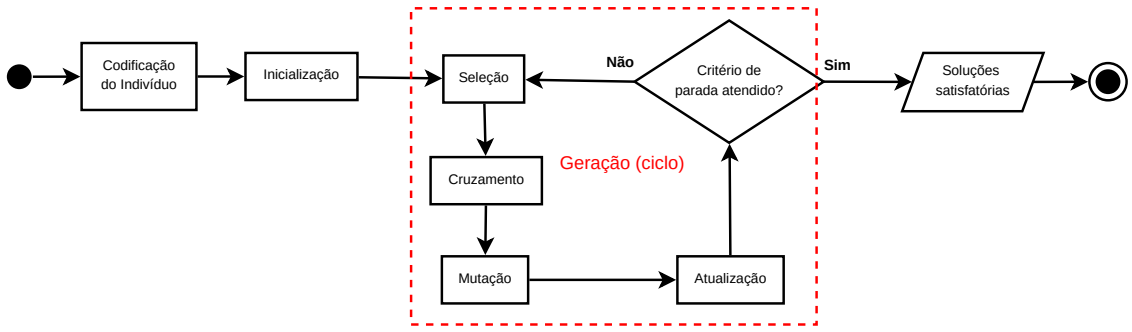

Figura 2.11: Fluxograma do algoritmo genético.

Um algoritmo genético passa pelas seguintes etapas [65]:

- **codificação do indivíduo**: Inicialmente, cria-se um indivíduo para representar da solução do problema [69];
- **inicialização**: basicamente produz uma população de soluções aleatórias (indivíduos) para o problema a ser otimizado;

- **avaliação**: avalia-se a aptidão dos indivíduos através de uma função *fitness*, ou função objetivo, para se estabelecer a qualidade de cada solução gerada resultando numa pontuação para cada indivíduo;
- **seleção**: indivíduos são selecionados para a reprodução a partir de sua aptidão;
- **cruzamento**: características das soluções são recombinadas, gerando novos indivíduos. Um ponto de corte é determinado aleatoriamente para, a partir dele haver a troca dos genes entre os indivíduos;
- **mutação**: existe uma probabilidade de que características dos novos indivíduos resultantes do processo de cruzamento sejam alteradas, acrescentando assim variedade à população. Normalmente são selecionados, aleatoriamente, dois genes que são permutados entre si;
- **atualização**: os indivíduos criados nesta geração são inseridos na população;
- **finalização**: verifica se as condições de encerramento da evolução foram atingidas, retornando para a etapa de avaliação em caso negativo e encerrando a execução em caso positivo.

A seguir são explanados alguns dos principais conceitos de Algoritmos genéticos.

2.8.1 Indivíduos

Os indivíduos representam as possíveis soluções para o problema em questão [70, 69]. A solução precisa ser codificada para permitir a sua manipulação pelo algoritmo genético e, após o fim do processo evolutivo, o indivíduo necessita ser decodificado para obter a solução final. Uma das partes cruciais do algoritmo genético é a escolha da codificação do indivíduo, uma vez que esta escolha impacta no desempenho. O tipo de codificação a escolhido depende diretamente do problema a ser solucionado [70].

Existem várias formas de representação de um indivíduo em um AG. A mais simples e comum é a representação binária de tamanho fixo, na qual um indivíduo é composto por

uma cadeia de bits (cromossomo) composto por valores em zero (0) ou um (1). [71]. As características mais importantes do indivíduo são [70, 65]:

- **Genótipo**: informação presente na estrutura de dados dos indivíduos (codificação);
- **Fenótipo**: resultado da decodificação do indivíduo;
- **Grau de adaptação ou aptidão (*fitness*)**: representa o quão boa é a solução representada por um indivíduo para problema proposto. É calculado por uma função objetivo ou *fitness* (geralmente denotada $f_O(x)$);

É interessante ressaltar que, em um algoritmo genético convencional, um indivíduo nunca sabe o significado das informações que ele carrega [72]. Além disso, na maioria dos AGs, assume-se que cada indivíduo é formado por um único cromossomo, fato pelo qual é normal usar os termos indivíduo e cromossomo de maneira indistinta [67].

No exemplo a seguir, o objetivo é encontrar o valor máximo do gráfico representado na Figura 2.12. A função *fitness* ou função objetivo é representada pela função que gera a parábola do gráfico, ou seja, $f_O(x) = -x^2 + 180x - 179$. O intuito do exemplo é encontrar o maior valor de $f_O(x)$ utilizando algoritmos genéticos.

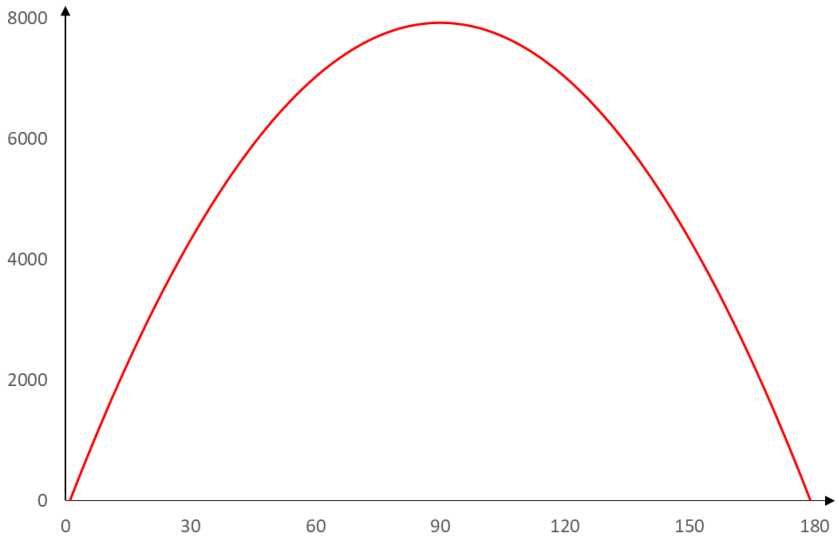

Figura 2.12: Exemplo de problema.

Para cada indivíduo x, obtém-se uma aptidão através da referida função *fitness* conforme exemplo demonstrado na Tabela 2.4.

Indivíduo (x)	Aptidão ($f_O(x)$)
1	0
25	3696
76	7725
90	7921
132	6157
171	1360
179	0

Tabela 2.4: Exemplo de indivíduos e suas respectivas aptidões.

Codificando o indivíduo como uma estrutura binária é obtido o cromossomo ilustrado na Figura 2.13. Neste exemplo é apresentado um indivíduo, para $x = 76$, transformado em binário e ajustado ao genótipo do cromossomo.

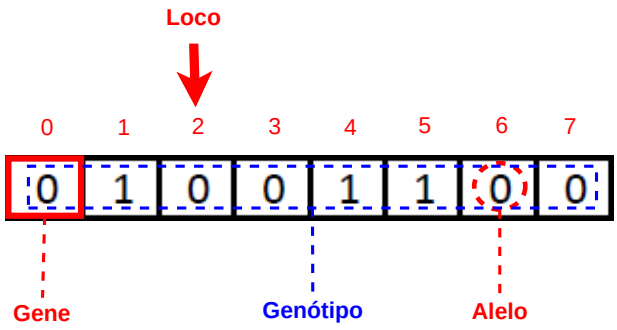

Figura 2.13: Exemplo de Cromossomo

Pode-se observar, neste exemplo, que o fenótipo do indivíduo corresponde ao valor em decimal de 76. O genótipo é a transformação deste número em binário (no exemplo, 01001100) para que possa ser representado por um vetor (cromossomo, ver Figura 2.13). O gene é a característica, que no caso possui um espaço de busca de 0 ou 1. Os $0s$ ou $1s$, propriamente ditos, são os valores dessa característica denominados alelos. O loco é a posição no vetor cromossômico em que se encontra um determinado gene.

2.8.2 Populações

A evolução do AG só é possível pela dinâmica populacional [61]. Essa dinâmica atua propagando características importantes a gerações seguintes (cruzamento) enquanto novas são testadas (mutação) [65]. Ao propagar estas características desejáveis, um AG manipula a frequência com que determinadas sequências de genes dos indivíduos estão presentes nas populações [65, 70].

As populações possuem as seguintes características[65, 70]:

- **Geração**: representa cada vez que a população passou pelo processo evolutivo (seleção, reprodução, mutação e atualização);
- **Média de Adaptação**: média dos resultados do valor de aptidão de cada indivíduo. Considerando que a aptidão é obtida por $f_O(x)$ a média de adaptação será:

$$M_A = \frac{\sum_{i=1}^{n} f_O(i)}{n} \tag{2.9}$$

  sendo n o tamanho da população

- **Grau de Convergência**: representa a proximidade da média de adaptação da geração atual em relação às gerações anteriores. A convergência da população para um valor ótimo de adaptação é de responsabilidade do AG. Um fato negativo ligado ao grau de convergência é a convergência prematura que ocorre quando uma população converge rapidamente para uma média de adaptação sub-ótima;
- **Diversidade**: representa a variação entre os genótipos da população. Ela é crucial para ampliar o espaço de busca. Uma baixa diversidade está ligada diretamente à convergência prematura;
- **Elite**: é formada pelos melhores (mais adaptados) indivíduos da população. O elitismo é uma técnica trivial nos AGs, na qual os m melhores indivíduos são mantidos para a próxima geração.

2.8.3 Inicialização ou População Inicial

A inicialização de um AG compreende a geração da sua população inicial de soluções, frequentemente realizada de maneira aleatória, podendo ocorrer situações em que é necessária uma seleção heurística da papulação de maneira a introduzir, logo de início, um ou mais indivíduos "interessantes" [73]. A inicialização é responsável por definir a quantidade inicial de indivíduos pertencentes à população inicial e quais serão os valores de seus genes na primeira geração [74]. Geralmente, essa etapa é feita por meio de funções randômicas para gerar indivíduos, com o intuito de obter uma maior diversidade da população. Existem alternativas ao processo aleatório com o objetivo de superar deficiências quando quando se deseja um melhor desempenho e/ou quando o indivíduo representado é mais complexo. Tradicionalmente, os operadores de inicialização são [61, 75]:

- **Inicialização aleatória uniforme**: é atribuído, a cada gene do indivíduo, um valor do conjunto de alelos, sorteado de maneira uniforme e aleatória;
- **Inicialização aleatória não uniforme**: certos valores a serem atribuído aos genes tendem a serem escolhidos com uma maior frequência em relação aos demais valores;
- **Inicialização aleatória com *dope***: são inseridos indivíduos otimizados na população. A presença desses indivíduos otimizados pode tendenciar o AG a uma convergência prematura;
- **Inicialização parcialmente enumerativa**: indivíduos são inseridos de determinada maneira na qual a população inicia o processo evolutivo possuindo todos os combinações possíveis de uma determinada ordem.

Diversos trabalhos comprovam que a inicialização, normalmente, não é crítica, contanto que a população inicial possua cromossomos suficientemente variados [76, 61].

2.8.4 Avaliação

No processo de avaliação, cada indivíduo é avaliado para que seja definida a sua aptidão [70]. A avaliação da população é feita pela função objetivo que é utilizada para determinar

o quão boa uma solução é para resolução efetiva de um problema. A avaliação somada à forma de codificação do indivíduo, são, normalmente, as únicas características do AG com relação direta ao domínio do problema [69].

Para cada indivíduo, A função objetivo retorna uma medida de quão bem adaptado ao ambiente ele está, ou seja, quanto maior o valor retornado pela função, maior é a expectativa do indivíduo sobreviver no ambiente e se reproduzir, transmitindo parte de seus genes (características) às gerações posteriores [67].

É necessário lembrar, entretanto, que a escolha da função objetivo é para grande parte das aplicações a fase crítica do processo evolutivo, pois ela deverá ser avaliada para cada indivíduo da população durante o processo evolutivo [73].

2.8.5 Seleção

A etapa de seleção é encarregada pela continuidade das melhores características em gerações futuras da espécie [70]. A seleção é um processo dirigido, funciona de maneira determinística ou quase determinística, sendo também, um processo cumulativo, onde melhorias encontradas são transmitidas para as futuras gerações [77].

Nesta etapa, os indivíduos mais aptos são selecionados para o cruzamento. Esses indivíduos são escolhidos apoiados em sua aptidão. Os métodos de seleção essenciais são [61, 75]:

- ***ranking***: os indivíduos são dispostos em um *ranking* conforme a aptidão de cada um. A probabilidade de um indivíduo ser selecionado é atribuída de acordo com a posição que ocupa no ranking;
- **roleta**: este método procede da seguinte forma: é calculado o somatório das aptidões da população ($total$). Em seguida, é escolhido aleatoriamente um valor i de forma que $i \in [0; total]$. Feito isso, é selecionado o indivíduo x que se situa no intervalo do somatório compatível com i [78];
- **Uniforme**: a probabilidade de ser escolhido é igual para todos os indivíduos da população [79]. Como efeito colateral, a taxa de convergência é lenta;

- **torneio**: Este método não confere aos indivíduos probabilidades de maneira explícita, a ideia é propiciar um torneio entre um grupo de k ($k \geq 2$) indivíduos aleatoriamente selecionados da população. k é o tamanho do torneio. Assim, o indivíduo com o maior valor de aptidão no grupo é selecionado para reprodução [69, 73]. Quanto maior o valor de k, maior a pressão seletiva, em outras palavras, maior a rapidez com que os indivíduos mais aptos dominam a população, acarretando no descarte dos menos aptos[69]. Dentre as vantagens da seleção torneio estão [73]:

  - não resulta em convergência prematura;
  - combate a inação;
  - esforços computacionais extras se tornam desnecessários, tais como os ordenamentos;
  - torna-se desnecessária a aptidão explícita;

Para este trabalho foi adotado o método de seleção **Torneio** por ser um dos modelos mais simples de implementar possuindo resultados satisfatórios [73].

2.8.6 Reprodução ou Cruzamento

Os indivíduos, uma vez selecionados, são submetidos ao cruzamento (*crossover*), onde os genes dos pais são combinados para geração de filhos [65, 70]. A etapa do cruzamento permite a troca de genes entre dois indivíduos (pais), combinando suas características de maneira que exista uma expectativa significativa dos novos indivíduos produzidos (filhos) serem mais adaptados que seus pais [71]. No cruzamento escolhe-se aleatoriamente um ponto de corte nos pais, dividindo cada cromossomo em duas partes, de maneira a recombinar essas partes para gerar os filhos. Dado dois cromossomos x e y com comprimento L, escolhe-se, aleatoriamente, um número P (ponto de corte) tal que $0 < P < L$. O filho s_0 receberá os genes de 0 até o índice 2 (P) do cromossomo x, e os genes de $P + 1$ até L do cromossomo y. O filho s_1 herdará os genes de $P + 1$ até L do cromossomo x e os genes de 1 até P do cromossomo y conforme exemplificado na Figura 2.14. Existem outros tipos de cruzamento, mais detalhes em [78, 61].

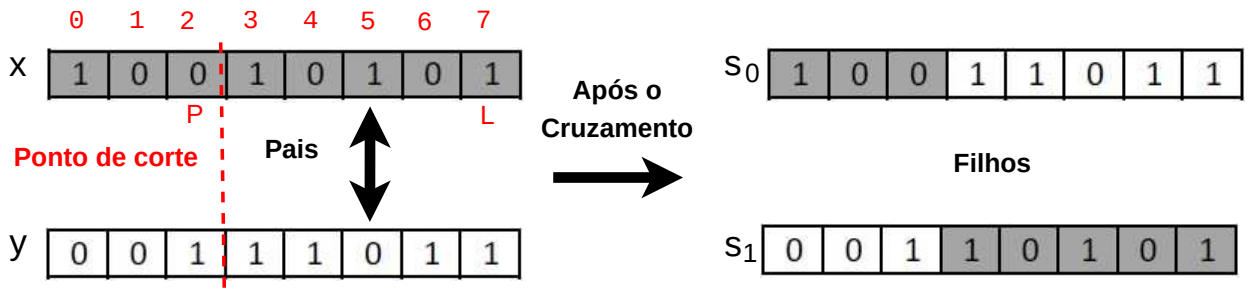

Figura 2.14: Cruzamento.

2.8.7 Mutação

A mutação é responsável por introduzir pequenas mudanças na estrutura cromossômica do indivíduo, estando condicionada a uma probabilidade (taxa de mutação) para que as alterações na estrutura ocorram. A mutação é fundamental por garantir uma diversidade dos indivíduos da população, possibilitando uma maior exploração do espaço de busca. Entretanto, uma taxa de mutação elevada pode acarretar na perda de características fundamentais da população [78, 61, 75]. A mutação adotada neste foi a mutação flip. De acordo com a Figura 2.15, o gene que irá sofrer mutação é sorteado tendo o seu valor substituído por outro valor escolhido, aleatoriamente, dentro do espaço de busca. No exemplo de um indivíduo de representação binária, a mutação troca o valor do gene de 0 (zero) para 1 (um) e vice-versa [70];

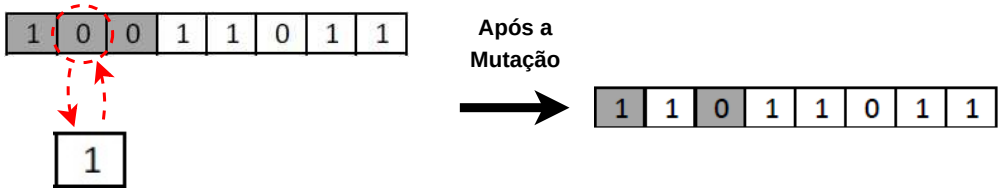

Figura 2.15: Mutação *flip*.

O objetivo da mutação é introduzir novas informações às populações, impedindo que elas fiquem saturadas de cromossomos similares, ao passo que visa elevar a variedade populacional possibilitando uma maior exploração do espaço de busca [73, 71]. Normalmente, a mutação é aplicada com pequenas probabilidades, pois resultariam em uma busca aleatória caso taxas

de probabilidade elevadas fossem aplicadas [69]. Em geral, para cada *bit* da representação, a probabilidade de acontecer mutação é da ordem de $0,1\%$ [80]. A taxa de mutação ideal está condicionada ao problema a ser solucionado, entretanto, predominantemente, a taxa de mutação varia entre $0,1\%$ e 10% [73].

2.8.8 Atualização ou Elitismo

Ao fim do cruzamento e da mutação, a etapa da atualização é encarregada de requalificar a população para a próxima geração. Essa atualização está condicionada à política empregada pelo AG. Na política de atualização tradicional o número de indivíduos da população se mantém o mesmo em todas as gerações, embora existam alternativas [61, 78].

Em outras abordagens, o total de indivíduos produzidos numa geração pode ser menor do que a população, a população pode variar seu tamanho a cada geração e o pré-requisito de inserção de novos indivíduos pode ser variado. Neste último, por exemplo, os indivíduos são introduzidos se o valor da aptidão do filho for melhor do que a do pai. Em outro exemplo, todos os filhos gerados são introduzidos mas são mantidos n melhores indivíduos (elitismo) [61, 78].

O elitismo é uma estratégia com o propósito de preservar as melhores soluções encontradas na geração atual. Em sua lógica mais trivial, o elitismo conserva os n_{elit} melhores indivíduos da geração atual, mantendo-os na população para a próxima geração sem nenhuma mudança. Dessa forma, os melhores indivíduos não são só passados de uma geração para outra, como também colaboram na geração dos novos indivíduos da nova geração [69].

2.8.9 Critério de parada ou Finalização

O critério de parada é uma etapa que aponta o fim da execução do algoritmo. Os critérios de parada estão diretamente condicionados ao problema a ser solucionado, variando desde o número de gerações produzidas até o grau de convergência da população atual [61]. O ideal seria que o algoritmo findasse logo que uma solução satisfatória fosse gerada, já que os AGs tratam de problemas de otimização [67].

Na maioria das otimizações, não se pode afirmar, precisamente, se uma dada solução representa um ótimo global. Como consequência, comumente é utilizado como critério de parada, um número máximo de gerações ou um tempo limite de processamento. Outro critério bastante utilizado é a ideia de estagnação, ou seja, quando nenhuma melhoria nos indivíduos da população é observada após o decorrer de várias gerações consecutivas (convergência) [67].

Capítulo 3

Trabalhos Relacionados

Vários trabalhos vêm sendo desenvolvidos com o objetivo de reduzir o consumo de energia e aperfeiçoar técnicas e modelos sustentáveis em *data centers*. Além disso, um outro aspecto que é levado em consideração é a necessidade de se ter o sistema sempre disponível, e assim, pesquisas relacionadas a esse tema também vem atraindo atenção da comunidade acadêmica e das empresas. Técnicas de otimização vem sendo aplicadas para otimizar tanto a disponibilidade, como o consumo e a sustentabilidade.

Callou et. al [22] usam uma abordagem de otimização multi-objetivo através do "procedimento de busca gulosa adaptativa randomizada" (*GRASP*) para otimização de arquiteturas de *data center*. Modelos em SPN, RBD e EFM foram através do ambiente Mercury e a partir de uma lista de equipamentos com diferentes parâmetros (MTTF, custo, eficiência energética), foi utilizado o algoritmo GRASP de maneira a maximizar a disponibilidade, minimizando o custo e o impacto ambiental. Diferente desse trabalho, essa dissertação faz uso de algorítimos genéticos como heurística de otimização que resulta sempre numa solução próxima da ótima, enquanto o *GRASP*, que é baseado em uma busca randômica, pode retornar uma solução não satisfatória para a otimização.

Ferreira et. al [39] propõem um algoritmo de distribuição de carga de energia (*PLDA*), baseado no algoritmo de *Ford-Fulkerson*, que minimiza o consumo de energia em um modelo proposto de fluxo de energia (EFM). A aplicabilidade do PLDA proposto foi verificada por um estudo de caso que analisa seis arquiteturas elétricas de nuvem privada observando

resultados significativos, como a redução no consumo de energia de 10,7% e uma redução no impacto ambiental. Porém, Ferreira et. al não focam na otimização da disponibilidade e do custo.

Yamini [81] propõe um algoritmo que utiliza o método de Pareto e a distância euclidiana para maximizar o uso dos recursos do processador, reduzindo assim o consumo de energia. Funções de custo são incorporados a heurística para encontrar possibilidades de poupar energia. O autor não propõe a otimização da disponibilidade que é um dos objetivos deste trabalho.

Mukherjee et. al [82] sugerem algoritmos para reagendar serviços de CPUs sobrecarregadas, bem como para identificar CPUs ociosas para reduzir o consumo de energia e o custo operacional. Os autores propõem uma computação em nuvem ecológica com foco na TI da nuvem. Entretanto os autores não focam na otimização da disponibilidade, e otimizam apenas o custo operacional. O presente trabalho proposto na dissertação, em contra partida, otimiza o custo total levando em consideração o custo do consumo de energia e o custo da aquisição dos equipamentos da infraestrutura elétrica de *data centers*.

Kar et. al [83] propõem a otimização, através de algoritmos genéticos, do consumo de energia, poluição ambiental, e do tempo de resposta das tarefas na alocação de recursos de escalonamento em *data centers*. Os autores não sugerem a otimização da disponibilidade e do custo, diferentemente deste trabalho.

Wang et. al [84] sugere um algoritmo de otimização denominado $CARPO$, para minimizar o consumo de energia das redes dos *data centers*. Através da consolidação dinâmica do tráfego de rede, o algoritmo objetiva a redução da quantidade de *switches* ativados nestas redes. A finalidade é diminuir os fluxos de dados em um pequeno grupo de *links* e desligar equipamentos de rede não utilizados para economizar energia. Para que isso aconteça, o fluxo de dados é modelado como um problema ótimo de atribuição de fluxo, no qual as restrições de tráfego são cumpridas enquanto o consumo de energia é minimizado.

Portaluri et. al [85] propõem um alocador de tarefas, baseado em algorítimos genéticos Multi-objetivos ($MOGA$) e recozimento simulado (SA), que reduz o consumo de energia nos servidores e *switches*. Os autores, porém, não propõem a otimização da disponibilidade nem

focam na otimização do custo considerando o consumo de energia e o valor de aquisição dos equipamentos.

Sharma et. al [86] propõem um *Algoritmo Genético Melhorado* (IGA) para a otimização de recursos subutilizados de *data centers* clássicos através de uma previsão de carga com a finalidade de reduzir o custo, o consumo de energia, o desperdício de recursos. Este IGA consiste numa modificação na geração da população inicial, gerando a melhor solução para a população inicial. Sharma et. al, entretanto, não sugerem a otimização da disponibilidade que é um dos focos deste trabalho.

Tham e Ang [87], para obterem a disponibilidade do *cluster* do *data center*, propõem modelos de Cadeias de Markov de Tempo Contínuo (CTMC). Uma abordagem apoiada na equação de otimização de *Bellman* [88] e algoritmos baseados em busca gulosa foi proposta para melhorar a disponibilidade do sistema ao determinar a política de máquina secundária ótima utilizada. A abordagem considera uma política de atribuição arbitrária. Posteriormente, é adotado o diagrama de processo de decisão de Markov (MDP) [89] (gerado a partir do CTMC) com a equação ótima de *Bellman* para resolver os valores de estado da política atual. Outras políticas são geradas pelos algoritmos baseados em busca gulosa e avaliadas na abordagem de otimização para maximizar a disponibilidade do sistema. Os autores focam na otimização da disponibilidade da infraestrutura de TI, mas não levam em consideração o custo e o impacto ambiental que a política de máquina secundária acarreta. Diferentemente, este trabalho otimiza a disponibilidade focando também na redução do impacto ambiental e do custo.

Para otimizar o consumo de energia, o desempenho do sistema e a confiabilidade, Luo et. al [90] sugerem um algoritmo de agendamento de recursos. Esse algoritmo se fundamenta numa abordagem de modelagem correlacionada empregando modelos de *Semi-Markov*, transformação de *Laplace-Stieltjes* e uma abordagem bayesiana para analisar as relações entre confiabilidade e desempenho, e as relações entre confiabilidade e consumo de energia. Os autores focam na otimização da confiabilidade, consumo e custo, aliados ao desempenho da infraestrutura de TI. Não foi o foco do trabalho a otimização da infraestrutura elétrica.

Chou et. al [91] propõem um mecanismo dinâmico de alocação de recursos de economia de energia (DPRA) com base em um algoritmo de otimização de enxame de partículas para

otimizar a eficiência energética. O mecanismo DPRA considera o consumo de energia da máquina física (PM) e da máquina virtual (VM) abordando também eficiência energética do sistema de ar condicionado no *data center*. Aliado a otimização, o método de regressão de mínimos quadrados é empregado para predizer o uso de recursos de PM para alocar VM e erradicar migrações de VM. Não foi o foco dos autores otimizar a a disponibilidade, impacto ambiental e custo.

Ghamkhari et. al [92] propõem uma estrutura de otimização de portfólio de energia que aprecie um conjunto de opções de energia para *data centers*. A análise sugerida se resume à resolução de programas estocásticos de inteiros mistos lineares. Através de um mercado experimental de eletricidade e dados da carga de trabalho de Internet, são apresentadas várias observações numéricas quanto ao lucro a curto e longo prazo e gerenciamento de risco em diferentes espaços temporais de cada opção energética. Outros parâmetros também são considerados para a otimização, tais como, nível de serviço dos *data centers* e sua capacidade de explorar certas opções de energia e armazenamento. Ghamkhari et. al não sugerem tratamento da disponibilidade do *data center* em seu conjunto de opções de energia, focando apenas na otimização do consumo de energia e no lucro (redução do custo).

Bosse et. al [93] propõem a otimização de um projeto de serviço de TI tolerante a falhas para minimizar os custos e as emissões de gases do efeito estufa (GEE) sujeitos a um certo nível de disponibilidade em um *data center*. Para otimizar, um problema de alocação de redundância (RAP) multi-objetivo é modelado a partir do projeto, e otimizado através de algoritmos genéticos. Dados de um estudo de caso real são considerados para conduzir experimentos demonstrando que as emissões de GEE podem ser reduzidas consideravelmente. Os autores, porém, não otimizam a disponibilidade, apenas atribuem um certo grau de disponibilidade para considerar que o projeto a ser otimizado é tolerante a falhas.

Anan et. al [94] apresentam uma abordagem através de posicionamento dinâmico de máquinas virtuais para otimizar o custo e o consumo de energia de serviços de computação em nuvem. Essa abordagem considera os acordos de nível de serviço definidos pelo cliente para otimizar a alocação de máquinas virtuais. Com isso, é possível fornecer os recursos necessários para que o cliente receba um serviço aceitável com o número mínimo de servidores ativos. Entretanto, os autores, não consideram a tolerância a falhas do sistema em sua

otimização.

Ziafat e Babamir [95] apresentam uma abordagem utilizando o algoritmo multi-objetivo NSGA-II e o agrupamento *K-means* para minimizar o custo e maximizar a disponibilidade na escolha de um *data center*. A abordagem proposta foi empregada em alguns *data centers* distribuídos geograficamente. Os resultados foram comparados com otimização através de busca gulosa e de algoritmos comuns aleatórios para mostrar que a otimização sugerida supera as outras duas heurísticas. Os autores não propõem a sustentabilidade como parâmetro para a otimização da escolha do *data center*.

A Tabela 3.1 representa um sumário que compara os principais assuntos focados neste trabalho com os trabalhos mencionados neste capítulo.

Trabalhos	Otimização	Custo	Sustentabilidade	Disponibilidade
[22]	x	x	x	x
[39]	x	x	x	
[81]	x	x	x	
[82]	x			
[83]	x	x	x	
[84]	x	x	x	
[85]	x	x		
[86]	x	x	x	
[87]	x		x	x
[90]	x	x	x	x
[91]	x	x	x	
[92]	x	x	x	
[93]	x	x	x	x
[94]	x	x	x	
[95]	x	x		x
Este trabalho	x	x	x	x

Tabela 3.1: Sumário dos trabalhos relacionados.

Pode-se concluir que poucos trabalhos fazem esse estudo integrado de custo, disponi-

bilidade e sustentabilidade. Como observado em [22], [90] e [93], estes trabalhos utilizam técnicas de otimização para esta integração dos parâmetros estudados. Este trabalho se difere porque propõe um algoritmo genético como heurística de otimização destes parâmetros.

Capítulo 4

Metodologia

O objetivo deste capítulo é apresentar as etapas adotadas para se realizar a otimização das arquiteturas de *data center* com o algoritmo genético poposto. Vale ressaltar que o *Algoritmo Genético* se comunica com a ferramenta Mercury [4] (e os modelos feitos nesta ferramenta) para poder realizar a otimização dos resultados. Mercury é uma ferramenta utilizada para modelar sistemas através de SPN, CTMC, EFM e RBD [96].

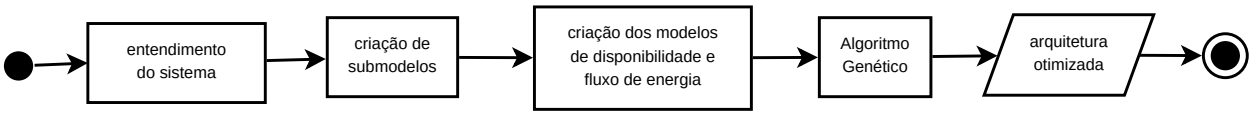

Figura 4.1: Metodologia

A Figura 4.1 ilustra as etapas da metodologia adotada para a otimização integrada do custo, disponibilidade e sustentabilidade das infraestruturas elétricas de data center. A primeira etapa consiste no entendimento do sistema para a criação de um submodelo, a partir da arquitetura de *data center* que se deseja otimizar, que servirá como base para auxiliar na criação dos modelos RBD, EFM e SPN, e na codificação do indivíduo (modelagem do cromossomo).

Os submodelos (segunda etapa) são uma abstração mais simples das arquiteturas para facilitar o entendimento do fluxo e as disposições em série e paralelo dos componentes,

facilitando assim a criação dos modelos de disponibilidade e fluxo como ilustra o exemplo da Figura 4.2.

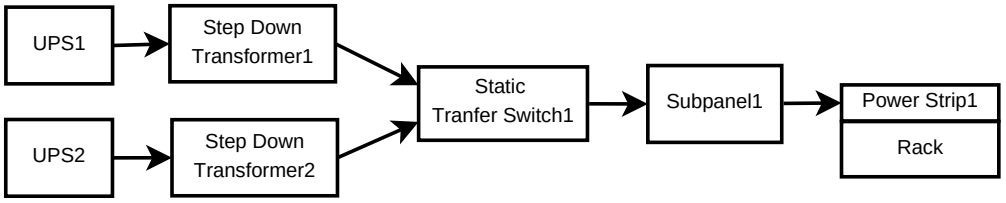

Figura 4.2: Exemplo de submodelo baseado em uma arquitetura elétrica de *data center*.

Na terceira etapa, a partir dos submodelos das arquiteturas, são criados, no Mercury, modelos em SPN ou RBD (ver Figura 4.3), para obter a disponibilidade, e EFM (ver Figura 4.4) para obter a exergia operacional e o custo.

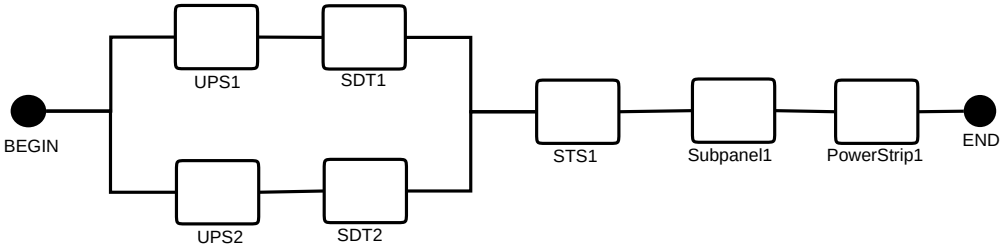

Figura 4.3: Modelo RBD baseado no submodelo apresentado na Figura 4.2.

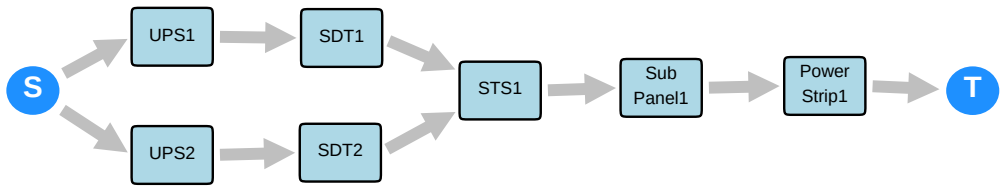

Figura 4.4: Modelo EFM baseado no submodelo apresentado na Figura 4.2.

Os modelos em SPN são utilizados para obter a disponibilidade quando a arquitetura possui redundância com dependência entre seus componentes redundantes (ver Figura 4.5). A Figura 4.5-a apresenta um submodelo criado para representar uma arquitetura elétrica básica. Essa arquitetura possui um UPS e um gerador (GEN), na qual o UPS está ligado e o gerador permanece desligado, sendo acionado em caso de falha do UPS. Ou seja, o gerador depende da falha do UPS para ser acionado. Essa dependência não é possível de ser modelada

com RBD, e por isso é utilizado o SPN, que, no exemplo, é representado pelo modelo *cold standby* como ilustra a Figura 4.5-b. A disponibilidade obtida através desse modelo SPN (P(#UPS_ON = 1) OR (#GEN_ON = 1)) pode ser utilizada para representar um bloco do modelo RBD (ver Figura 4.5-c)e, assim, se computar a disponibilidade da arquitetura.

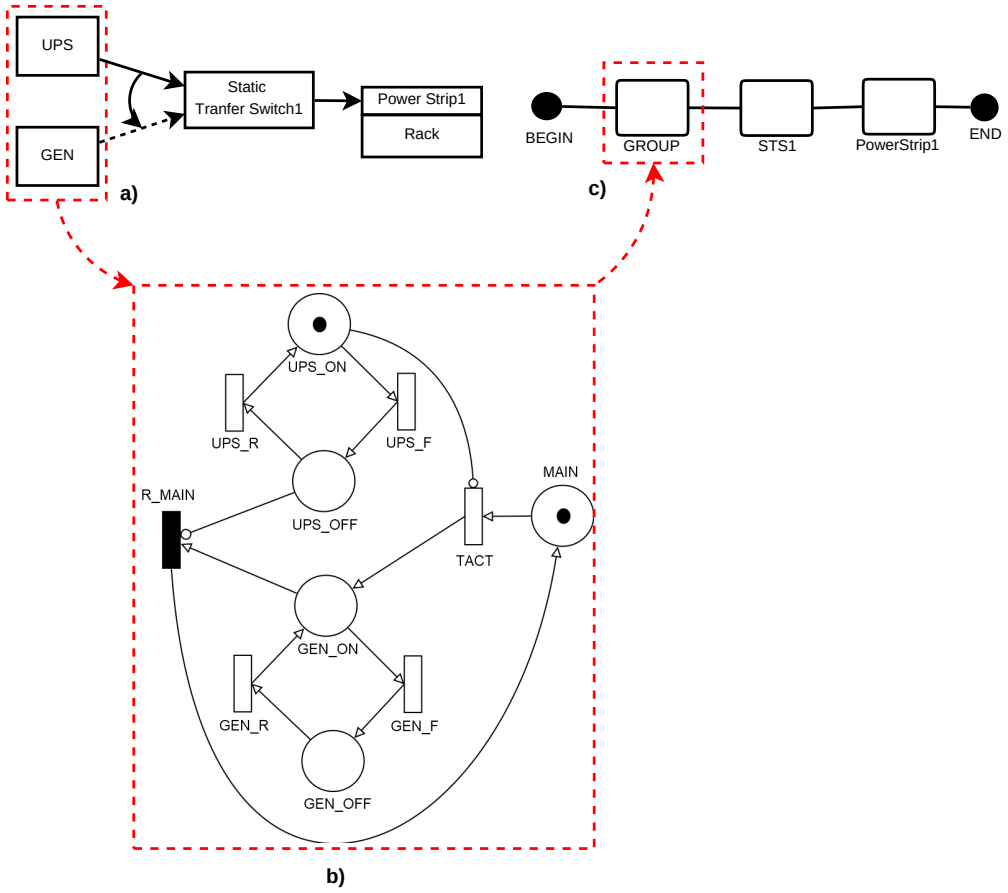

Figura 4.5: Uso do modelo SPN e RBD para se obter a disponibilidade representando redundância de componentes.

É necessário o cálculo prévio da disponibilidade, através dos modelos em RBD e SPN, para inseri-la no modelo em EFM afim de obter a exergia e o custo. Esse processo de integração entre os três modelos é possível através da "Linguagem de *Script* do Mercury" [97]. O exemplo da Figura 4.6 ilustra como funciona a integração dos modelos através do *script* do Mercury. Na Figura 4.6-a, a disponibilidade é calculada no modelo em SPN e em seguida passada como atributo do bloco UPS do modelo em RBD na Figura 4.6-b. O modelo em RBD calcula, juntamente com os MTTFs e MTTRs dos outros blocos, a disponibilidade

do modelo. Essa disponibilidade é passada como parâmetro para o modelo EFM ilustrado na Figura 4.6-c que de posse dos valores de custo da energia elétrica e da eficiência energética dos equipamentos, avalia o custo e a exergia.

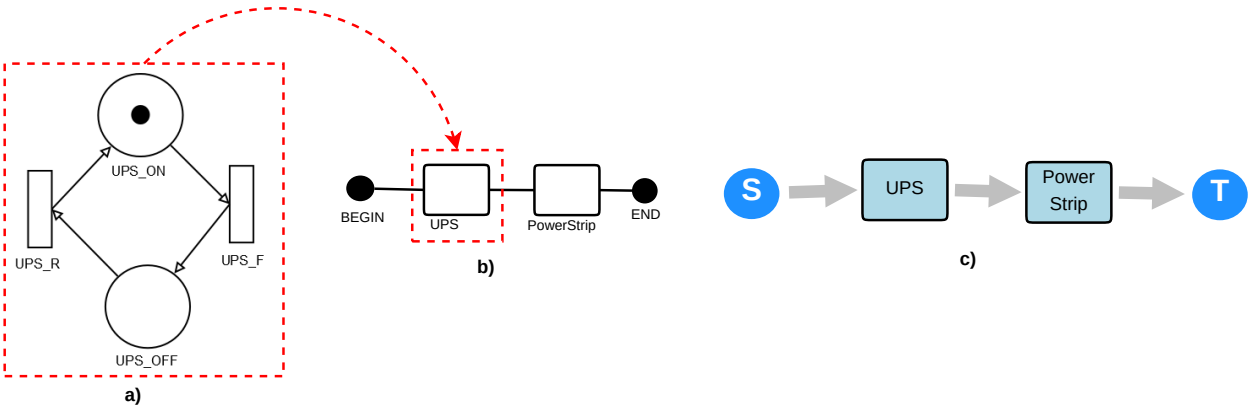

Figura 4.6: Exemplo de integração entre os modelos em SPN, RBD e EFM.

Para representar o exemplo da Figura 4.6 na linguagem de *script* do Mercury, é necessário, primeiro declarar e montar os modelos a partir da sintaxe do *script*. A Figura 4.7 mostra a representação do modelo em SPN da Figura 4.6-a.

```
1  SPN SPNModel{
2
3       place UPS_OFF;
4       place UPS_ON( tokens= 1 );
5
6       timedTransition UPS_F(
7            inputs = [UPS_ON],
8            outputs = [UPS_OFF],
9            delay = tempoFalha
10      );
11      timedTransition UPS_R(
12           inputs = [UPS_OFF],
13           outputs = [UPS_ON],
14           delay = tempoReparo
15      );
16      metric A1 = stationaryAnalysis( expression = "P{#UPS_ON=1}" );
17 }
```

Figura 4.7: Representação da Figura 4.6-a na linguagem de *script* do Mercury.

A primeira linha declara o tipo de modelo (que no caso é SPN) e o nome do mesmo.

Nas linhas 3 e 4, são declarados e denominados os lugares e a quantidade de *tokens*. Da linha 6 até a linha 15, são definidas as transições e seus lugares de entrada (*inputs*), de saída (*outputs*), e o *delay* que para a transição de falha (UPS_F) representa o MTTF, e para transição de reparo (UPS_R) representa o MTTR. A linha 16 denomina a disponibilidade como a métrica A1, e define o tipo de análise e a expressão de cálculo da disponibilidade.

A Figura 4.8 representa o modelo em RBD referente a Figura 4.6-b na linguagem de *script* do Mercury.

```
19 RBD RBDModel{
20
21    hierarchy UPS( availability = solve (SPNModel, A1));
22    block PowerStrip( MTTF = mttf1, MTTR = mttr1);
23    series s0(UPS, PowerStrip );
24
25    top s0;
26
27    metric av = availability;
28    metric rel = reliability( time = 8760 );
29    metric mttf = mttf;
30    metric mttr = mttr;
31 }
```

Figura 4.8: Representação da Figura 4.6-b na linguagem de *script* do Mercury.

Pode-se observar que na linha 19 é definido o tipo e o nome do modelo. Na linha 21, o primeiro bloco (UPS) é definido como *hierarchy* pelo fato de receber, como parâmetro, a disponibilidade, que, no caso, é previamente calculada pelo modelo em SPN descrito no *script* (ver Figura 4.7) através do método *solve*. O bloco *PowerStrip* é definido na linha 22 e recebe como atributos o MTTF e o MTTR. O método *series* (linha 23) determina que os blocos, descritos nele, estão em série. O método *top*, na linha 25, define a série principal do modelo. Da linha 27 à 30, são definidas as métricas que poderão ser calculadas no modelo. A métrica *av*, por exemplo, calcula a disponibilidade do modelo.

Obtida a disponibilidade através dos modelos SPN e RBD, a mesma é passada como parâmetro para o modelo EFM ilustrado na Figura 4.6-c através do mesmo *script* do Mercury como mostra a Figura 4.9.

```
33 EFM EFMModel{
34      component SourcePoint1(
35           type = "SourcePoint",
36           parameters = (
37                efficiency = 100.0,
38                retailPrice = 0.0
39           )
40      );
41      component UPS_5kVA1(
42           type = "UPS_5kVA",
43           parameters = (
44                maxPower = 5.0,
45                efficiency = e1,
46                retailPrice = r1,
47                embeddedEnergy = 3.1392
48           )
49      );
50      component PowerStrip1(
51           type = "PowerStrip",
52           parameters = (
53                maxPower = 5.0,
54                efficiency = e2,
55                retailPrice = r2,
56                embeddedEnergy = 0.35568
57           )
58      );
59      component TargetPoint1(
60           type = "TargetPoint",
61           parameters = (
62                efficiency = 100.0,
63                retailPrice = 0.0
64           )
65      );
66      arc SourcePoint1 -> UPS_5kVA1;
67      arc UPS_5kVA1 -> PowerStrip1;
68      arc PowerStrip1 -> TargetPoint1;
69
70      metric ic = initialCost( eletricityCost = 0.11);
71      metric oc = operationalCost( eletricityCost = 0,11, availability = solve (RBDModel, av), time = 8760 );
72      metric ee = embeddedExergy;
73      metric oe = operationalExergy( time = 8760, availability = solve (RBDModel, av));
74      metric tc(ic + oc);
75      metric te(ee + oe);
76 }
```

Figura 4.9: Representação da Figura 4.6-c na linguagem de *script* do Mercury.

A definição do modelo EFM inicia na linha 33. Da linha 34 à 40 e da linha 59 à 64 são definidos o *SourcePoint* e o *TargetPoint*, respectivamente. Da linha 41 à 58 são definidos os componentes, os tipos (*type*) de cada e os atributos (*parameters*) que já possuem um valor por padrão no Mercury. Neste trabalho, os parâmetros que serão utilizados no modelo EFM são a eficiência (*efficiency*) e o preço de aquisição (*retailPrice*), e que no exemplo são atribuídos pelas variáveis $e1$, $e2$, $r1$ e $r2$. Da linha 66 à 68, são definidos os arcos que ligam os componentes. As métricas são estabelecidas da linha 70 à 75. Pode-se observar que na

linha 71 é utilizada a disponibilidade antes calculada no modelo RBD como parâmetro para obter o custo operacional. A disponibilidade também é utilizada para o cálculo da exergia operacional na linha 73.

A linguagem de *scrpit* do Mercury possui, também, uma função principal (*main*) que chama os demais modelos e métodos declarados e imprimir os resultados obtidos (ver Figura 4.10).

```
78  main {
79
80      A1 = solve(SPNModel, A1);
81      println(A1);
82
83      av = solve(RBDModel, av);
84      rel = solve(RBDModel, rel);
85      mttf = solve(RBDModel, mttf);
86      mttr = solve(RBDModel, mttr);
87
88      println("Availability: " .. av );
89      println("Reliability: " .. rel );
90      println("Mean time to failure: " .. mttf );
91      println("Mean time to repair: " .. mttr );
92
93      ic = solve(EFMModel, ic);
94      println("Acquisition Cost: " .. ic);
95      oc = solve(EFMModel, oc);
96      println("Operational Cost: " .. oc);
97      tc = solve(EFMModel, tc);
98      println("Total Cost: " .. tc);
99
100     println("");
101
102     oe = solve(EFMModel, oe);
103     println("Operational Exergy: " .. oe);
104
105 }
```

Figura 4.10: Função principal do *script*.

Na quarta etapa, o algoritmo genético é aplicado para otimizar as arquiteturas a partir dos modelos RBD, SPN e EFM. Ao fim todo processo, espera-se obter um conjunto de soluções satisfatórias a partir da arquitetura otimizada. Mais detalhes sobre a quarta etapa são fornecidos na Seção 4.1.

4.1 Algoritmo Genético

A quarta etapa consiste na aplicação do algoritmo genético. O objetivo é utilizar o AG integrado com Mercury para avaliar os submodelos das arquiteturas e obter os parâmetros desejados para a otimização. Essa integração entre o algoritmo genético e o Mercury é possível através da linguagem de *script* do Mercury como representado na Figura 4.11.

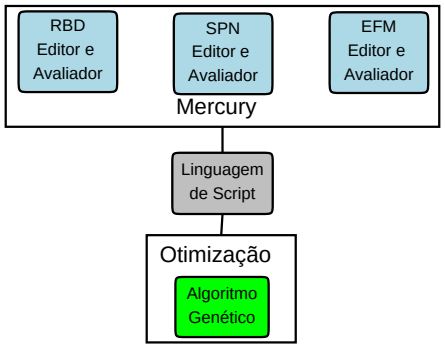

Figura 4.11: Integração entre o Mercury e o algoritmo genético.

No fluxograma ilustrado na Figura 4.12, pode-se observar as etapas do AG proposto, sendo a integração com o Mercury efetivada nas etapas "Geração da população inicial", "Cruzamento" e "Mutação", nas quais os cromossomos são gerados e alterados. Tanto a integração com Mercury como as demais etapas são melhor detalhadas ao decorrer desta seção.

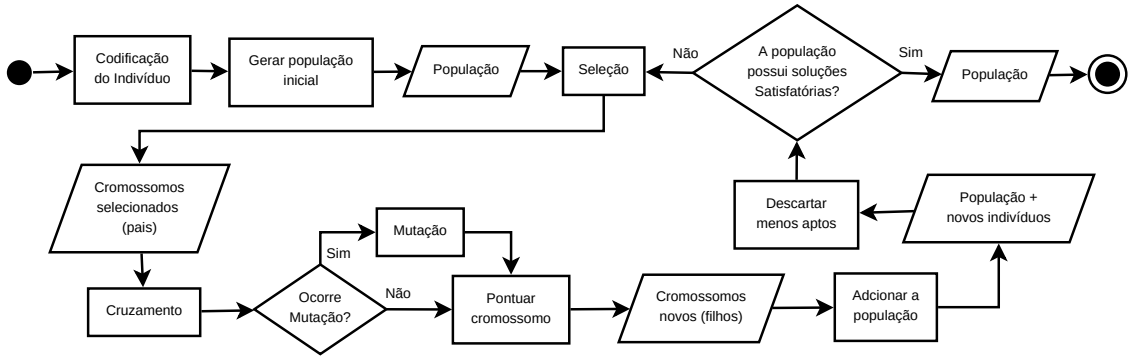

Figura 4.12: Fluxograma do algoritmo genético proposto.

Cada etapa da Figura 4.12 é definida pelo Algoritmo 1 que apresenta o algoritmo genético proposto neste trabalho. Cada parte deste algoritmo será detalhada no decorre desta seção.

Algorithm 1 Algoritmo Genético

```
1: populacao[ ] ← gerarPopulacao(listaEquipamentos)      ▷ Geração da população inicial
2: para i ← 0 até qGen faça                              ▷ qGen = quantidade de gerações
3:     selecionados[ ] ← selecaoTorneio(populacao)
4:     filhos[ ] ← cruzamento(selecionados)
5:     para j ← 0 até tamanho(filhos[ ]) faça
6:         filhos[j] ← mutacao(filhos[j])
7:                                                    ▷ a mutação pode ou não acontecer
8:         filhos[j] ← pontuarCromossomo(filhos[j])
9:     fim para
10:    populacao[ ] ← populacao[ ] + filhos[ ]
11:    populacao[ ] ← descartarPiores(populacao[ ])
12: fim para
13: retorna populacao
```

No algoritmo genético proposto, o indivíduo é codificado com um único cromossomo, sendo assim, pode-se considerar que o indivíduo é o cromossomo. O cromossomo é estruturado de acordo com o submodelo de maneira a agrupar todos seus componentes em um vetor único de genes. Cada gene é alocado em uma posição do vetor, independentemente de estarem em série ou paralelo, conforme pode ser visto na Figura 4.13. Desta forma, torna-se mais viável o cruzamento dos cromossomos. Como no exemplo da Figura 4.13, as Séries 1 e 2 são agrupadas no vetor de genes, uma após a outra, e em seguida, concatenadas com a Série 3 para formar o cromossomo.

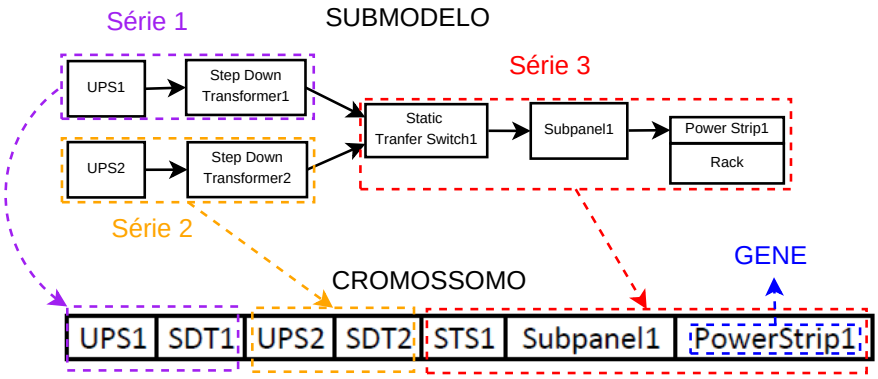

Figura 4.13: Codificação do cromossomo.

Após a codificação do cromossomo, inicia-se a geração da população inicial na qual cada cromossomo possui uma solução aleatória. Para esta fase do algoritmo genético, é utilizado um arquivo de entrada contendo uma lista de equipamentos correspondentes aos componentes presentes na arquitetura para a geração dos indivíduos. A geração da população inicial é representada pelo fluxograma da Figura 4.14.

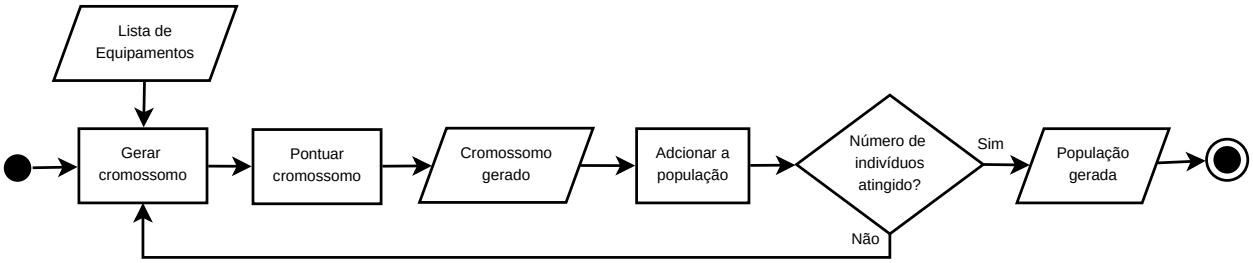

Figura 4.14: Geração da população inicial.

O Algoritmo 2 apresenta o passo-a-passo da geração da população inicial. Pode-se observar, na linha 2, que o tamanho da população (quantidade de cromossomos), é definido de acordo com o estudo de caso a ser aplicado. O retorno desta função é um vetor de cromossomos, cada um contendo uma solução aleatória.

Algorithm 2 Geração da população inicial

```
1: função GERARPOPULACAO(listaEquipamentos)
2:     para i ← 0 até tamPop faça          ▷ tamPop = tamanho desejado da população
3:         cromossomo ← gerarCromossomo(listaEquipamentos)
4:         cromossomo ← pontuarCromossomo(cromossomo)
5:         populacao[ ].adicionar(cromossomo)
6:     fim para
7:     retorna populacao[ ]
8: fim função
```

O cromossomo é um objeto formado, basicamente, por um vetor de genes e a pontuação que representa o valor de sua aptidão. Para gerar a população inicial é necessário gerar cada cromossomo, o que implica em sortear os equipamentos para preencher cada gene e,

em seguida, pontuar o cromossomo a partir da avaliação dos modelos de disponibilidade e fluxo no Mercury. O Algoritmo 3 apresenta as etapas da geração do cromossomo. A entrada da função é um vetor com equipamentos gerado a partir de um arquivo. O retorno é um cromossomo ainda não pontuado e com o vetor de genes preenchido com o componente da arquitetura correspondente a cada posição do vetor.

Algorithm 3 Geração do cromossomo

```
1: função GERARCROMOSSOMO(listaEquipamentos[ ])
2:     para i ← 0 até tamanho(cromossomo.genes[ ]) faça
3:         cromossomo.genes[i] ← sortearEquipamento(listaEquipamentos[ ])
4:     fim para
5:     retorna cromossomo
6: fim função
```

A Figura 4.15 ilustra como cada equipamento é sorteado e alocado em um gene. Para cada equipamento presente no submodelo, esse arquivo de entrada fornece equipamentos com diferentes MTTFs, custo de aquisição e eficiência energética. A partir da leitura deste arquivo, o algoritmo genético instancia cada cromossomo da população de acordo com submodelo da arquitetura proposta (ver Figura 4.13). Em cada posição do vetor cromossômico, é alocado aleatoriamente, a partir da lista de equipamentos, um componente do mesmo tipo descrito no gene.

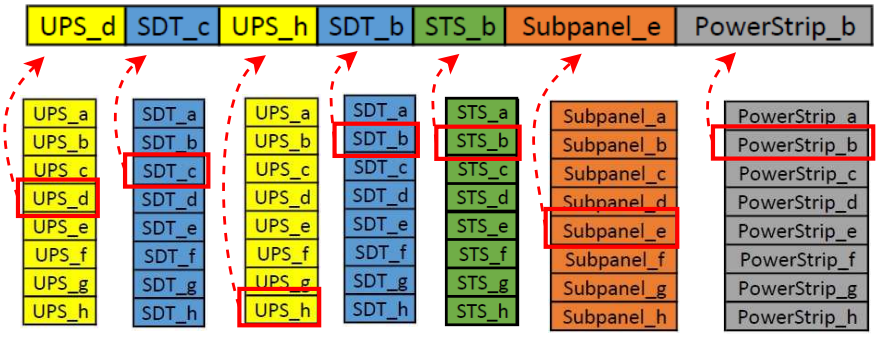

Figura 4.15: Geração de cromossomo aleatório.

Assim que gerado o cromossomo, ele precisa ser pontuado mensurar sua aptidão. Os equipamentos presentes no cromossomo são atribuídos aos modelos, criados na etapa 3 da

metodologia, RBD, SPN e EFM no Mercury. Em seguida, os modelos RBD e SPN são avaliados para se obter a disponibilidade da arquitetura em análise. Esse resultado é utilizado como parâmetro de entrada no modelo EFM, empregado para se quantificar a exergia operacional e o custo.

De posse destas métricas: disponibilidade, exergia operacional e custo, estas são utilizadas para o cálculo da *Função Fitness* multiobjetiva afim de pontuar o cromossomo com um valor que possa determinar a qualidade da solução obtida. Esse processo de pontuação pode ser observado no fluxograma ilustrado na Figura 4.16.

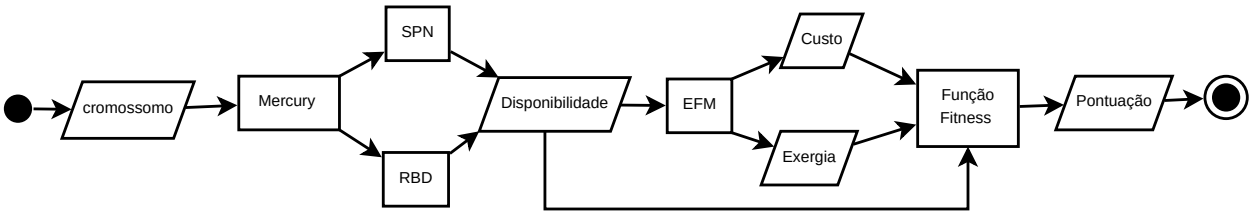

Figura 4.16: Pontuação.

A função *pontuarCromossomo* é descrita pelo Algoritmo 4. Na linha 2, a integração do algoritmo genético com o Mercury fica evidenciada pela função *scriptMercury*. Este método utiliza, como parâmetro de entrada, um arquivo contendo um *script* do Mercury que descreve os modelos em SPN, RBD e EFM (etapa 3, da metodologia), já integrados, da arquitetura a ser otimizada. Essa função transmite ao *script* os atributos de cada componente da arquitetura presente no cromossomo. Pode-se observar este processo nas linhas 5 à 8 com o MTTF, MTTR, curto de aquisição e a eficiência sendo passados para o *script*. Em seguida, o *script* calcula, a partir da avaliação dos modelos em SPN, RBD e EFM no Mercury, disponibilidade, exergia e custo (linhas 10 à 13). Por fim, esse, parâmetros são passados para a função *Fitness* (linha 14) que retorna uma pontuação (valor de aptidão) para o cromossomo.

Para cada estudo de caso, uma função *Fitness* deve ser criada devido as diferenças nos graus de complexidade de cada caso. A função *Fitness* multiobjetiva equilibra as três métricas de maneira que, quanto maior a disponibilidade, menor a exergia e menor o custo, maior será a pontuação atribuída para a solução contida no cromossomo.

Algorithm 4 Pontuar cromossomo

```
1: função pontuarCromossomo(cromossomo)
2:     script = scriptMercury("script.mry")          ▷ Integração do AG com o Mercury
3:                                  ▷ script.mry é o arquivo onde o script foi escrito
4:     para i ← 0 até tamanho(cromossomo.genes[ ]) faça
5:         script ← script(cromossomo.genes[i].mttf)
6:         script ← script(cromossomo.genes[i].mttr)
7:         script ← script(cromossomo.genes[i].custoAquisicao)
8:         script ← script(cromossomo.genes[i].eficiencia)
9:     fim para
10:    disp ← script.executar(RBD, disponibilidade)
11:                ▷ RBD e disponibilidade: tipo de modelo e métrica respectivamente
12:    exergiaOp ← script.executar(EFM, exergiaOperacional)
13:    custoT ← script.executar(EFM, custoTotal)
14:    cromossomo.pontuacao ← funcaoFitness(disp, esergiaOp, custoT)
15:    retorna cromossomo
16: fim função
```

Após gerados e pontuados todos os cromossomos da população inicial, o próximo passo do algoritmo genético é a seleção. São sorteados quatro indivíduos da população, dos quais os dois melhores (com pontuação maior) são selecionados para realizar o cruzamento. Este método de seleção é conhecido como Torneio (ver Algoritmo 5).

Algorithm 5 Seleção Torneio

```
1: função selecaoTorneio(populacao[ ])
2:     torneio[ ] ← sortear(populacao[ ], 4)          ▷ sorteia 4 cromossomos da população
3:     selecionados[ ] ← selecionarMelhores(torneio[ ], 2)        ▷ seleciona os dois melhores
4:     retorna selecionados[ ]
5: fim função
```

Depois de selecionados os cromossomos (pais), começa o processo de cruzamento descrito no Algoritmo 6. É estabelecido, aleatoriamente, um ponto de corte (linha 4) para, a partir

dele, haver o cruzamento dos genes dos pais para formar novos cromossomos (filhos). Esse ponto de corte que é um valor inteiro que será atribuído ao índice onde o vetor de genes é cortado para a troca. Essa troca de genes é definida nas linhas 6 à 14. O *filho1* recebe os genes da posição 0 até a posição $pontoCorte - 1$ do *pai1* e os genes da posição *pontoCorte* até o fim do vetor de genes do *pai2*. jà o *filho2* recebe os genes da posição 0 até a posição $pontoCorte - 1$ do *pai2* e os genes da posição *pontoCorte* até o fim do vetor de genes do *pai1*. o retorno desta função é um vetor com os dois novos cromossomos gerado no cruzamento.

Algorithm 6 Cruzamento

```
1:  função CRUZAMENTO(selecionados[ ])
2:      pai1 ← selecionados[0]
3:      pai2 ← selecionados[1]
4:      pontoCorte ← sortearInteiro(tamanho(pai1.genes[ ]) - 1)
5:                                  ▷ sorteia um ponto de corte pelo tamanho do cromossomo
6:      para i ← 0 até pontoCorte faça
7:          filho1.genes[ ].adcionar(pai1.genes[i])
8:      fim para
9:      para i ← pontoCorte até tamanho(pai2.genes[ ]) faça
10:         filho1.genes[ ].adcionar(pai2.genes[i])
11:     fim para
12:     para i ← 0 até pontoCorte faça
13:         filho2.genes[ ].adcionar(pai2.genes[i])
14:     fim para
15:     para i ← pontoCorte até tamanho(pai1.genes[ ]) faça
16:         filho2.genes[ ].adicionar(pai1.genes[i])
17:     fim para
18:     filhos[ ].adicionar(filho1, filho2)
19:     retorna filhos[ ]
20: fim função
```

Troca de genes estabelecida pelo cruzamento pode ser ilustrada pelo exemplo ilustrado na Figura 4.17.

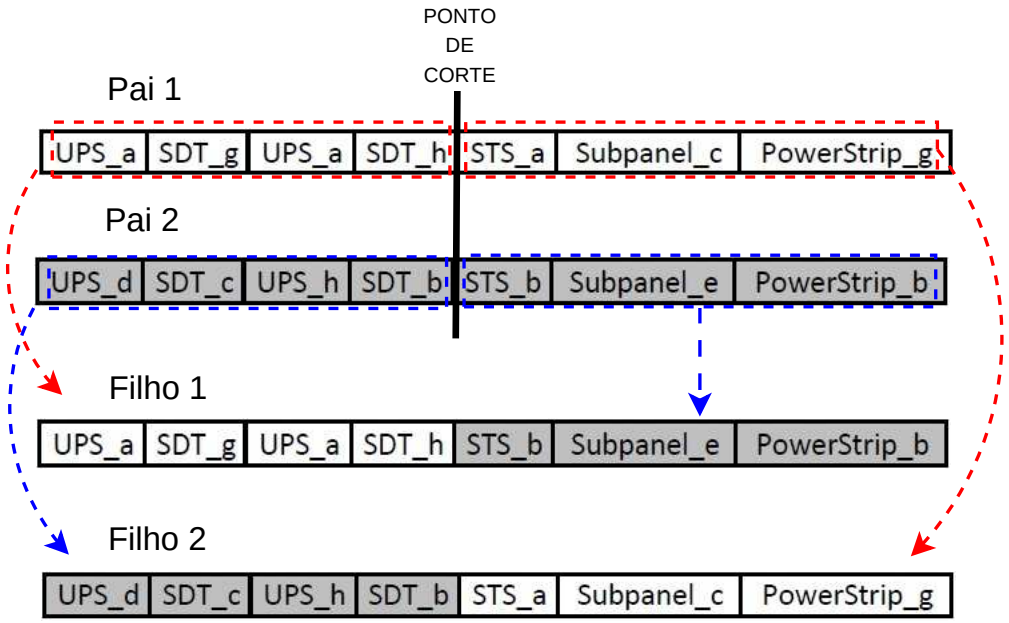

Figura 4.17: Exemplo de cruzamento.

Nestes dois novos cromossomos gerados existe uma pequena probabilidade de haver mutação (ex., 5%), ou seja, de um gene aleatório ser escolhido para ter seu valor trocado. O Algoritmo 7 demonstra que, caso haja a mutação, um gene do cromossomo é sorteado para que o equipamento nele contido seja substituído por outro equipamento escolhido aleatoriamente na lista gerada pelo arquivo de entrada (linhas 4 e 5).

Algorithm 7 Mutação

```
1: função MUTACAO(cromossomo)
2:     probabilidade ← sortearInteiro(100)
3:     se probabilidade < 5 então          ▷ probabilidade de 5% de ocorrer mutação
4:         i ← sortearInteiro(tamanho(cromossomo.genes[ ])-1)
5:         cromossomo.genes[i] ← sortearEquipamento(listaEquipamentos)
6:     fim se
7:     retorna cromossomo
8: fim função
```

A Figura 4.18 expõe um exemplo do processo de mutação descrito no Algoritmo 7.

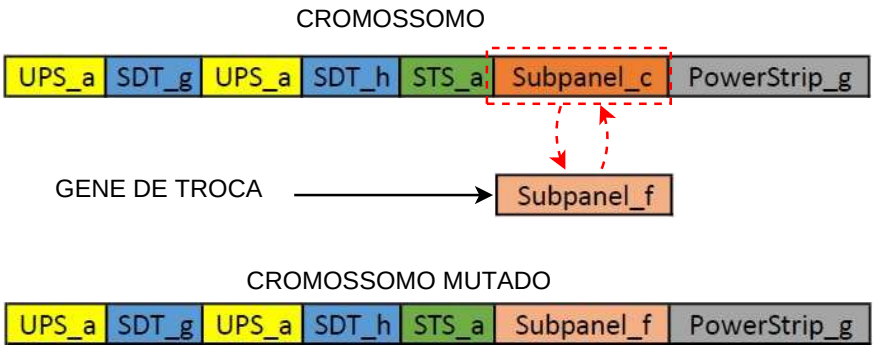

Figura 4.18: Exemplo de mutação.

Após o cruzamento e mutação (caso haja), os dois novo indivíduos passam pelo processo de pontuação descrito pela Figura 4.16 e pelo Algoritmo 4 já explicado nesta seção. Estes novos cromossomos, depois de pontuados, são inseridos à população. Por fim, após esta inserção, os dois piores indivíduos (com menor pontuação) de toda a população são descartados (Elitismo).

O algoritmo genético é repetido a partir da seleção até que se atinja o critério de parada e o algoritmo retorne uma população de soluções satisfatórias. Para finalizar, o algoritmo genético observa se a população de cromossomos apresenta soluções satisfatórias através através do grau de convergência da população. Verifica-se a média da pontuação da população em cada geração. Quando essas médias tendem a ser iguais por algumas gerações é sinal de que não haverá mais melhoras significativas na população, mesmo rodando mais gerações.

Após um determinado número gerações, o algoritmo genético tende a estabilizar os resultados da otimização, retornando uma população homogênea de soluções satisfatórias. Ou seja, a partir deste número, as melhoras na população são insignificantes ou não acontecem mais.

Capítulo 5

Estudo de caso

Este capítulo apresenta dois estudos de caso que tem como objetivo ilustrar a aplicabilidade da metodologia proposta em um cenário real da infraestrutura elétrica de um *data center* típico. O primeiro estudo foca em um cenário mais simples para validar a estratégia proposta. O segundo estudo analisa um cenário real e aplica otimização em um infraestrutura de um *data center* típico.

5.1 Estudo de caso I

Nesta seção, um estudo de caso apresenta a otimização de infraestruturas elétricas básicas de um data center e compara com um algoritmo de força bruta. Este algoritmo analisa todas as possibilidades de equipamentos dada a arquitetura que se deseja otimizar. Dessa forma, o objetivo desse primeiro estudo de caso é comparar o tempo de execução e a eficácia da estratégia de otimização proposta em relação com a solução ótima que é provida pela combinação de todas as possibilidades através do algoritmo de força bruta.

5.1.1 Modelos

Cinco arquiteturas (A1, A2, A3, A4 e A5) de graus de complexidade diferentes foram adotadas com o intuito de mostrar a eficácia do algoritmo genético para otimizar a dispo-

nibilidade, exergia operacional e o custo total em um curto espaço de tempo em relação ao algoritmo de força bruta, independente desta complexidade. A Figura 5.1 mostra os submodelos das cinco arquiteturas elétricas básicas de *data center* que foram adotadas como estudo de caso neste trabalho.

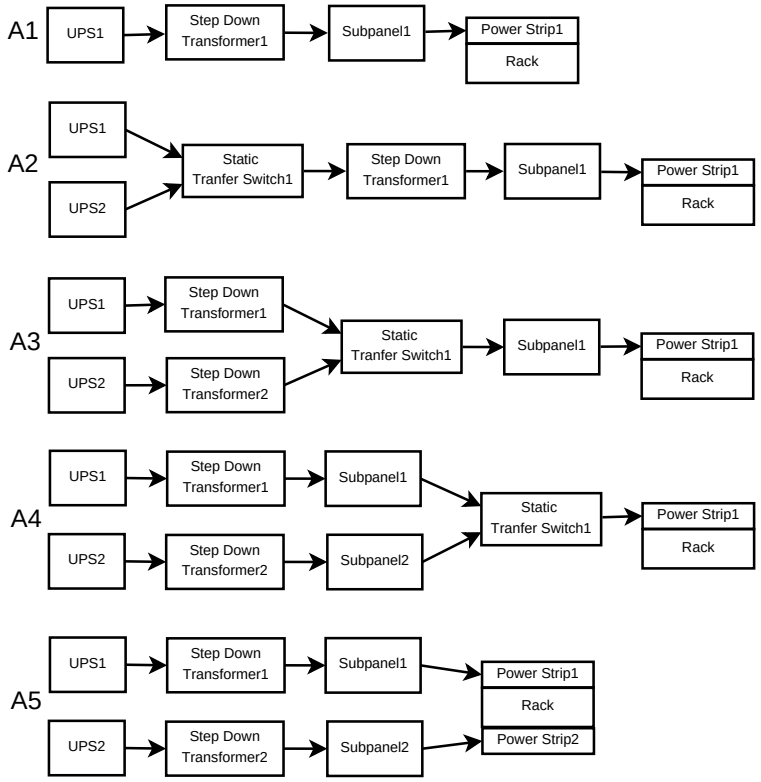

Figura 5.1: Arquiteturas elétricas básicas de *data center*

Para cada arquitetura, são criados seus modelos RBD (ver Figura 5.2) e EFM (ver Figura 5.3) de acordo com os submodelos presentes na Figura 5.1. Com esses modelos, durante a execução do algoritmo genético, cada cromossomo gerado se ajusta ao modelo RBD para obter a disponibilidade. Em seguida, esta métrica, juntamente com os genes do cromossomo, é utilizada para ajustar o modelo EFM e obter o custo total e a exergia operacional. De posse destes resultados, a função *fitness* calcula a pontuação do cromossomo.

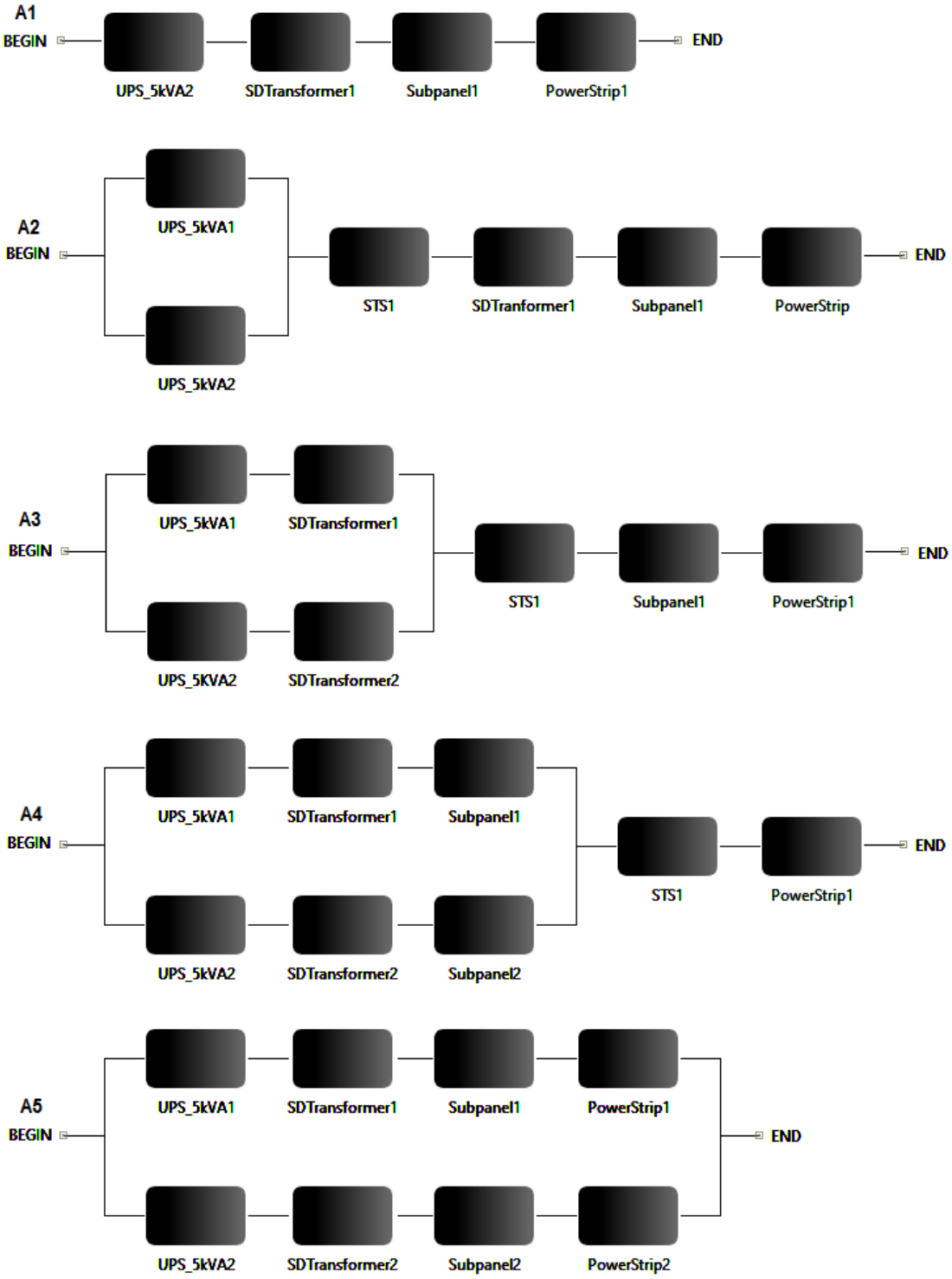

Figura 5.2: Modelos em RBD da arquiteturas básicas propostas

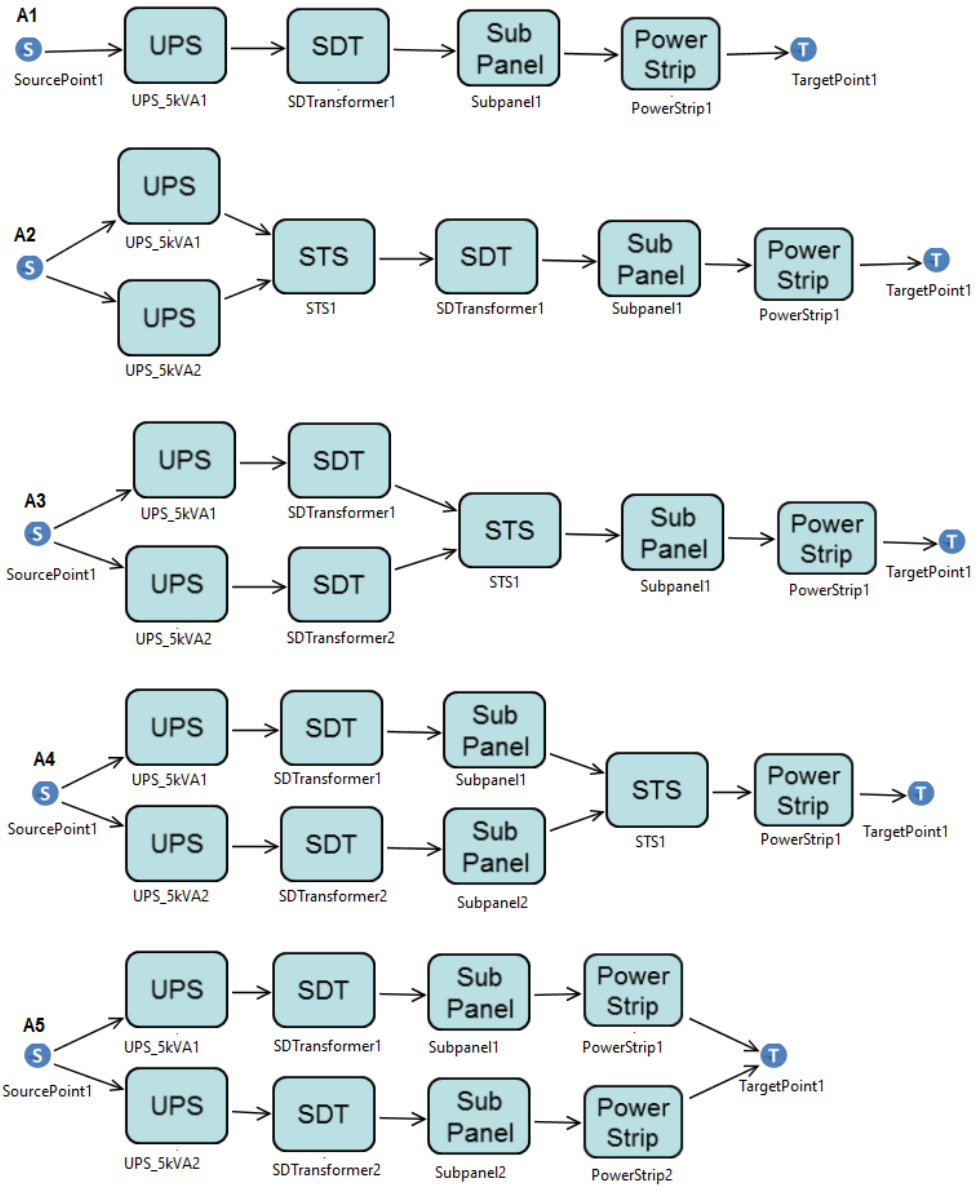

Figura 5.3: Modelos em EFM da arquiteturas básicas proposta

Cada cromossomo é modelado conforme as cinco arquiteturas adotadas, concatenando toda a arquitetura num vetor único como ilustra a Figura 5.4. Vários cromossomos são gerados até atingir uma população inicial de 30 indivíduos. Posteriormente, selecionam-se dois indivíduos para o cruzamento, com 5% de chance de haver mutação, gerando dois novos indivíduos que são inseridos na população. Em seguida toda a população é avaliada afim de se descartar os dois piores cromossomos (de menor pontuação de acordo com a função

fitness). Cada ciclo deste, a partir da população inicial, é uma geração.

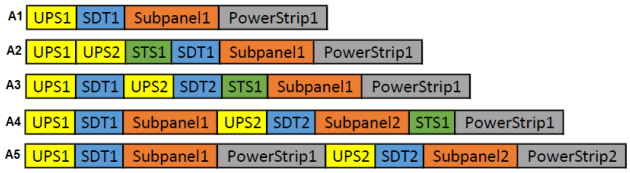

Figura 5.4: Cromossomos referentes a cada uma das arquiteturas básicas adotadas

Foi observado, experimentalmente, que com um número de gerações superior a 1000, a população de cromossomos não evoluía de forma significativa, sendo este o número de gerações utilizado para a otimização das arquiteturas analisadas. Desta forma, após as 1000 gerações, a população de cromossomos atinge o objetivo de trazer um conjunto de soluções satisfatórias, ou seja, otimizadas.

Para este trabalho, a função *fitness* é baseada em uma abordagem agregada [69], utilizando métodos que atribuem pesos a cada objetivo e os unem em uma função única. Essa função foi elaborada de maneira a maximizar a disponibilidade e minimizar a exergia operacional e o custo total lembrando que o aumento da primeira impacta diretamente nas outras duas métricas. Desta forma a função *fitness* pode ser representada de acordo com a Equação 5.1.

$$Pontuacao = (Disp \times 10) - (Custo \div 1000) - Exergia \tag{5.1}$$

A Equação 5.1 determina a qualidade da solução contida no cromossomo. Quanto maior o valor da *Pontuacao*, melhor é o cromossomo. A disponibilidade e o custo total foram ajustados para a mesma ordem de grandeza da exergia operacional afim de manter o equilíbrio entre estas variáveis e obter uma melhor visualização do resultado.

5.1.2 Resultados

A disponibilidade, por ser uma probabilidade, varia de 0 a 1, podendo também ser representada em porcentagem. Como as avaliações dos modelos apresentaram dízimas

muito próximas de 1, foi computada a disponibilidade em função da quantidade de 9s ($-\log(1-disp)$).

A Tabela 5.1 apresenta uma comparação dos resultados obtidos na otimização das 5 arquiteturas utilizando o algoritmo genético proposto com o algoritmo da força bruta. Nessa tabela, AG representa algoritmo genético, FB são os resultados obtidos pela força bruta, e ER foi o erro obtido. Além disso, a tabela também apresenta um comparativo do tempo de execução (Tempo da tabela) demandado para se obter os resultados sob cada abordagem.

Arquitetura A1	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
Algoritmos Genéticos	14447,30	19,84	3,5486	1,02
Força Bruta	14420,84	18,92	3,5278	5,95
Erro (%)	0,1831	4,6511	0,5859	580,18
Arquitetura A2	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
Algoritmos Genéticos	18984,57	30,19	3,5752	1,16
Força Bruta	18965,65	30,77	3,6320	360,64
Erro (%)	0,0996	1,8850	1,5644	31039,42
Arquitetura A3	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
Algoritmos Genéticos	19409,09	29,66	3,6231	1,25
Força Bruta	19302,60	27,22	3,6000	5620,21
Erro (%)	0,5486	8,2088	0,6384	446084,53
Arquitetura A4	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
Algoritmos Genéticos	19613,98	30,49	3,6718	1,35
Força Bruta	19585,69	30,77	3,7394	91730,10
Erro (%)	0,1441	0,9154	1,8071	6763260,63
Arquitetura A5	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
Algoritmos Genéticos	18679,81	20,55	7,0975	1,35
Força Bruta	18725,14	22,39	7,3533	62669,53
Erro (%)	0,2420	8,1739	3,478483	4628701,75

Tabela 5.1: Otimização das Arquiteturas Adotadas.

Na primeira linha da tabela, pode-se observar o custo , a exergia consumida , a dispo-

nibilidade obtida, e o tempo de execução. Na segunda linha, são observados os resultados da otimização através do algoritmo genético para a arquitetura A1. Na terceira linha são visualizados os resultados obtidos por força bruta para a mesma arquitetura. Na quarta linha é observado o erro obtido através da comparação entre os resultados da otimização pelo algoritmo genético e pela força bruta. A partir da quinta linha, segue-se a mesma ordem de resultados obtidos em cada uma das cinco arquiteturas adotadas neste estudo de caso.

É possível verificar que a técnica de otimização obtêm resultados consideráveis em todas as arquiteturas, observando-se um aumento na disponibilidade e uma redução na exergia operacional e no custo. Pode-se verificar que os resultados obtidos através da otimização com o algoritmo genético são muito próximos dos resultados obtidos por força bruta. No entanto, o tempo gasto para se executar cada uma das duas técnicas é bem distinto. A medida que a complexidade das arquiteturas aumenta, o tempo gasto para otimizar com força bruta cresce geometricamente. Já no caso da otimização através do algoritmo genético, o tempo gasto permanece aproximadamente o mesmo e significativamente menor que o tempo gasto pela força bruta. Comparando, na Tabela 5.1, o tempo gasto para otimizar a arquitetura A5 utilizando AG é aproximadamente 45000 vezes menor que o tempo gasto pela força bruta e com resultados muito próximos.

5.2 Estudo de caso II

Este estudo de caso apresenta a otimização de infraestruturas elétricas de *data center* enquadradas na classificação TIER de a acordo com a norma ANSI/TIA-942 [12] através do mesmo algoritmo genético proposto.

5.2.1 Modelos

Quatro arquiteturas elétricas de *data center*, enquadradas na classificação TIER (TIER1, TIER2, TIER3, TIER4), cada uma correspondente a um nível da classificação, foram adotadas para aplicação da otimização com o algoritmo genético proposto. O objetivo é otimizar a disponibilidade, exergia operacional e o custo em um curto espaço de tempo conforme

demonstrado no estudo de caso I. A comparação com o algoritmo de força bruta se torna inviável devido a complexidade das arquiteturas analisadas neste estudo caso em relação ao primeiro.

TIER I

É a arquitetura mais básica, com redundância N, possuindo um caminho único de distribuição (ver figura 5.5). Possui um gerador como alternativa ao fornecimento de energia.

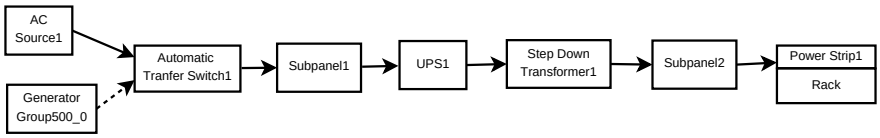

Figura 5.5: Submodelo da Arquitetura TIER I

Para representar a redundância do mecanismo de de ativação do gerador, a Figura 5.6 apresenta um modelo *cold stanby*. Este modelo representa a ativação do gerador mediante a falha da concessionária de energia. A disponibilidade, para este modelo, é obtida através da expressão de probabilidade: P{(#CON_ON = 1) OR (#GEN1_ON = 1)}

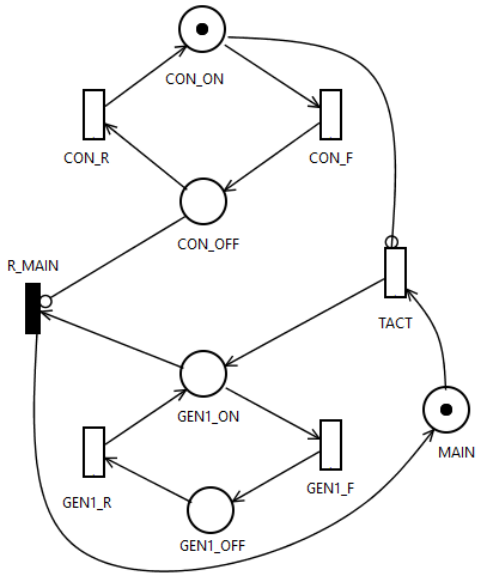

Figura 5.6: Modelo SPN referente à redundância concessionária/gerador da Arquitetura TIER I

Ao calcular a disponibilidade do modelo da Figura 5.6, este valor é utilizado no bloco CON_GEN do modelo RBD ilustrado na Figura 5.7. A avaliação desse modelo fornece a disponiblidade de sistema TIER I.

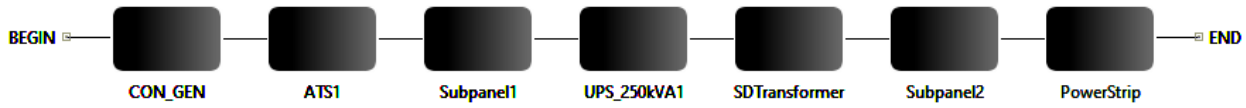

Figura 5.7: Modelo RBD referente à Arquitetura TIER I

Obtida a disponibilidade do modelo RBD anterior, este resultado é utilizado no modelo EFM (ver Figura 5.8) para o cálculo da exergia operacional e do custo. Estas métricas juntamente com a disponibilidade são utilizadas pelo algoritmo genético proposto para o cálculo da função *fitness* gerando a pontuação dos cromossomos para que otimização seja realizada. Os pesos das arestas representam a proporção do fluxo de energia que, no exemplo, é muito maior na concessionária por ser o fornecimento principal de energia, enquanto que o gerador só é acionado na falha deste fornecimento.

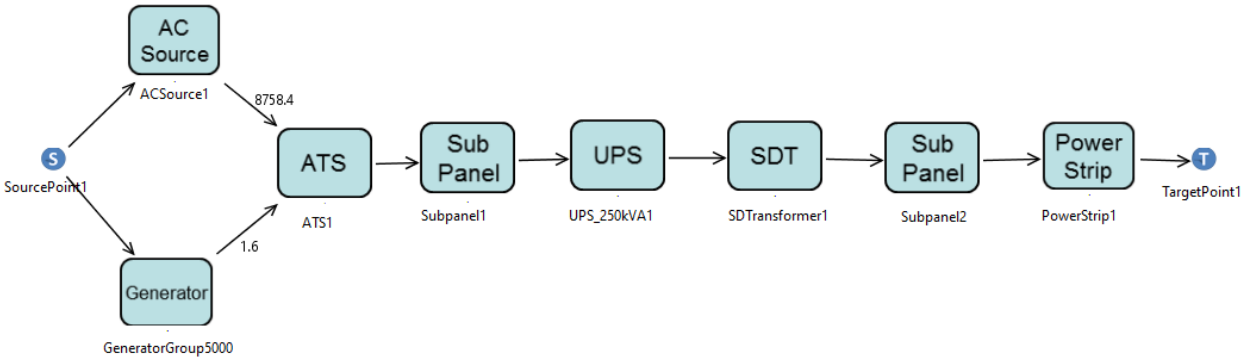

Figura 5.8: Modelo EFM referente à Arquitetura TIER I

TIER II

A Figura 5.9 representa o submodelo da arquitetura TIER II, segundo nível de complexidade da classificação TIER, possuindo uma redundância $N + 1$ nos componentes de

fornecimento elétrico, e com um caminho único de distribuição.

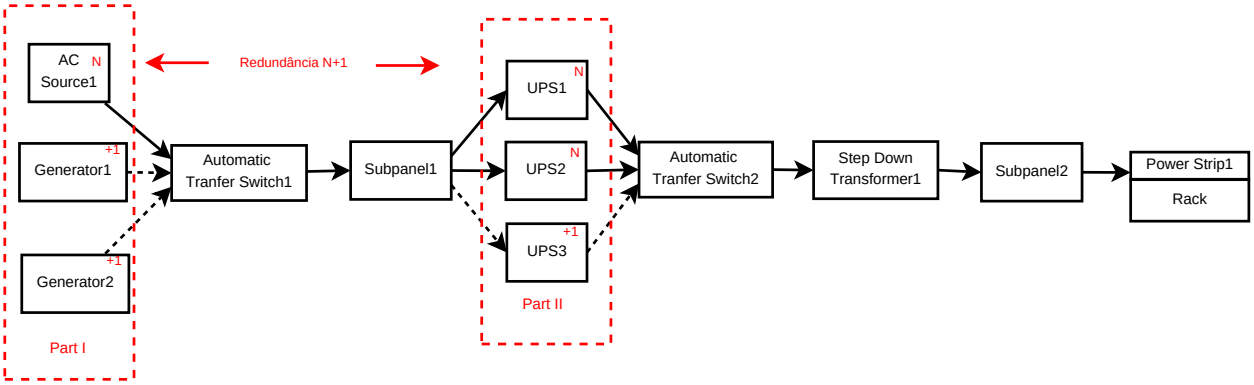

Figura 5.9: Submodelo da Arquitetura TIER II

Pra modelar as redundâncias desta arquitetura, dois modelos SPN foram criados para avaliar a disponibilidade em cada redundância. Em ambos modelos, se o *token* estiver em lugares com denominação terminada em "_ON", significa que o componente está operante. Caso contrário, ou seja, estando em lugares com denominação terminada em "_OFF", significa que o componente está inoperante. Caso as transições com denominação terminada em "_F" forem disparadas, isso representa a falha do equipamento. Entretanto se as transições com denominação terminada em "_R" forem disparadas, isso significa que o equipamento foi reabilitado.

Para representar a redundância concessionária/gerador1/gerador2 (*Part I* da Figura 5.9), a Figura 5.10 apresenta um modelo SPN de maior complexidade baseado no modelo *cold stanby*. Este modelo ilustra a disposição da concessionaria de energia, com a alternativa de dois geradores caso haja interrupção no fornecimento de energia elétrica. Caso a concessionária falhe, o gerador1 é acionado. Caso este falhe, o gerador2 é acionado. Esse subsistema (*Part I*) é considerado em funcionamento se ao menos a concessionária ou um dos geradores estiver em funcionamento. Para o calculo da disponibilidade deste modelo, a seguinte expressão de probabilidade é utilizada: P{(#CON_ON = 1) OR (#GEN1_ON = 1) OR (#GEN2_ON = 1)}.

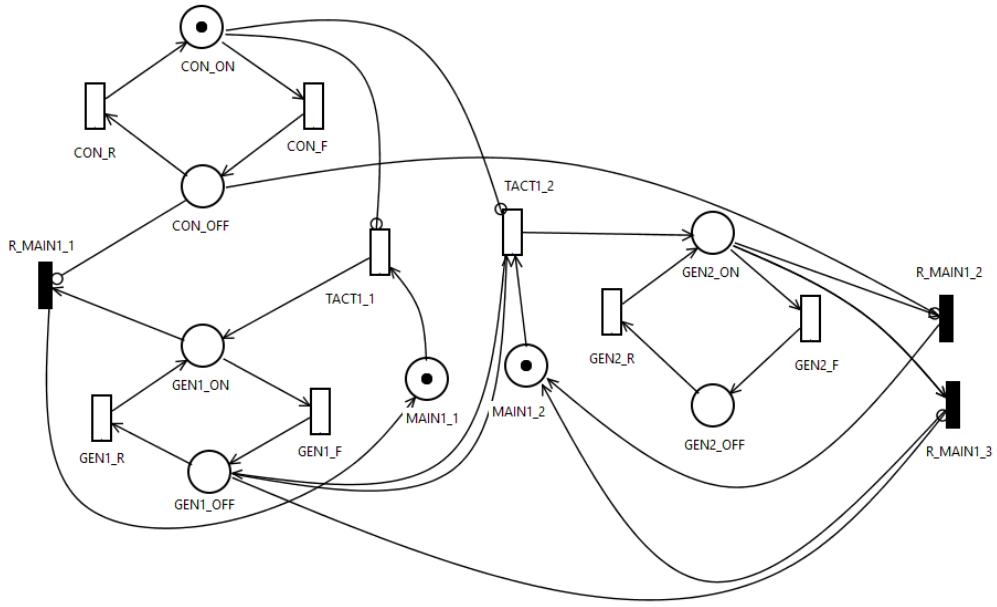

Figura 5.10: Modelo SPN referente à redundância concessionária/gerador1/gerador2

O lugar CON_ON inicia com um *token*, e a transição CON_F fica habilitada definindo que a concessionária está operante. Ao disparar, o *token* segue para o lugar CON_OFF definindo a falha da concessionária. As transições TACT1_1 e CON_R ficam habilitadas e com disparo da primeira, o módulo reserva (gerador1) é ativado removendo um *token* do lugar MAIN1_1 e o colocando em GEN1_ON. Para que o gerador1 falhe, a transição GEN1_F é disparada transferindo o *token* para GEN1_OFF (falha do gerador1), habilitando a transição TACT1_2. Com o diparo desta transição, o *token*, que estava em MAIN1_2, é transferido para GEN2_ON, acionando o gerador2.

Para que o módulo principal (concessionária) esteja desativado, um *token* está no lugar CON_OFF habilitando a transição CON_R para reparar o módulo principal. Disparada esta transição, o *token* é deslocado de CON_OFF para CON_ON reativando o módulo principal. Caso um dos módulos reserva esteja ativado (um *token* nos lugares GEN1_ON e/ou GEN2_ON), a transições instantâneas R_MAIN1_1, R_MAIN1_2 e R_MAIN1_3 são automaticamente disparadas removendo o *token* de volta para um dos lugares MAIN1_1, MAIN1_2 e MAIN1_3, desativando qualquer um dos geradores.

A Figura 5.11 apresenta um modelo SPN baseado num modelo *cold stanby* mais complexo com o objetivo representar a redundância UPS1/UPS2/UPS3. Este modelo ilustra

a disposição de dois UPSs funcionando simultaneamente, e, caso um falhe, o UPS reserva é acionado automaticamente. Esse subsistema (*Part II*) é considerado em funcionamento se ao menos um dos UPSs esteja em funcionamento. A disponibilidade é calculada pela seguinte expressão de probabilidade: P{((#UPS1_ON = 1) AND (#UPS3_ON = 1)) OR ((#UPS2_ON = 1) AND (#UPS3_ON = 1)) OR ((#UPS1_ON = 1) AND (#UPS2_ON = 1))}.

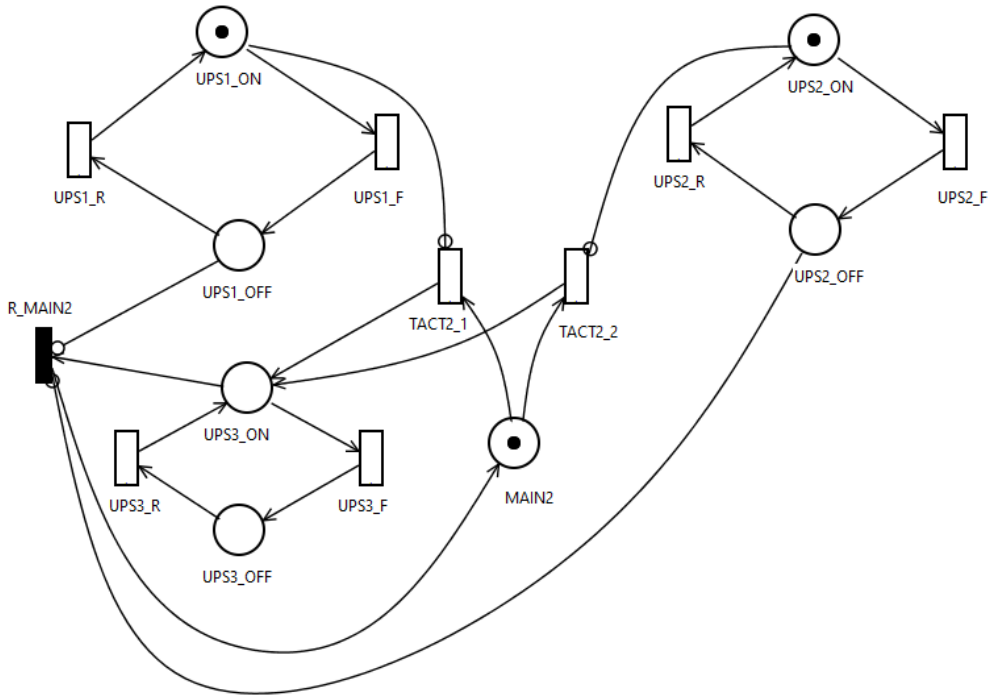

Figura 5.11: Modelo SPN referente à redundância dos UPS1/UPS2/UPS3

Os lugares UPS1_ON e UPS2_ON representam o estado de funcionamento dos módulos principais (UPS). Para que qualquer um destes módulos falhe, as transições UPS1_F ou UPS2_F tem que ser disparadas. Uma vez disparada a transição UPS1_F, um *token* é removido do lugar UPS1_ON e colocado no UPS1_OFF. Na ausência de *tokens* em UPS1_ON, a transição TACT2_1 é habilitada. Com o disparo desta transição, o módulo reserva (UPS3) é ativado removendo um *token* do lugar MAIN2 e adicionando em UPS3_ON. Assumindo o disparo da transição UPS2_F, um *token* é removido do UPS2_ON e colocado no UPS2_OFF. Na ausência de *tokens* em UPS2_ON, a transição TACT2_2 é habilitada.

Para que o módulo principal (UPS1) esteja desativado, um *token* está no lugar UPS1_OFF

habilitando a transição UPS1_R para reparar o módulo principal. Disparada esta transição, o *token* é deslocado de UPS1_OFF para UPS1_ON. Observando que o módulo reserva está ativado com um *token* em UPS3_ON, a transição instantânea R_MAIN2 é automaticamente disparada, removendo o *token* de UPS3_ON para MAIN2, desativando o módulo reserva (UPS3). Comportamento similar ocorre se o módulo desativado for o UPS2.

Obtidas as disponibilidades dos modelos SPN, o valor referente a redundância concessionária/gerador1/gerador2 (*Part I*) é atribuído ao bloco CON_GEN, e a disponibilidade referente a redundância UPS1/UPS2/UPS3 (*Part II*) é atribuída ao bloco GroupUPS, ambos do modelo RBD representado na Figura 5.12. Com a avaliação desse RBD, obtém-se a disponibilidade do modelo completo.

Figura 5.12: Modelo RBD referente à Arquitetura TIER II

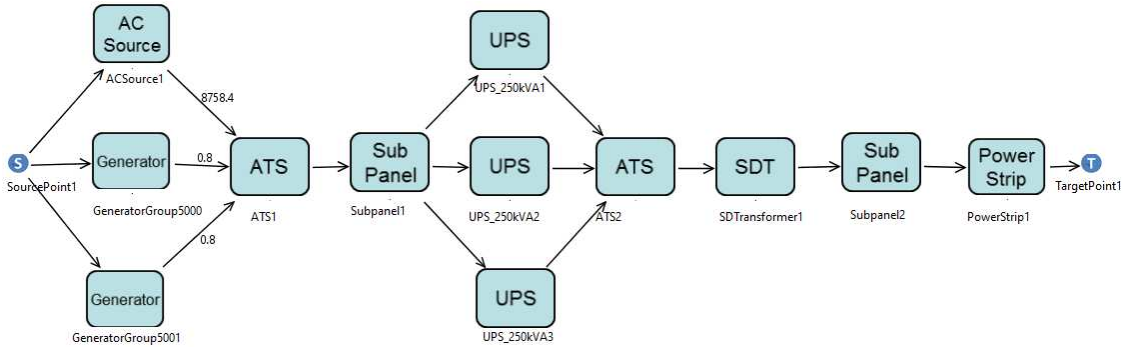

Figura 5.13: Modelo EFM referente à Arquitetura TIER II

TIER III

A arquitetura TIER III adotada neste estudo de caso, está no terceiro nível de complexidade da classificação TIER, possuindo uma redundância $N+1$ (mínimo) nos componentes de fornecimento elétrico. Esta arquitetura possui dois caminhos de distribuição, um principal (maior redundância) e outro secundário, funcionando simultaneamente conforme descrito na Figura 5.14.

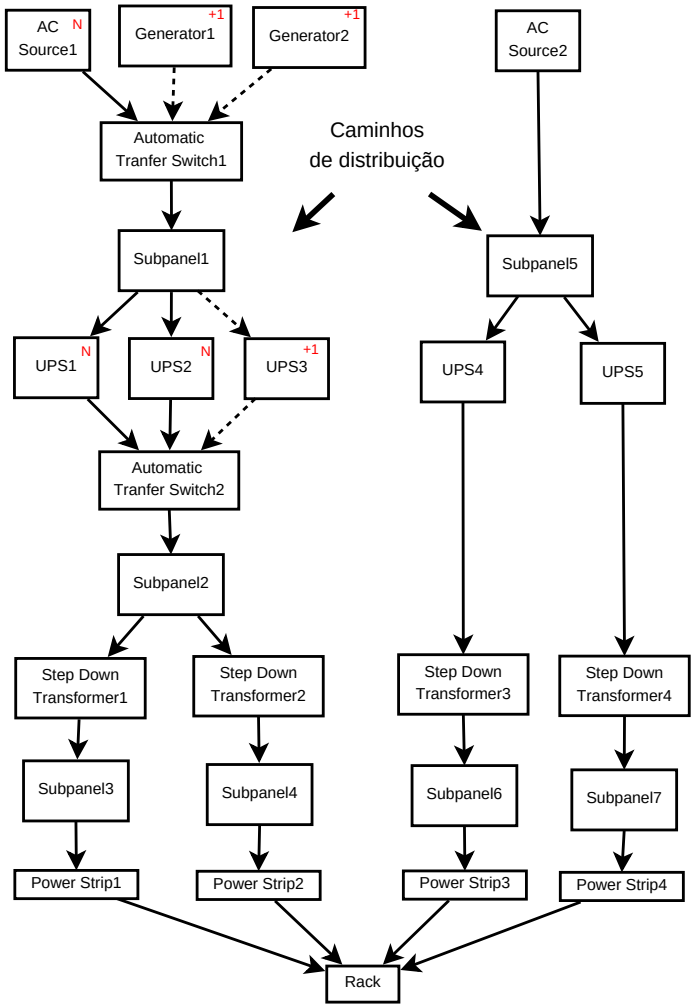

Figura 5.14: Submodelo da Arquitetura TIER III

Para obter a disponibilidade das redundâncias da arquitetura TIER III, o modelo representado na Figura 5.10 foi usado para obter a disponibilidade da redundância concessionária/gerador1/gerador2, e o modelo representado na Figura 5.11 é utilizado para obter a disponibilidade da redundância UPS1/UPS2/UPS3.

Avaliados estes modelos SPN, a disponibilidade da redundância concessionária/gerador1/gerador2 é atribuída ao bloco CON_GEN, e a redundância UPS1/UPS2/UPS3 tem sua disponibilidade ao bloco GroupUPSno modelo RBD representado na Figura 5.15. O resultado da disponiblidade computada por esse modelo RBD é usada como parâmetro no modelo EFM da Figura 5.16.

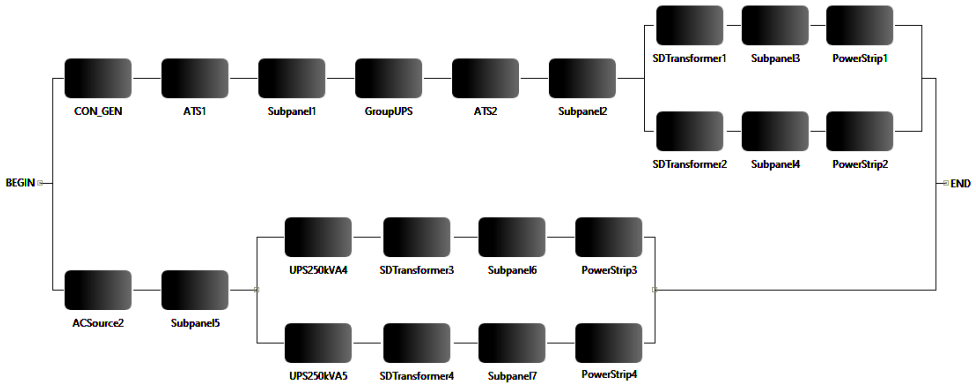

Figura 5.15: Modelo RBD referente à Arquitetura TIER III

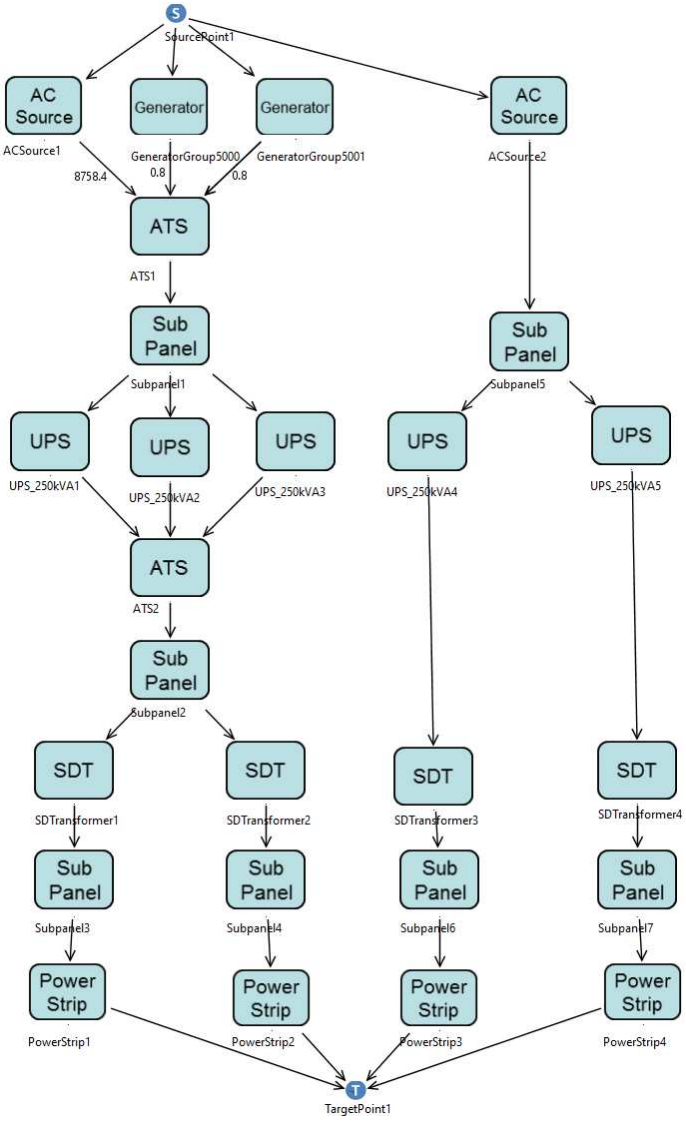

Figura 5.16: Modelo EFM referente à Arquitetura TIER III

TIER IV

Por fim, a arquitetura TIER IV adotada neste estudo de caso, está no nível mais alto de complexidade da classificação TIER. Esta arquitetura possui uma redundância de $2(N+1)$ nos componentes de fornecimento de energia elétrica. A arquitetura TIER IV possui dois caminhos de distribuição idênticos funcionando simultaneamente conforme ilustrado na Figura 5.17.

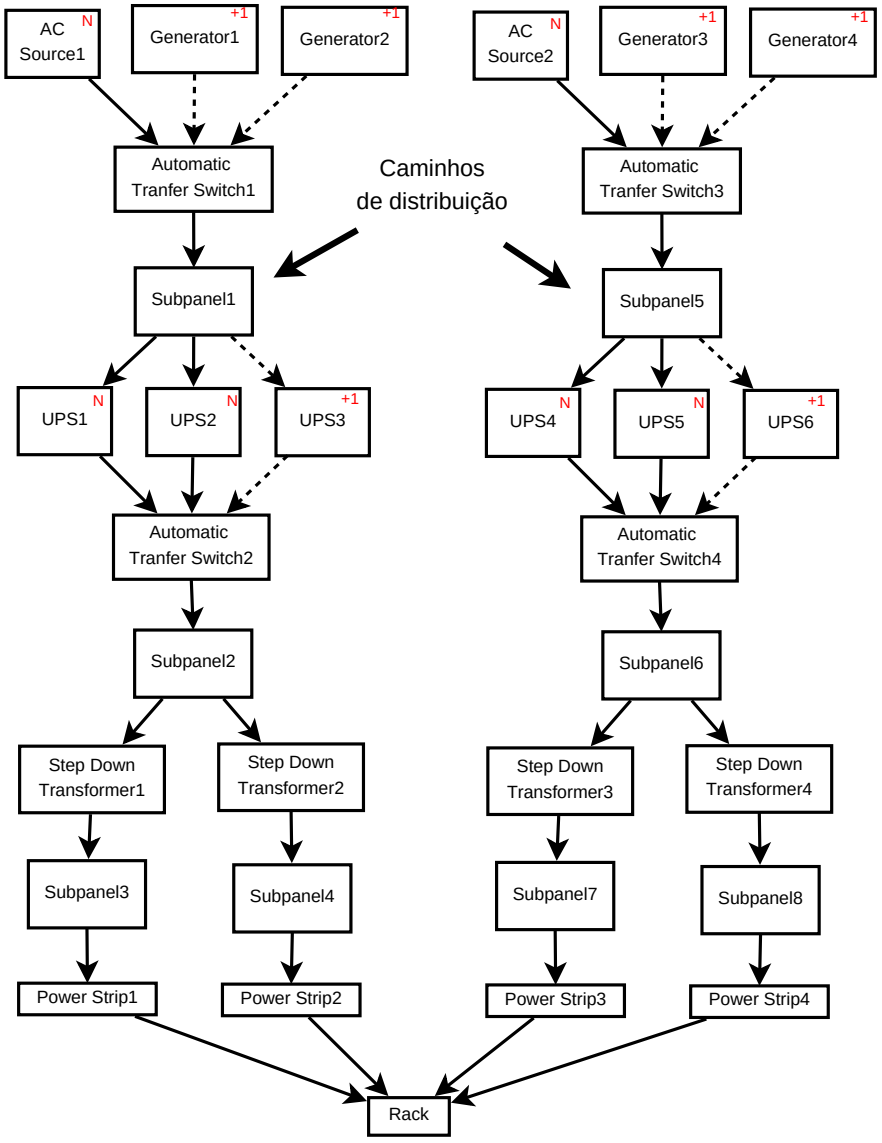

Figura 5.17: Submodelo da Arquitetura TIER IV

A arquitetura TIER IV usa os modelos das Figuras 5.10 e 5.11 para obter a disponibili-

dade dos submodelos conforme descrito anteriormente.

As disponibilidades obtidas pela avaliação do modelos SPN que representa as redundâncias concessionária/gerador1/gerador2 são atribuídas aos blocos CON_GEN1 e CON_GEN2; as disponibilidades avaliadas nas redundâncias UPS1/UPS2/UPS3 são atribuídas as blocos GroupUPS1 e GroupUPS2. Dessa forma, obtem-se o modelo RBD da Figura 5.18. Novamente, esse modelo é avaliado e a disponibilidade obtida é utilizada como parâmetro do modelo EFM proposto para representar a arquitetura TIER IV, conforme pode ser visto na Figura5.19.

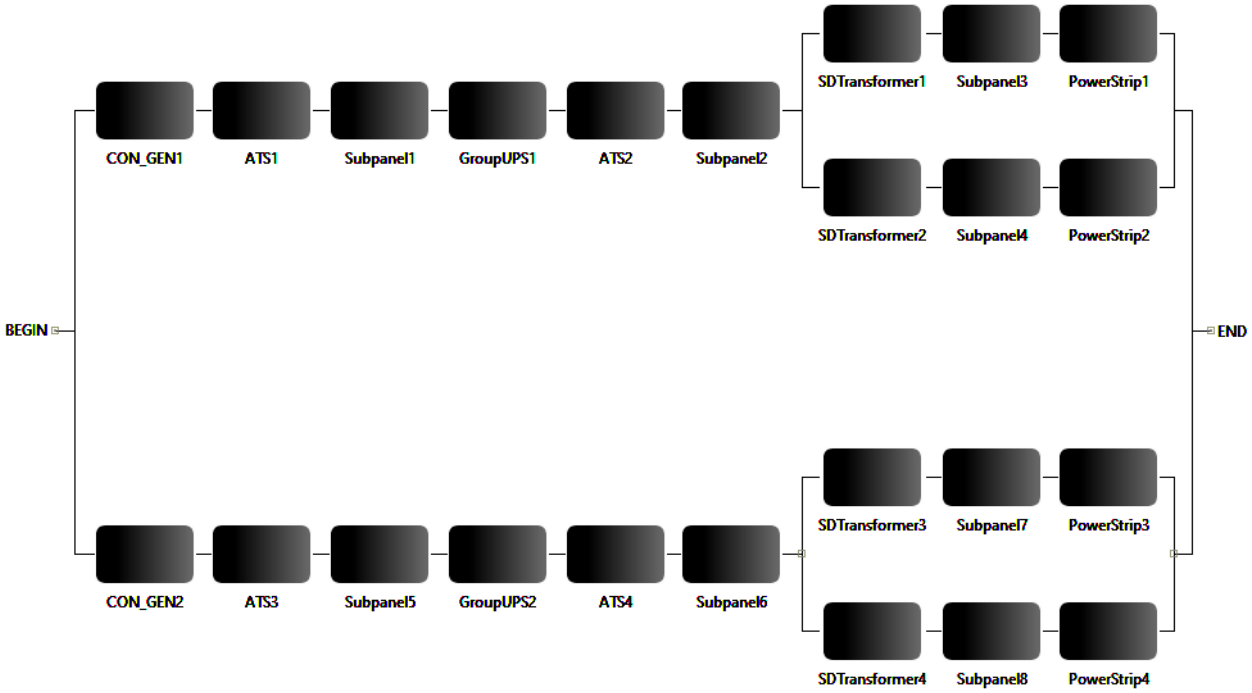

Figura 5.18: Modelo RBD referente à Arquitetura TIER IV

Novamente, esse modelo é avaliado e a disponibilidade obtida é utilizada como parâmetro do modelo EFM proposto para representar a arquitetura TIER IV, conforme pode ser visto na Figura5.19.

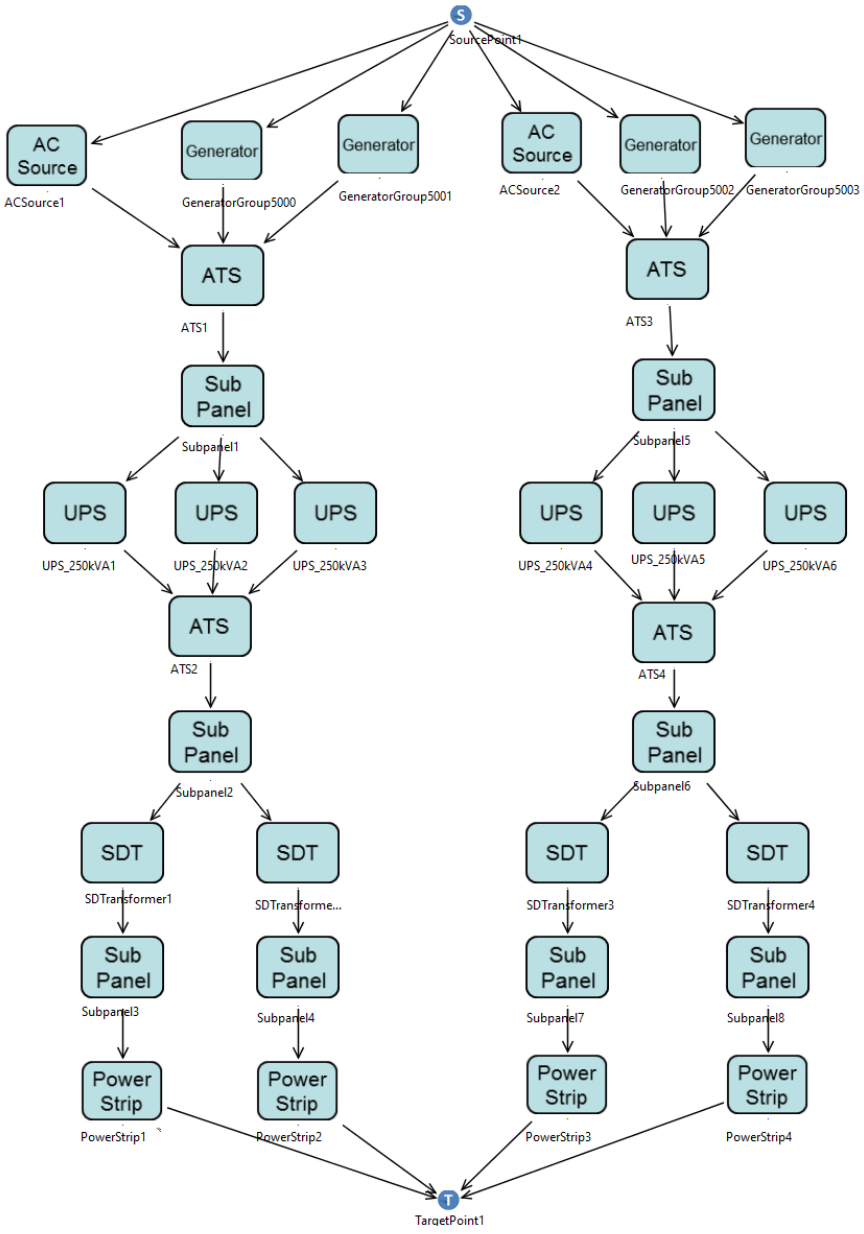

Figura 5.19: Modelo EFM referente à Arquitetura TIER IV

Aplicação do Algoritmo Genético

É gerada uma população inicial com 100 indivíduos. Feito isso, o algoritmo genético prossegue suas etapas. Dois indivíduos são escolhidos pelo processo de seleção para o cruzamento com 5% de chance de haver mutação. Os dois cromossomos escolhidos geram dois novos cromossomos que são inseridos à população. Toda a população é avaliada com o objetivo de se

descartar os dois piores cromossomos (de menor pontuação).A função *Fitness* utilizada foi a mesma empregada no estudo de caso I e descrita na Equação 5.1. O processo se repete, a partir da seleção até que a população de cromossomos apresente soluções satisfatórias. Cada repetição do processo é uma geração.

Foi observado que a população de cromossomos não evoluía com um número de gerações maior que 5000, chegando a convergência. Sendo assim, este o número de gerações foi adotada para a otimização das arquiteturas analisadas. Desta forma, após as 5000 gerações, a população de cromossomos obtem um conjunto de soluções satisfatórias, ou seja otimizadas.

5.2.2 Resultados

A Tabela 5.2 apresenta uma comparação entre os dados da população inicial gerada e os resultados obtidos na otimização das 4 arquiteturas TIER pelo algoritmo genético proposto.

Arquitetura TIER I	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
População Inicial	122322,20	203,39	3,3643	-
Otimização	116004,52	140,13	3,4005	463,34
Melhora (%)	5,16	31,10	1,06	-
Arquitetura TIER II	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
População Inicial	292795,44	283,82	3,3478	-
Otimização	269377,66	175,30	3,4158	905,07
Melhora (%)	7,99	38,23	1,99	-
Arquitetura TIER III	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
População Inicial	426433,72	288,13	7,1242	-
Otimização	388457,16	135,09	7,1986	965,16
Melhora (%)	8,90	53,11	1,03	-
Arquitetura TIER IV	Custo (USD)	Exergia (J)	Disponibilidade (9s)	Tempo (s)
População Inicial	614069,72	360,41	6,8386	-
Otimização	556867,42	169,01	6,9867	1950,44
Melhora (%)	9,31	53,10	2,11	-

Tabela 5.2: Otimização das Arquiteturas TIER Adotadas.

É possível verificar que otimização pelo algoritmo genético, já validado no estudo de caso I, obtêm resultados consideráveis em todas as arquiteturas, observando-se um aumento na disponibilidade e uma redução na exergia operacional e no custo total. Apesar de uma complexidade bem maior que a do estudo de caso I, o tempo de otimização é aceitável levando em consideração a quantidade de variáveis do problema.

Pode-se observar que quanto mais complexa a arquitetura, maior a sua disponibilidade, exergia e custo. Vale ressaltar que o objetivo da classificação TIER é classificar os *data centers* de acordo com o redundância da infraestrutura elétrica. Cada arquitetura apresentada neste estudo, deve ser adotada de acordo com a demanda disponibilidade dos serviços a serem oferecidos no *data center*, ou seja, o quanto eles precisar ser tolerantes a desligamentos por falha ou para manutenção

Capítulo 6

Conclusão

Devido a grande demanda de toda sociedade global por serviços de internet, a disponibilidade dos *data centers* que dão suporte a esses serviços vem se tornando cada vez mais indispensável. Não só usuários em seu dia-a-dia, mas também empresas de todo porte dependem, de maneira ininterrupta, que esses serviços estejam em pleno funcionamento para realizar suas operações. Isso significa que qualquer interrupção no serviço de Internet pode causar grades prejuízos a muitas empresas simultaneamente, deixando a economia passível de sofrer impactos.

Uma prática muito utilizada para aumentar a tolerância a falhas e, consequentemente, a disponibilidade nos *data centers* é aumentar a redundância introduzindo mais equipamentos à infraestrutura destes *data centers*. Entretanto, esta prática leva a um aumento do custo e do impacto ambiental causando prejuízos econômicos.

Aliado a estes fatores, os projetistas dos *data centers* não possuem recursos integrados de avaliação de disponibilidade, custo e sustentabilidade, dificultando a confecção de projetos que concebam infraestruturas que apresentem uma boa tolerância a falhas equilibrando custo e sustentabilidade.

Sendo assim, este trabalho propôs a aplicação de um algoritmo genético para otimização de arquiteturas elétricas de *data center* com o objetivo de aumentar a disponibilidade reduzindo o custo e o impacto ambiental. As arquiteturas foram otimizadas a partir de modelos

SPN, RBD e EFM, criados no ambiente Mercury, de maneira integrada com o algoritmo genético através da liguagem de *script* do Mercury, buscando maximizar a disponibilidade, minimizando o custo e a exergia.

Foram analisados dois estudos de caso:

- No primeiro, foram otimizadas cinco arquiteturas elétricas básicas de *data center* através do algoritmo genético proposto e de um algoritmo de força bruta, que analisa todas as combinações possíveis de equipamentos. Na comparação, foi observado que o algoritmo genético obteve resultados muito próximos do algoritmo de força bruta entretanto em um tempo extremamente menor, validando, assim, a eficiência do algoritmo genético.
- Já o segundo, e mais complexo, estudo de caso, aplicou o algoritmo genético proposto na otimização de quatro arquiteturas enquadradas na classificação TIER, sendo observados excelentes resultados de otimização em um tempo aceitável.

Pode-se concluir que os Algoritmos Genéticos são eficientes e eficazes, pois retornam um grupo de soluções boas muito próximas da melhor solução encontrada através do algoritmo de força bruta em um tempo muito menor. Essa diferença de tempo se torna cada vez mais acentuada a medida que se aumenta a complexidade da arquitetura a ser otimizada. Foi observada a facilidade da integração do algoritmo genético proposto com os modelos RBD, SPN e EFM através do ambiente Mercury com sua linguagem de *script*.

6.1 Contribuições

Este trabalho propôs um algoritmo genético integrado a modelos formais de avaliação de confiabilidade, sustentabilidade e custo, suportados pelo Mercury, de maneira a otimizar as métricas de disponibilidade, exergia e custo de arquiteturas elétricas de *data center*. Aliado a isso, a metodologia adotada propõe o passo-a-passo da otimização de arquiteturas em quaisquer níveis de complexidade. Além disso, é importante ressaltar que os modelos analisados nos estudos de caso também representam contribuições dessa pesquisa. Para um melhor entendimento, a seguir, seguem as contribuições dessa pesquisa:

- **Metodologia:** Através da linguagem de *script* provida pelo ambiente Mercury, a metodologia proposta permite constituir a comunicação entre o algoritmo genético e o Mercury para ajustar os modelos RBD, SPN e EFM. Dessa forma, é possível computar a disponibilidade, o custo e a exergia e utilizar para avaliar os cromossomos e poder, assim, realizar a otimização proposta.

- **Modelagem:** Um conjunto de modelos foi proposto para avaliar a disponibilidade, sustentabilidade e custo de diversas arquiteturas de *data centers*, de complexidades diferentes. Vale ressaltar que redundâncias complexas através de modelos SPN foram propostos. Além disso, tais modelos também foram representados na linguagem de *script* do Mercury..

- **Algoritmo Genético:** Foi proposto um algoritmo genético capaz de se comunicar com o Mercury, de maneira integrada, para otimizar, com eficiência, arquiteturas elétricas de *data center*. Através de uma lista de equipamentos, este algoritmo seleciona aleatoriamente cada componente, de acordo com o tipo estabelecido na arquitetura, desta lista para gerar os cromossomos da população inicial. Ao fim de todos os ciclos, o algoritmo genético retorna um grupo de soluções boas com ganho na disponibilidade e redução no custo e no impacto ambiental.

6.2 Trabalhos Futuros

Algoritmos genéticos podem ser aplicados a arquiteturas de *data centers* de grande porte, já que usar *Força Bruta* se torna inviável devido a alta complexidade das arquiteturas. Aliado a isto, as simulações dos modelos propostos se aproximam cada vez mais do mundo real, podendo, num momento posterior, este estudo ser aplicado não só a infraestruturas elétricas de *data center* mais complexas como também as demais infraestruturas de refrigeração e TI.

Como trabalhos futuros, trabalharemos a aplicação deste estudo nas três infraestruturas de *data center*, infraestrutura de TI, infraestrutura de refrigeração e a infraestrutura elétrica, em diversos graus de complexidade computacional e de maneira integrada.

Outras possibilidades de trabalhos futuros:

- Propor outros algoritmos multiobjetivos (ex., colônia de formigas) para comparar com o AG proposto nesse trabalho.
- Propor estratégias de otimização via equações lineares. Através de equações fechadas, pode-se obter e realizar a otimização das métricas nos sistemas de interesse.
- Propor outras métricas para serem analisadas e otimizadas pelo AG. Por exemplo, pode-se levar em consideração confiabilidade, emissão de CO2, custo de manutenção, etc.
- Propor otimização multiobjetivo levando em consideração outros modelos, por exemplo, modelos com manutenção preventiva e corretiva dos sistemas analisados para computar a dependabilidade.

Referências Bibliográficas

[1] E. Bauer and R. Adams, *Reliability and Availability of Cloud Computing*. IEEE PRESS, 2012.

[2] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pp. 124–131, IEEE, 2009.

[3] J. Figueiredo, P. Maciel, G. Callou, E. Tavares, E. Sousa, and B. Silva, "Estimating reliability importance and total cost of acquisition for data center power infrastructures," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pp. 421–426, oct. 2011.

[4] G. Callou, P. Maciel, D. Tutsch, J. Ferreira, J. Araújo, and R. Souza, "Estimating sustainability impact of high dependable data centers: A comparative study between brazilian and us energy mixes," *Computing*, vol. 95, no. 12, pp. 1137–1170, 2013.

[5] V. Dao, I. Langella, and J. Carbo, "From green to sustainability: Information technology and an integrated sustainability framework," *The Journal of Strategic Information Systems*, vol. 20, no. 1, pp. 63–79, 2011.

[6] G. L. Lunardi, R. S. Frio, and M. d. M. Brum, "Tecnologia da informação e sustentabilidade: levantamento das principais práticas verdes aplicadas à área de tecnologia," *Gerais : Revista Interinstitucional de Psicologia*, vol. 4, pp. 159 – 172, 12 2011.

[7] B. Weihl, E. Teetzel, J. Clidaras, C. Malone, J. Kava, and M. Ryan, "Sustainable data centers," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 17, no. 4, pp. 8–12, 2011.

[8] A. J. Chen, M.-C. Boudreau, and R. T. Watson, "Information systems and ecological sustainability," *Journal of Systems and Information Technology*, vol. 10, no. 3, pp. 186–201, 2008.

[9] R. Meulen, "Gartner: Data centres account for 23% of global ict c02 emissions." `http://www.gartner.com/newsroom/id/530912`, 2007. Accessed: 2017-07-26.

[10] A. W. Hodges and M. Rahmani, "Fuel sources and carbon dioxide emissions by electric power plants in the united states," 2010.

[11] G. Callou, *Planning of Sustainable Data Centers with High Availability: An Integrated Modeling Approach to Evaluate and Optimize Sustainability, Dependability and Cost of Data Center Systems*. LAP LAMBERT Academic Publishing, 2014.

[12] T. I. Association, "Telecommunications infrastructure sstandard for data centers ansi/tia-942," 2005.

[13] U. I. P. Services, "Tier certification." `https://uptimeinstitute.com`.

[14] P. S. Marin, *Data Centers-Desvendando Cada Passo-Conceitos, Projeto, Infraestrutura Física e Eficiência Energética*. São Paulo - SP: Érica, 2011.

[15] W. Zucchi and A. Amâncio, "Construindo um data center," *Revista USP*, no. 97, pp. 43–58, 2013.

[16] M. Levy and J. O. Hallstrom, "A new approach to data center infrastructure monitoring and management (dcimm)," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–6, Jan 2017.

[17] R. Rosa, M. Aranda, and P. Antoniolli, "Segurança física em datacenters: estudo de caso," *Refas-Revista Fatec Zona Sul*, vol. 3, no. 4, pp. 1–22, 2017.

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

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

[20] R. Miller, "A look inside amazon's data centers." http://www.datacenterknowledge.com/archives/2011/06/09/a-look-inside-amazons-data-centers/, 2011. Accessed: 2015-12-25.

[21] U. Hölzle, "More computing, less power." https://googleblog.blogspot.com.br/2009/01/more-computing-less-power.html, 2009. Accessed: 2015-12-25.

[22] G. Callou, J. Ferreira, P. Maciel, D. Tutsch, and R. Souza, "An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost," *Energies*, vol. 7, no. 1, pp. 238–277, 2014.

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

[24] J. F. Meyer and W. H. Sanders, "Specification and construction of performability models," in *Proceedings of the Second International Workshop on Performability Modeling of Computer and Communication Systems*, pp. 28–30, 1993.

[25] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, pp. 11 – 33, jan.-march 2004.

[26] P. Maciel, K. S. Trivedi, R. Matias, and D. S. Kim, *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, ch. Dependability Modeling. Premier Reference Source, Igi Global, 2011.

[27] E. SOUSA, *Modelagem de desempenho, dependabilidade e custo para o planejamento de infraestruturas de nuvens privadas.* PhD thesis, 2015.

[28] D. K. Pradhan, *Fault-tolerant computer system design.* Prentice-Hall, 1996.

[29] C. Araújo, "Avaliação e modelagem de desempenho para planejamento de capacidade do sistema de transferência eletrônica de fundos utilizando tráfego em rajada," Master's thesis, Dissertação, Universidade Federal de Pernambuco, Centro de Informática, 2009.

[30] C. Ebeling, *An Introduction to Reliability and Maintainability Engineering*. Waveland Press, 1997.

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

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

[33] J. W. Rupe, "Reliability of computer systems and networks fault tolerance, analysis, and design," *IIE Transactions*, vol. 35, no. 6, pp. 586–587, 2003.

[34] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysisof a disaster-recovery-as-a-service solution," *Computing*, vol. V, pp. 1–26, 2017.

[35] E. Bauer, R. Adams, and D. Eustace, *Beyond redundancy: how geographic redundancy can improve service availability and reliability of computer-based systems*. John Wiley & Sons, 2011.

[36] T. J. Kotas, "The exergy method of thermal plant analysis," 1985.

[37] J. Szargut, D. Morris, and F. Steward, "Energy analysis of thermal, chemical, and metallurgical processes," 1988.

[38] G. Alves, P. Maciel, R. Lima, and F. Magnani, *Supply Chain Management - Applications and Simulations*, ch. 8 - Business and Environment Performance Evaluation in Supply Chains: a Formal Model-Driven Approach, pp. 157–182. InTech, 2011.

[39] J. Ferreira, G. Callou, J. Dantas, R. Souza, and P. Maciel, "An algorithm to optimize electrical flows," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 109–114, IEEE Computer Society, 2013.

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

[41] P. Merlin and D. Farber, "Recoverability of communication protocols–implications of a theoretical study," *IEEE transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[42] M. A. Marsan, "Stochastic petri nets: an elementary introduction," in *European Workshop on Applications and Theory in Petri Nets*, pp. 1–29, Springer, 1988.

[43] K. Jensen, "Coloured petri nets: A high level language for system design and analysis," in *International Conference on Application and Theory of Petri Nets*, pp. 342–416, Springer, 1989.

[44] V. Janoušek, *Modelling Objects by Petri Nets*. PhD thesis, PhD. thesis, Brno University of Technology, Brno, Czech Republic, 1998.

[45] E. C. d. ANDRADE, *Modelagem e análise de mecanismos de tratamento de interrupções em infraestruturas computacionais dos sistemas distribuídos*. PhD thesis, 2014.

[46] 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.

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

[48] B. SILVA, *A framework for availability, performance and survivability evaluation of disaster tolerant cloud computing systems*. PhD thesis, 2016.

[49] M. A. 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, p. 2, 1998.

[50] G. Balbo, "Introduction to stochastic petri nets," *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands, July 3-7, 2000: Revised Lectures*, 2001.

[51] C. Araújo, P. Maciel, M. Torquato, G. Callou, and E. Andrade, "Availability evaluation of digital library cloud services," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pp. 666–671, june 2014.

[52] 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.

[53] S. S. Rao and S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[54] B. Nogueira, *Exploração multiobjetivo do espaço de projeto de sistemas embarcados de tempo-real não críticos*. PhD thesis, Universidade Federal de Pernambuco. Centro de Informática, 2015.

[55] S. Ávila, *Otimização multiobjetivo e análise de sensibilidade para concepção de dispositivos: aplicação: síntese de antenas refletoras para comunicação via satélite*. PhD thesis, Universidade Federal de Santa Catarina, 2006.

[56] C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information systems*, vol. 1, no. 3, pp. 129–156, 1999.

[57] C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE computational intelligence magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[58] F. Y. Edgeworth, "Mathematical physics, an essay on the application of mathematics to the moral sciences," 1881.

[59] V. Pareto, "Cours d'economie politique," *F. Rouge, Lausanne*, 1896.

[60] R. Azuma, "Otimização multiobjetivo em problema de estoque e roteamento gerenciados pelo fornecedor," Master's thesis, Universidade Estadual de Campinas, 2011.

[61] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading: Addison-Wesley, 1989.

[62] J. H. Holland, *Adaptation in Natural and Artificial System*. University of Michigan, 1975.

[63] H. Oliveira, "Algoritmo evolutivo no tratamento do problema de roteamento de veículos com janela de tempo." Departamento de Ciência da Computação, Universidade Federal de Lavras, 2005. Monografia (Graduação em Bacharelado em Ciências da Computação).

[64] D. Vasconcellos, I. Abril, and V. Martínez, "Compensación de potencia reactiva en sistemas desbalanceados utilizando algoritmos genéticos," *Ingeniare. Revista chilena de ingeniería*, vol. 20, no. 3, pp. 284–292, 2012.

[65] D. Lucas, "Algoritmos genéticos: uma introdução." Universidade Federal do Rio Grande do Sul, 2002. Apostila elaborada sob a orientação de Luís Otávio Álvares, para a disciplina de Ferramentas de Inteligência Artificial.

[66] M. Pacheco, "Algoritmos genéticos: princípios e aplicações," 1999.

[67] O. Junior, "Otimização de horários em instituições de ensino superior através de algoritmos genéticos," Master's thesis, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina., 2000.

[68] D. Coury, M. Oleskovicz, and S. Souza, "Genetic algorithms applied to a faster distance protection of transmission lines," *Sba: Controle & Automação Sociedade Brasileira de Automatica*, vol. 22, no. 4, pp. 334–344, 2011.

[69] G. Pappa, "Seleção de atributos utilizando algoritmos genéticos multiobjetivos," Master's thesis, Programa de Pós Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná. Curitiba, 2002.

[70] D. Lucena, "Algoritmos evolutivo multiobjetivo para seleção de variáveis em problemas de calibração multivariada," Master's thesis, Universidade Federal de Goiás, 2013.

[71] R. Hinterding, "Representation, mutation and crossover issues in evolutionary computation," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2, pp. 916–923, IEEE, 2000.

[72] V. Dhar and R. Stein, *Seven methods for transforming corporate data into business intelligence*. Prentice Hall Englewood Cliffs, New Jersey, 1997.

[73] R. de Castro, *Otimização de estruturas com multi-objetivos via algoritmos genéticos*. PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2001.

[74] I. Marques, "Predição de séries temporais utilizando algoritmos genéticos," Master's thesis, Universidade Federal do Rio Grande do Sul. Instituto de Informática, 2012.

[75] A. Geyer-Schulz, *Fuzzy rule-based expert systems and genetic machine learning*, vol. 3. Physica Verlag, 1997.

[76] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," 1995.

[77] R. Dawkins, *A escalada do monte improvável: uma defesa da teoria da evolução*. Editora Companhia das Letras, 1996.

[78] L. Davis, "Handbook of genetic algorithms," 1991.

[79] M. Wall, "Galib: A c++ library of genetic algorithm components," *Mechanical Engineering Department, Massachusetts Institute of Technology*, vol. 87, p. 54, 1996.

[80] S. Nesmachnow, "Algoritmos genéticos paralelos y su aplicación al diseño de redes de comunicaciones confiables," Master's thesis, Universidad de la República de Montevideo. Intituto de Computación - Facultad de Ingeniería, 2004.

[81] R. Yamini, "Power management in cloud computing using green algorithm," in *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pp. 128–133, march 2012.

[82] K.Mukherjee and G.Sahoo, "Green cloud: An algorithmic approach," *International Journal of Computer Applications*, vol. 9, no. 9, pp. 1–6, 2010.

[83] I. Kar, R. N. R. Parida, and H. Das, "Energy aware scheduling using genetic algorithm in cloud data centers," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 3545–3550, March 2016.

[84] X. Wang, X. Wang, K. Zheng, Y. Yao, and Q. Cao, "Correlation-aware traffic consolidation for power optimization of data center networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 992–1006, April 2016.

[85] G. Portaluri and S. Giordano, "Power efficient resource allocation in cloud computing data centers using multi-objective genetic algorithms and simulated annealing," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pp. 319–321, Oct 2015.

[86] N. K. Sharma and R. M. R. Guddeti, "Multi-objective resources allocation using improved genetic algorithm at cloud data center," in *2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 73–77, Oct 2016.

[87] C.-W. Ang and C.-K. Tham, "Analysis and optimization of service availability in a ha cluster with load-dependent machine availability," *IEEE Transactions on parallel and distributed systems*, vol. 18, no. 9, 2007.

[88] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[89] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[90] L. Luo, H. Li, X. Qiu, and Y. Tang, "A resource optimization algorithm of cloud data center based on correlated model of reliability, performance and energy," in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 416–417, Aug 2016.

[91] L. D. Chou, H. F. Chen, F. H. Tseng, H. C. Chao, and Y. J. Chang, "Dpra: Dynamic power-saving resource allocation for cloud data center using particle swarm optimization," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2017.

[92] M. Ghamkhari, A. Wierman, and H. Mohsenian-Rad, "Energy portfolio optimization of data centers," *IEEE Transactions on Smart Grid*, vol. 8, pp. 1898–1910, July 2017.

[93] S. Bosse, N. Jamous, F. Kramer, and K. Turowski, "Introducing greenhouse emissions in cost optimization of fault-tolerant data center design," in *2016 IEEE 18th Conference on Business Informatics (CBI)*, vol. 01, pp. 163–172, Aug 2016.

[94] M. Anan, N. Nasser, A. Ahmed, and A. Alfuqaha, "Optimization of power and migration cost in virtualized data centers," in *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–5, April 2016.

[95] H. Ziafat and S. M. Babamir, "Towards optimization of availability and cost in selection of geo-distributed clouds datacenter," in *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–5, Oct 2016.

[96] B. Silva, R. Matos, G. Callou, J. Figueiredo, J. Dantas, A. Lobo Júnior, V. Alves, and P. Maciel, "Mercury: An integrated environment for performance and dependability

evaluation of general systems," in *45th Dependable Systems and Networks Conference*, vol. v, pp. 1–6, 2015.

[97] D. M. OLIVEIRA, R. MATOS, J. DANTAS, J. FERREIRA, B. Silva, G. Callou, P. MACIEL, and A. BRINKMANN, "Advanced stochastic petri net modeling with the mercury scripting language," in *11th EAI International Conference on Performance Evaluation Methodologies and Tools*, 2017.

Apêndice A

Exemplo de arquivo de entrada do AG

Segue um exemplo do arquivo TXT utilizado como dado entrada pelo algoritmo genético proposto.

id;name;type;mttf;eff;aquisitionCost
0;UPS_250KVA0;UPS_250KVA;51021.0;85.005;51215.0
1;UPS_250KVA0;UPS_250KVA;63334.0;87.005;57987.0
2;UPS_250KVA0;UPS_250KVA;40630.0;96.005;63125.0
3;UPS_250KVA0;UPS_250KVA;35562.0;86.005;59621.0
4;UPS_250KVA0;UPS_250KVA;48436.0;92.005;60589.0
5;UPS_250KVA1;UPS_250KVA;41631.0;89.005;65513.0
6;UPS_250KVA1;UPS_250KVA;32787.0;92.005;62122.0
7;UPS_250KVA1;UPS_250KVA;48250.0;99.005;56329.0
8;UPS_250KVA1;UPS_250KVA;54112.0;91.005;61423.0
9;UPS_250KVA1;UPS_250KVA;72996.0;86.005;64951.0
0;SDTransformer2;SDTransformer;329300.5;89.725;468.5
1;SDTransformer2;SDTransformer;148598.5;88.725;647.5
2;SDTransformer2;SDTransformer;186861.5;95.725;478.5
3;SDTransformer2;SDTransformer;174445.5;91.725;611.5
4;SDTransformer2;SDTransformer;258492.5;98.725;438.5
5;SDTransformer3;SDTransformer;250334.5;96.725;453.5

6;SDTransformer3;SDTransformer;340802.5;85.725;437.5
7;SDTransformer3;SDTransformer;258210.5;84.725;558.5
8;SDTransformer3;SDTransformer;248651.5;87.725;599.5
9;SDTransformer3;SDTransformer;357689.5;86.725;525.5
0;Subpanel4;Subpanel;395784.0;98.915;181.0
1;Subpanel4;Subpanel;400100.0;98.915;200.0
2;Subpanel4;Subpanel;152642.0;93.915;156.0
3;Subpanel4;Subpanel;434474.0;91.915;224.0
4;Subpanel4;Subpanel;289043.0;91.915;224.0
5;Subpanel5;Subpanel;211778.0;87.915;163.0
6;Subpanel5;Subpanel;343562.0;91.915;221.0
7;Subpanel5;Subpanel;197450.0;89.915;167.0
8;Subpanel5;Subpanel;400289.0;84.915;191.0
9;Subpanel5;Subpanel;451949.0;88.915;202.0
0;STS6;STS;49094.0;84.575;694.0
1;STS6;STS;36732.0;85.575;879.0
2;STS6;STS;58284.0;93.575;667.0
3;STS6;STS;55121.0;84.575;723.0
4;STS6;STS;52890.0;97.575;773.0
5;STS7;STS;43094.0;86.575;823.0
6;STS7;STS;40732.0;92.575;752.0
7;STS7;STS;47284.0;95.575;690.0
8;STS7;STS;50121.0;87.575;730.0
9;STS7;STS;55890.0;89.575;795.0
0;PowerStrip8;PowerStrip;119386.756;96.575;153.0
1;PowerStrip8;PowerStrip;169663.756;90.575;210.0
2;PowerStrip8;PowerStrip;274097.756;92.575;208.0
3;PowerStrip8;PowerStrip;158813.756;87.575;195.0
4;PowerStrip8;PowerStrip;258666.756;96.575;167.0
5;PowerStrip9;PowerStrip;100386.756;97.575;174.0
6;PowerStrip9;PowerStrip;130663.756;92.575;200.0
7;PowerStrip9;PowerStrip;201097.756;94.575;156.0

8;PowerStrip9;PowerStrip;230813.756;89.575;184.0

9;PowerStrip9;PowerStrip;134666.756;91.575;190.0

0;GeneratorGroup50010;GeneratorGroup500;52321.0;25.0;52530.0

1;GeneratorGroup50010;GeneratorGroup500;64355.0;29.3;60214.0

2;GeneratorGroup50010;GeneratorGroup500;57542.0;21.1;59789.0

3;GeneratorGroup50010;GeneratorGroup500;49654.0;35.0;63548.0

4;GeneratorGroup50010;GeneratorGroup500;60124.0;20.0;55100.0

5;GeneratorGroup50010;GeneratorGroup500;53521.0;22.0;64521.0

6;GeneratorGroup50010;GeneratorGroup500;54375.0;21.5;56855.0

7;GeneratorGroup50010;GeneratorGroup500;58225.0;28.1;59333.0

8;GeneratorGroup50010;GeneratorGroup500;51100.0;33.0;49200.0

9;GeneratorGroup50010;GeneratorGroup500;58745.0;26.0;65433.0