UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

FRANCISCO MONTE SOUSA SOBRINHO

ESTRATÉGIAS DE OTIMIZAÇÃO BASEADA EM ALGORITMOS MULTIOBJETIVO EVOLUCIONÁRIOS PARA O PROJETO DE SISTEMAS ELÉTRICOS DE DATA CENTERS

RECIFE – PE

2022

FRANCISCO MONTE SOUSA SOBRINHO

ESTRATÉGIAS DE OTIMIZAÇÃO BASEADA EM ALGORITMOS MULTIOBJETIVO EVOLUCIONÁRIOS PARA O PROJETO DE SISTEMAS ELÉTRICOS DE DATA CENTERS

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Prof. Dr. Gustavo Rau de Almeida Callou

RECIFE – PE

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S725e Sousa Sobrinho, Francisco Monte
Estratégias de otimização baseada em algoritmos multiobjetivo evolucionários para o projeto de sistemas elétricos de data centers / Francisco Monte Sousa Sobrinho. - 2022.
109 f. : il.

Orientador: Gustavo Rau de Almeida Callou.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2022.

1. Data center. 2. Disponibilidade. 3. Modelagem. 4. Otimização Multiobjetivo. I. Callou, Gustavo Rau de Almeida, orient. II. Título

CDD 004

FRANCISCO MONTE SOUSA SOBRINHO

ESTRATÉGIAS DE OTIMIZAÇÃO BASEADA EM ALGORITMOS MULTIOBJETIVO EVOLUCIONÁRIOS PARA O PROJETO DE SISTEMAS ELÉTRICOS DE DATA CENTERS

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada á Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em: 27 de julho de 2022.

BANCA EXAMINADORA

Prof. Dr. Gustavo Rau de Almeida Callou (Orientador)
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Bruno Costa e Silva Nogueira
Universidade Federal de Alagoas
Instituto de Computação

Prof. Dr. Francisco Airton Pereira da Silva
Universidade Federal do Piauí
Coordenação do Curso de Sistemas de Informação

Dedico esta dissertação a minha filha Larissa

Agradecimentos

Ao Espirito Santo de Deus pelo dom da vida.

A minha esposa e filha pela paciência das horas que não pude dar a devida atenção.

Aos meus pais pelos incentivos e orações com gestos de carinho que me cercaram durante toda a vida.

Pelo meu amigo Wenderson companheiro nas boas discussões e entendimento dos algoritmos. Aos demais colegas do grupo de pesquisa *System Modeling and Optimization Research Group* (SMORG) da UFRPE que pretendo continuar coloborando e realizando novas pesquisas.

O inegável apoio do meu orientador que nunca desistiu deste projeto. Muito obrigado, professor Gustavo Callou.

Agradeço também aos meus amigos e familiares que sempre acreditaram na realização deste trabalho.

"Melhor é o fim das coisas do que o princípio delas."

(Eclesiastes 7:8-a)

Resumo

Atualmente, há uma demanda crescente por armazenamento de dados em mídias sociais e streaming. Isso gera a necessidade de infraestruturas robustas de *data center* que tenham alta disponibilidade, baixo custo e alta eficiência energética. No entanto, esses objetivos são muitas vezes conflitantes. Por exemplo, uma fonte de energia ininterrupta (UPS) aumentará a disponibilidade do sistema, mas poderá afetar no custo ou consumo de energia. A fim de melhorar o projeto de arquiteturas elétricas de *data centers*, esta dissertação propõe duas estratégias de otimização de múltiplos critérios (disponibilidade, custo e eficiência energética) baseada em algoritmos evolucionários: Algoritmo Genético de Ordenação de Soluções Não-Dominados II (NSGA-II) e Otimização por Enxame de Partículas Multiobjetivo (MOPSO). Diversos trabalhos vêm sendo desenvolvidos com essas técnicas, tais como (HUO et al., 2021), (ZHU; HAN, 2021) e (SRIVASTAVA et al., 2021). Para verificar a aplicabilidade da estratégia proposta, foram apresentados três estudos de caso. O primeiro estudo de caso foi conduzido comparando os resultados obtidos da estratégia baseada no NSGA-II com o algoritmo de força bruta, ambos com finalidade de otimizar os mesmos modelos de infraestruturas elétricas de um *data center*. No segundo estudo de caso foram mantidos os mesmos modelos adotados no primeiro estudo de caso, sendo que a estratégia de otimização utilizada foi baseada no MOPSO, comparando-a com o algoritmo de força bruta. Validadas as estratégias de otimização é proposto o terceiro estudo de caso com novos modelos (mais complexos) de infraestrutura elétrica de *data center* com base nas arquiteturas TIER. Neste último estudo, o algoritmo de força bruta não foi utilizado devido ao aumento da base de dados dos equipamentos, ampliando a complexidade combinatória, utilizou-se portanto, a comparação das duas estratégias propostas.

Os resultados obtidos no primeiro estudo de caso mostraram uma redução no tempo de execução em até 589 vezes em relação a força bruta na modelagem A6 (a mais complexa do projeto), além da última geração trazer conjunto de soluções próxima à curva Pareto ótimo. No segundo estudo a estratégia baseada no MOPSO também se mostrou válida, pois os resultados da última geração, apresentaram uma curva aproximada de Pareto com relação ao algoritmo de força bruta, além de ter tempo de execução 502 vezes menor comparado ao força bruta. Avaliando os resultados do terceiro estudo de caso que faz uso de modelagens TIERs das infraestruturas *data center* pode-se dizer que a estratégia baseada no NSGA-II

trouxe resultados quantitativamente melhores em comparação a estratégia baseada no MOPSO, porém dentro do intervalo de confiança de 95%, constatou-se similaridade entre os resultados das funções objetivo das duas estratégias.

Palavras-chave: *Data Center*, Diagramas de Bloco de Confiabilidade, Disponibilidade, Estratégia, Modelagem, Otimização Multiobjetivo .

Abstract

Nowadays, there is a growing demand for social media and streaming data storage. This demand drives the need for robust data center infrastructures with high availability, low cost, and high energy efficiency. However, these goals are often conflicting. For example, an uninterruptible power supply (UPS) additional will increase availability at cost or consumption of power. To improve the design of electrical data center architectures, this Dissertation adopts a methodology that uses two multiobjective optimization strategies (availability, cost, and energy efficiency) based on evolutionary algorithms: Genetic Algorithm of Non-dominated Solutions Ordering II (NSGA-II) and Swarm Optimization Multi-Objective Particles (MOPSO). Several works have been developed with these techniques, such as (HUO et al., 2021), (ZHU; HAN, 2021) and (SRIVASTAVA et al., 2021).

Three case studies were presented to verify the applicability of the proposed strategies. The first case study was conducted by comparing the results obtained from the strategy based on the NSGA-II with the brute force algorithm to optimize the same electrical infrastructure models of a data center. In the second case study, the same models adopted in the first case study were maintained, and the optimization strategy used was based on MOPSO, comparing it with the brute force algorithm. Once the optimization strategies are validated, the third case study with new (more complex) models of electrical infrastructure in the data center based on TIER architectures. In this previous study, the force algorithm gross was not used due to the increase in the equipment database, expanding the combinatorial complexity. Therefore, the comparison of the two strategies was used proposals.

The results obtained in the first case study showed a reduction in execution time by up to 589 times compared to brute force in the A6 modeling (the most complex of the project), in addition to the last generation bringing a set of solutions close to the Pareto optimal curve. In the second study, the strategy based on MOPSO also proved to be valid, as the results of the last generation brought an approximate Pareto curve within the 95% confidence interval in relation to the brute force algorithm, in addition to having 502 times shorter execution time compared to the brute force algorithm. Evaluating the results of the third case study, whether a strategy based on the NSGA-II obtained better results than a strategy based on the MOPSO of research study analysis can be obtained through similar models among the TIER results of data center infrastructures.

Keywords: Availability, *Data Center*, Reliability Block Diagrams, Modeling, Multi-Objective Optimization and Strategy

Lista de Figuras

Lista de tabelas

Lista de algoritmos

Lista de Siglas

ATS	*Automatic transfer switch*
DRL	*Deep Reinforcement Learning*
DUE	*De-UnderEstimation*
EFM	*Energy Flow Model*
GRASP	*Greedy Randomized Adaptive Search Procedure*
MOPSO	*Multi-Objective Particle Swarm Optimization*
MTTF	*Mean Time to Failure*
MTTR	*Mean Time to Repair*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
RBD	*Reliability Block Diagram*
SDT	*Step Down Transformers*
SPN	*Stochastic Petri Net*
STS	*Static Transfer Switch*
UPS	*Uninterruptible Power Supplies*
VEGA	*Vector Evaluated Genetic Algorithm*

Sumário

1 Introdução

O aumento na necessidade de armazenamento de dados em servidores nos últimos anos, advinda principalmente pela expansão de serviços tecnológicos e aplicações da internet como as mídias sociais e serviços de streaming (LEI; MASANET, 2021), trouxe consigo o desafio para os projetistas desenvolverem robustas infraestruturas de *data center*. A incorporação de um *data center* é uma forma segura de garantir altas taxas de disponibilidade de sistemas cruciais para as organizações como, por exemplo, Sistemas Integrados de Gestão Empresarial (ERP), transações bancárias online, sistemas de comercio eletrônico, banco de dados (WANG et al., 2021). Trata-se de um ambiente controlado, em condições ideais de temperatura e umidade, sistema automático de supressão de incêndio, controle e monitoramento de acesso, fornecimento energético ininterrupto e de qualidade.

Atualmente segue-se a regulamentação da norma (TIA-942, 2005). Inicialmente com abrangência apenas nos EUA e agora com espaço nos órgãos regulamentadores em diversos países, inclusive no Brasil, a regulamentação trata de três tópicos relacionados ao projeto de *data centers*: arquitetura, comunicação elétrica e comunicação mecânica. A construção de um *data center* eficiente é resultado da sinergia destes três fatores.

Conhecido por ambientes de missão crítica, os *data centers* devem estar sempre aptos a operar de forma contínua. Deste modo, é essencial manter infraestruturas que possuam alta disponibilidade para permitir que o ambiente funcione corretamente (MELO et al., 2021). O subsistema elétrico, um dos três pilares da infraestrutura de um *data center* (subsistemas elétrico, refrigeração e tecnologia da informação), possui critérios que precisam ser atendidos por uma equipe capacitada durante seu projeto (ENDO, 2020). Dentre estes critérios, uma infraestrutura elétrica dos *data centers* deve oferecer alta disponibilidade, com custo reduzido e eficiência energética maximizada.

Muitas vezes estes critérios são conflitantes (REIS et al., 2020), por exemplo, considerando duas arquiteturas similares, mas com equipamentos diferentes, onde a primeira possua maior disponibilidade que a segunda. Em contrapartida, a segunda, mesmo com menor disponibilidade, pode ter um custo total de equipamentos inferior em relação à primeira. Para além disso, existe um número crescente de problemas de otimização multiobjetivo reconhecidos em vários campos, como agendamento (WANG et al., 2017), alocação de recursos (MASHWANI; SALHI, 2016) e *machine learning* (ZHANG

et al., 2021), e que precisam ser resolvidos em ambiente críticos como os *data centers*.

A otimização multiobjetivo gera como resultado o conjunto solução que auxiliará o projetista (tomador de decisão) na construção da arquitetura adequada para cada demanda. Necessita-se, portanto, implementar estratégias baseadas em algoritmos de otimização evolucionários, para assim obter o conjunto de soluções aproximadas de Pareto e analisar seu desempenho com o conjunto solução ótimo.

Este trabalho análisa duas estratégias de otimização multiobjetivo que visam maximizar a disponibilidade e minimizar o consumo energético e os custos. Como se trata de um problema combinatório, onde cada escolha de equipamentos para montagem da arquitetura de *data centers* impacta nos resultados, é necessário se valer de uma estratégia meta-heurística que busque otimizar de forma integrada a disponibilidade, o custo e o consumo energético. A fim de testar a aplicabilidade das estratégias propostas, foram feitas comparações com uma técnica de otimização global que gera a curva de Pareto ótima. Esta técnica consiste no Algoritmo de força bruta, que enumera todos as possíveis soluções e checa cada indivíduo gerado dado um modelo de arquitetura elétrica de *data center*, podendo assim se chegar a uma solução ótima.

O aumento no número de equipamentos na base de dados e utilizando modelos de arquitetura elétrica de *data center* mais complexos faz com que o número de comparações cresça exponencialmente. Neste caso, o uso de um algoritmo de busca exaustiva não tem efetividade, sendo então necessário realizar análise dos resultados das estratégias meta-heurísticas propostas. Esses algoritmos foram codificados e integrados à ferramenta Stars (LEONARDO; CALLOU, 2021) em sua funcionalidade de otimização.

1.1 Motivação

Os projetistas de *data centers* necessitam de técnicas e ferramentas que possam auxiliá-los nas estimativas de custos de aquisição e operação dos equipamentos que compõem a infraestrutura elétrica. Há, então, o desafio de criar modelos e extrair deles métricas antes da implantação do sistema real. Por ser um ambiente de elevada demanda é imprescindível que haja alta disponibilidade. Outro critério que deve ter atenção é o consumo de energético requisitado pelos *data centers* que quadruplicaram nos últimos dez anos (WANG et al., 2021). Além disso, a diminuição do desperdício de energia elétrica

deve ser levada em consideração no momento do projeto da infraestrutura elétrica do *data center* considerando, por exemplo, a eficiência energética de seus equipamentos.

Esse trabalho teve foco em três métricas (funções objetivos) a serem otimizadas durante o projeto da infraestrutura elétrica de *data center*: disponibilidade, custo e eficiência energética. A disponibilidade foi escolhida devido ao forte avanço de novas tecnologias que demandam mais equipamentos, causando um aumento na complexidade das infraestruturas dos *data centers* que precisam permanecer em pleno funcionamento. Empresas que oferecem serviços de *data centers* devem fornecer um serviço que atenda a expectativa de desempenho que os clientes exigem, além de evitar que os serviços sejam interrompidos (VALENTIM; CALLOU, 2021).

Além de uma infraestrutura confiável que minimize as falhas, é necessário avaliar seus custos. Os custos são cruciais e dependem da arquitetura elétrica que será utilizada (CHALISE et al., 2015). As redundâncias existentes no projeto de um *data center* com alta disponibilidade precisam levar em conta os custos, isso gera gargalos na decisão final do projetista de qual tipo de equipamento utilizar. Por outro lado, o custo de indisponibilidade ou custo de *downtime* são os custos provenientes da perda da condição de operar o negócio quando os serviços não são fornecidos.

Portanto, os *data centers* necessitam de estratégias para assegurar o pleno funcionamento, a fim de evitar possíveis problemas de indisponibilidade, mas ao mesmo tempo reduzir os custos com projeto e alta eficiência energética. Desse modo, o presente trabalho tem as seguintes questões: existe alguma estratégia que otimize de forma simultânea a disponibilidade, custo e eficiência energética da infraestrutura elétrica de *data center*? O conjunto solução dessa estratégia gera o conjunto Pareto aproximados semelhante ao conjunto Pareto ótimo? A estratégia possui integração entre técnicas de modelagens formais (SPN, RBD e EFM)?

1.2 Objetivos

A presente pesquisa tem como objetivo a proposição de estratégias baseadas em algoritmos evolutivos, que otimizem o custo, a disponibilidade e exergia operacional de infraestruturas elétricas de *data centers*. Para atingir esse objetivo geral foram propostos os seguintes objetivos específicos:

- Propor modelos formais (RBD, EFM e SPN) na representação das infraestruturas elétricas de *data centers* com o objetivo de avaliar a disponibilidade, custo e exergia operacional;
- Aplicar modelos de redundâncias *hot standby* e *cold standby* com objetivo de modelar *data centers* tolerantes a falhas;
- Propor estratégias de otimização multiobjetivo baseadas em meta-heurísticas evolutivas que minimizem custo e desperdício energético e maximize a disponibilidade;
- Realizar análise comparativa entre o conjunto solução das estratégias propostas (curva de Pareto aproximado) e o algoritmo de força bruta que consegue avaliar todos os cenários possíveis (fronteira de Pareto ótima), para validar as estratégias.
- Avaliar comportamentos das estratégias propostas para modelos de arquiteturas mais complexas.

1.3 Estrutura do Trabalho

Além da introdução, esta dissertação está organizada em mais seis capítulos. No Capítulo 2 são introduzidos os conceitos fundamentais sobre disponibilidade, custo, exergia, redes de Petri, diagramas de bloco de confiabilidade, modelo de fluxo de energia, subsistema elétrico de *data center* e otimização multiobjetivo. No Capítulo 3 são apresentados alguns dos principais trabalhos realizados com temas pertinentes a pesquisa. No Capítulo 4 é explicada a metodologia usada para no desenvolvimento das duas estratégias de otimização propostas. As estratégias propostas são apresentada no Capítulo 5, nele são apresentados Algoritmos baseados nas duas técnicas de otimização multiobjetivo (NSGA II e MOPSO), e outros Algoritmos acessórios, critério de parada adotado e a validação dos resultados. O Capítulo 6 é dedicado aos estudos de caso realizados, bem como as análises estatísticas dos resultados obtidos. No Capítulo 7 são apresentadas a conclusão a respeito do trabalho, considerações e possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta alguns conceitos básicos para um melhor entendimento da pesquisa desenvolvida. Inicialmente, define-se *data center* e as suas infraestruturas. Em seguida são apresentados conceitos sobre disponibilidade, exergia e custo. Logo após é a vez dos formalismos adotados neste trabalho: redes de Petri estocásticas, diagramas de bloco de confiabilidade e *energy model flow*. Por fim, são apresentados os métodos de otimização multiobjetivo.

2.1 *Data center*

Um *data center* é o ambiente com alto desempenho e grande capacidade de armazenamento que fornece suporte para aplicativos de negócios corporativos e que é composto por equipamentos fundamentais pelo armazenamento e organização de dados (DAYARATHNA et al., 2016). São ambientes que possuem um papel crítico que necessitam de sistemas de distribuição de energia, sistemas de resfriamento, servidores web e de aplicativos, servidores de arquivo e impressão.

Estes ambientes necessitam que estejam em pleno funcionamento e tenha configurações que sejam tolerantes a falhas. Por isso, quando se trata de *data center* está intrinsecamente ligada a ideia de ter elevado índice de disponibilidade. Numa abordagem de alto nível pode-se dizer que a obtenção desta alta disponibilidade passa pela composição de suas infraestruturas. (FERREIRA et al., 2020) diz que o *data center* deve ter: (i) rede redundante para manter o sistema funcionando se um dispositivo de rede falhar, e (ii) múltiplas fontes de energia elétrica, responsáveis pelo suporte da carga elétrica do *data center*.

A Figura 1 apresenta as infraestruturas básicas para implementação de um *data center*. As infraestruturas possuem um sistema de alimentação energética, um conjunto formado por equipamentos de resfriamento do ambiente e o de Tecnologia da informação onde estão, por exemplo os servidores. Cada uma layer é composta por equipamentos próprios.

A infraestrutura de um *data center* geralmente, garante que os componentes de TI funcionem de forma confiável e inclui: (i) infraestrutura de TI; (ii) infraestrutura de

resfriamento; e (iii) infraestrutura de energia (UDDIN et al., 2014). A infraestrutura de resfriamento é aquela que atende os requisitos de climatização dos sistemas informatizados nela abrigados, faz-se necessário que o sistema seja de precisão. Ela é composta por uma variedade de equipamentos, como: compressores mecânicos (condicionadores de ar), resfriadores e torres de resfriamento.

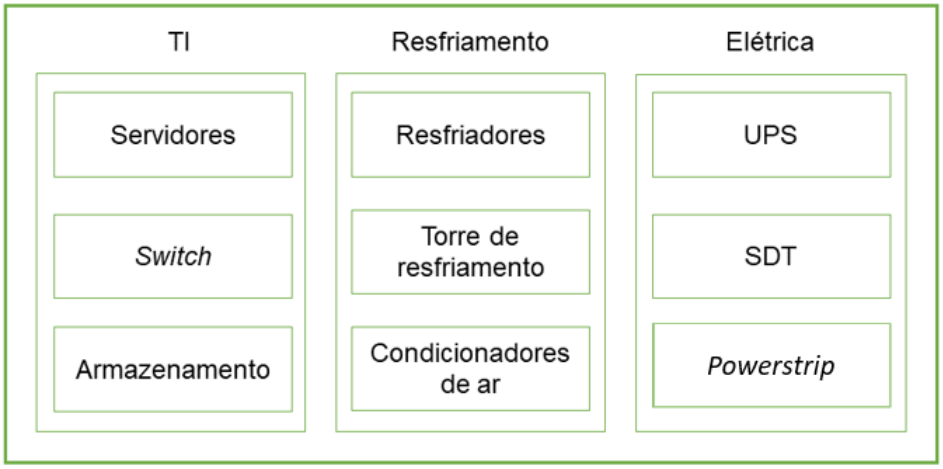


Figura 1 – Infraestruturas básicas do *data center*

A infraestrutura elétrica de um *data center* é aquela que provê às demais infraestruturas a alimentação energética adequada, composta basicamente por fontes de alimentação sem interrupção (UPS), um transformador (SDT), nos casos de redundância usa-se um chaveador elétrico manual (STS), um painel elétrico (*subpanel*), réguas de tomadas elétricas (*power strip*). A infraestrutura de TI que fornece os serviços de *software* implantados nos servidores e de redes que incluem os *hardwares* tais como *switches*, roteadores e armazenamento.

Este trabalho foca na infraestrutura elétrica considerando os atributos de eficiência energética, o tempo médio de falha (MTTF), o tempo médio de reparo (MTTR) de cada um deles e custo de aquisição dos equipamentos que compõe esta infraestrutura.

2.2 Disponibilidade

A disponibilidade é a probabilidade de um sistema funcionar corretamente em um momento específico ou durante um período de tempo predefinido (NGUYEN et al., 2019). Trata-se, então, da capacidade que o sistema tem de permanecer ativo e eficaz por um determinado intervalo de tempo.

A disponibilidade de um sistema pode ser computada a partir dos tempos de falhas (MTTF - *Mean Time To Failure*) e reparo (MTTR- *Mean Time To Repair*) do sistema conforme mostrado na Equação . O resultado obtido pelo cálculo realizado na Equação 2.1:

$$D = \frac{MTTF}{MTTF + MTTR} \tag{2.1}$$

Por se tratar de um ambiente que necessita de disponibilidade acima de 99%, por exemplo 99,999%, é preciso utilizar uma forma de melhor organizar o resultado da disponibilidade. Por isso, alguns trabalhos assim como (ROSENDO et al., 2019) trabalham com a disponibilidade em número de noves. A Equação 2.2, onde D representa a disponibilidade calculada em 2.1. Ela quantifica os algarismos consecutivos iguais a 9, que estão após a vírgula o que organiza a apresentação do resultado.

$$D_{9's} = -log_{10}(1 - D) \tag{2.2}$$

2.3 Exergia

A exergia é uma métrica que pode ser utilizada para quantificar a energia desperdiçada, ou seja, energia perdida pelo equipamento que não conseguiu convertê-la em trabalho útil (BEZERRA et al., 2020). O seu conceito vem da segunda lei da termodinâmica que diz que todo trabalho realizado por uma máquina possui uma certa perda (ROSEN, 2007). Uma das funções objetivo a ser otimizada neste trabalho é a exergia, pois quanto menor o valor desta métrica, maior será a eficiência da arquitetura avaliada.

2.4 Modelo de Fluxo de Energia

O EFM é um modelo proposto por (CALLOU, 2013) para representar o fluxo de energia elétrica ou carga térmica entre os equipamentos do sistema. Esta modelagem leva em consideração a eficiência e a capacidade máxima de carga energética que cada componente pode suportar, como também de fornecer.

Na Figura 2-a é possível observar o EFM de um subsistema de uma infraestrutura

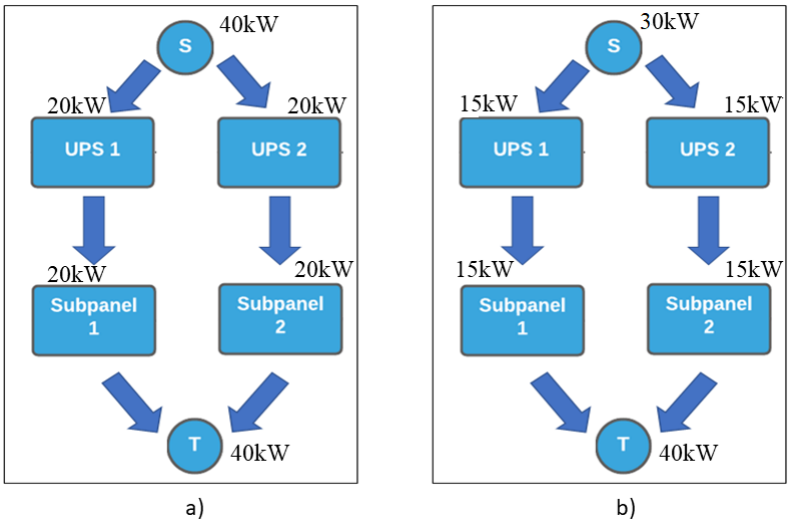


Figura 2 – EFM. a) Fluxo de energia realizado com sucesso; b) Fluxo de energia falho.

elétrica de um *data center*. Neste exemplo, a potência máxima suportada pelos equipamentos UPS e *subpanel* é de 20kW e a energia demandada por outros subsistemas (equipamentos de *cooling* e/ou TI) é de 40kW. O valor demandado, no modelo EFM, é associado ao nó *target* – T. No EFM, S representa a entrada da carga energética no sistema. No exemplo será considerado que os UPSs e *subpanels* possuem as mesmas configurações para cada modelo.

Mantendo os mesmos equipamentos, o cenário b difere apenas na energia de entrada do sistema (30kW). Dessa forma, é possível perceber que na Figura 2-b que cada UPS fornece potência de 15 kW para os *subpanels*, porém haverá o comprometimento do pleno funcionamento de outros subsistemas que dependem da alimentação energética, uma vez que o subsistema energético não será capaz de atender a potência exigida em T (40kW).

Conforme explicado anteriormente (Seção 2.3), a exergia operacional pode ser utilizada para avaliar a eficiência energética dos equipamentos que compõem o sistema. Como uma infraestrutura de *data center* é composta por diversos equipamentos elétricos, para se ter o cálculo da exergia operacional (Ex_{op}) é necessário o somatório das energias dissipadas por cada um deles durante um intervalo (T) e no efetivo funcionamento, calculado através da disponibilidade (D) da arquitetura avaliada. A Equação 2.3 representa a exergia operacional do sistema, onde i representa cada equipamento e n a quantidade.

$$Ex_{op} = \sum_{i=1}^{\mathrm{n}} Ex_{op_i} \times T \times D \tag{2.3}$$

O custo é um dos objetivos a ser otimizado neste trabalho. O modelo EFM, além de

considerar a exergia, avalia também a métrica do custo (C). Este custo leva em conta dois outros custos. O primeiro é o custo de aquisição (CA), obtido por pesquisa de mercado, e o outro é o custo operacional (CO), computado pelo produto da energia elétrica consumida pelo sistema ($E_{consumida}$), T o período assumido; E_{custo} corresponde ao preço da energia (US$/kWh) e D representa a disponibilidade do sistema, conforme mostrado na Equação 2.4

$$CO = E_{consumida} \times T \times E_{custo} \times D \tag{2.4}$$

2.5 Redes de Petri Estocásticas

A rede de Petri é uma ferramenta para modelar e analisar sistemas de eventos discretos (REISIG, 1985). O tempo e as escolhas probabilísticas são aspectos essenciais para um modelo de avaliação de desempenho. As redes de Petri estocástica (Stochastic Petri Nets - SPN) são uma extensão de rede de Petri onde as transições podem disparar após um atraso probabilístico determinado por uma variável aleatória. A definição formal das SPNs pode ser encontrada em (MARSAN, 1990). As SPNs representam uma técnica de análise e modelagem amplamente difundida, útil para avaliar métricas como, por exemplo, a disponibilidade de um sistema.

Os elementos básicos que compõem uma SPN são: arco, lugar, *token* (ou ficha) e transição. O arco pode ser padrão ou inibidor, já a transição pode ser do tipo estocástica (temporizada) ou imediata. Na Figura 3 é possível visualizar os principais elementos das SPNs

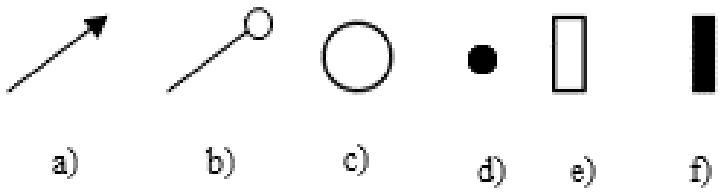


Figura 3 – Elementos das redes de Petri Estocásticas. a) arco; b) arco inibidor; c) lugar; d) ficha; e) transição estocástica; f) transição imediata

Modelar estados de um sistema é um dos objetivos de se trabalhar com SPN.

A Figura 4 mostra um exemplo em SPN que possui dois lugares para representar os estados de um equipamento (UPS). O lugar UPS_ON representa o estado em pleno funcionando, o outro lugar (UPS_OFF) representa o estado quebrado. A disponibilidade pode ser computada pela probabilidade de se ter um *token* no lugar UPS_ON, ou seja, $D = P\{\#UPS\_ON = 1\}$.

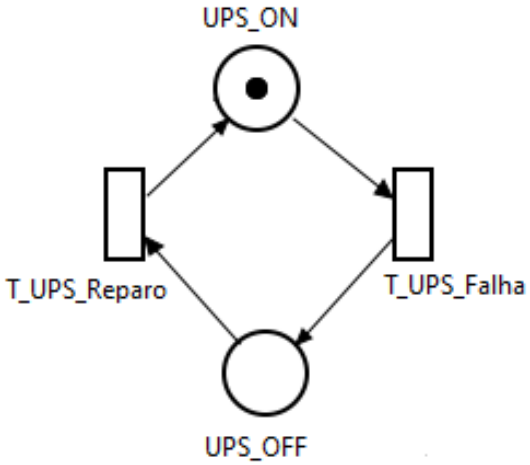


Figura 4 – Modelo SPN do UPS

2.6 Diagramas de Bloco de Confiabilidade

O RBD é um método de modelagem utilizado para se computar a disponibilidade do sistema em análise (WANG et al., 2004). Este diagrama pode incluir componentes conectados em série ou em paralelo, segundo é apresentado na Figura 5. Quanto a composição em série, se houver a falha de um equipamento, o sistema para de funcionar. Já na disposição em paralelo, o sistema para de funcionar, se todos os equipamentos falharem.

Cada bloco representa um componente e tem associado os valores médios dos tempos de falha e reparo. Em um RBD, a entrada é dada pela extremidade da direita, e a saída é observada na outra extremidade. O sistema está funcional, se há pelo menos um caminho de componentes funcionando desde a entrada até a saída. Caso contrário, o sistema se encontra no estado de falha (LOMAN; WANG, 2002).

A Figura 5(a) mostra a representação do RBD em série. O cálculo da disponibilidade do RBD em série (D_s) com n blocos é dado pela Equação 2.5:

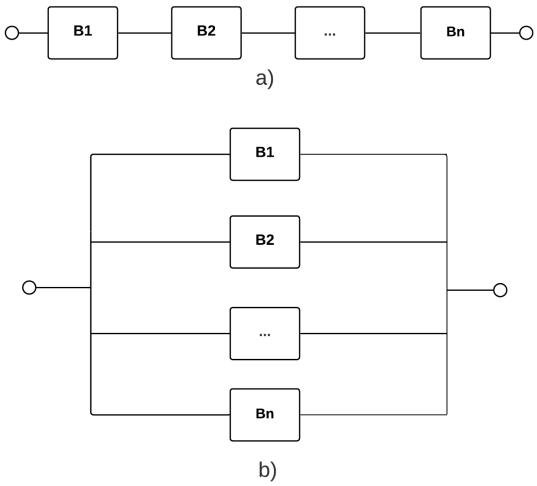


Figura 5 – Estrutura básica do RBD. a) Série. b) Paralelo.

$$D_s = \prod_{i=1}^{n} D_i \tag{2.5}$$

onde D_i é a disponibilidade do bloco i.

Para uma disposição paralela Figura 5(b), o sistema funciona se pelo menos um componente funcionar. Assumindo um sistema com n componentes independentes a disponibilidade é obtida por:

$$D_p = 1 - \prod_{i=1}^{n} (1 - D_i) \tag{2.6}$$

onde D_p é a disponibilidade do bloco i.

2.7 Otimização Multiobjetivo

Muitos problemas encontrados na engenharia seriam elaborados de maneira mais adequada se descritos em termos de múltiplos objetivos a serem otimizados. Os sistemas elétricos de potência são exemplos de campo que apresenta amplo espectro de aplicações de algoritmos de otimização multiobjetivo. Especificamente, em planejamento de redes de distribuição de energia elétrica, tal como abordado nesta dissertação, normalmente é importante minimizar o custo de instalação e operação do sistema ao mesmo tempo em que também se faz necessário otimizar índices de desempenho diversos, como limites de desvio de tensão e distorção harmônica. Por outro lado, a melhora no desempenho elétrico costuma

estar associada a custos maiores, ou seja, é necessário buscar um compromisso entre custo e desempenho, objetivos conflitantes que devem ser otimizados simultaneamente.

Entre as principais técnicas para resolver problemas de otimização multiobjetivo, podemos citar os algoritmos evolucionários. De maneira geral, os algoritmos evolucionários se caracterizam por utilizar uma população de indivíduos que evoluem em paralelo, objetivando chegar o mais próximo possível das melhores soluções.

De acordo com (VELDHUIZEN; LAMONT, 2000), a maior parte dos problemas reais encontrados na área de otimização envolve a obtenção de diversas metas que devem ser atingidas simultaneamente. Elas geralmente são conflitantes, ou seja, não existe uma solução única que otimize todas ao mesmo tempo. Para tal classe de problemas devemos encontrar um conjunto de soluções diferentes.

Problemas dessa natureza são chamados de problemas de otimização multiobjetivo por envolverem minimização (ou maximização) simultânea de um conjunto de objetivos satisfazendo a um conjunto de restrições. Neste caso, a tomada de decisão será de responsabilidade do analista, que deverá ponderar os objetivos globais do problema e escolher uma entre as soluções do conjunto de soluções eficientes (ARROYO; ARMENTANO, 2005).

Este é sem dúvida um importante tópico da otimização, tanto para pesquisadores quanto para engenheiros, não só por causa de sua aplicabilidade nos problemas reais, mas também pelas questões ainda em aberto nessa área (COELLO, 2000).

2.7.1 Conceitos Básicos

Para (KONAK et al., 2006) a otimização multiobjetivo pode ser definida como o problema de achar um vetor de variáveis de decisão cujos elementos representam as funções objetivos. Essas funções formam uma descrição matemática do critério de otimalidades que estão em conflito umas com as outras. Neste caso, o termo "otimizar" significa encontrar um conjunto de soluções que não podem ser melhoradas simultaneamente para o analista. De acordo com (CHADLI, 2007), um problema de otimização multiobjetivo pode ser formulado como :

$$(min)\ z = f(x) = (f_1(x), f_2(x), \ldots, f_{OB}(x)) \tag{2.7}$$

$$g(x) = (g_1(x), g_2(x), \ldots, g_r(x)) \leq b \tag{2.8}$$

$$x = (x_1, x_2, x_3, \ldots, x_n) \in X \tag{2.9}$$

$$z = (z_1, z_2, z_3, \ldots, z_r) \in Z \tag{2.10}$$

na qual x é o vetor de decisão, OB o número de objetivos, z é o vetor objetivo, X denota o espaço de busca de decisões, e $z = f(x)$ é a imagem de X, denominada espaço objetivo.

O conjunto de restrições $g(x) \leq b \mid b \in \mathbb{R}^+$ e o espaço X determinam o conjunto das soluções viáveis ou factíveis: $X^* = \{x \in X \mid g(x) < b\}$.

Portanto, o problema multiobjetivo pode ser escrito da forma:

$$(min)\ z = (f_1(x), f_2(x), \ldots, f_r(x)) \tag{2.11}$$

$$x \in X^* \tag{2.12}$$

A imagem de X^* é denominada espaço objetivo factível e é denotada por $Z^* = f(X^*) = \{f(x) \mid x \in X^*\}$.

Enquanto que na otimização mono-objetivo uma solução ótima é claramente identificada, pois o espaço de solução é ordenado, na otimização multiobjetivo, por outro lado, há um conjunto de alternativas, geralmente conhecidas como soluções Pareto-ótimas, também podem ser denominadas como soluções eficientes, ou conjunto admissível do problema.

Em (MISHRA et al., 2002) o conceito de Pareto-ótimo constitui a origem da busca na otimização multiobjetivo. Pela definição, um vetor z é Pareto-ótimo se não existe um outro vetor viável z que possa melhorar algum objetivo, sem causar uma piora em pelo menos um outro objetivo. Em outras palavras, um vetor solução z pertence ao conjunto de soluções Pareto-ótimo se não existe nenhum vetor solução z^* que domine z.

Considerando um problema de minimização, temos:

- z domina z^*, ou simplesmente, $z \prec z^*$ se, e somente se, $z_j \leq z_j^*$ para todo j e, para pelo menos um j, $z_j < z_j^*$;

- z e z^* s possuem o mesmo grau de dominância se, e somente se, z $\nprec$ z* e $z^* \nprec z$.

Neste último caso, as soluções não podem ser identificadas como melhor que as outras, a menos que informações de preferência em relação aos objetivos sejam incluídas. Para (EHRGOTT et al., 2014), a otimização multiobjetivo diverge da otimização mono-objetivo, devido ao fato de raramente admitir uma simples solução. Por isso, a solução é composta por uma família de soluções pareto-eficientes (ou pareto-ótimas) que devem ser considerados equivalentes, em vista da ausência de informação referente à importância de cada objetivo.

A Figura 6 mostra o quadrante de análise para a formação do conjunto. O conjunto de Pareto é formado nos quadrantes sinalizados como indiferentes ao se analisar a solução C.

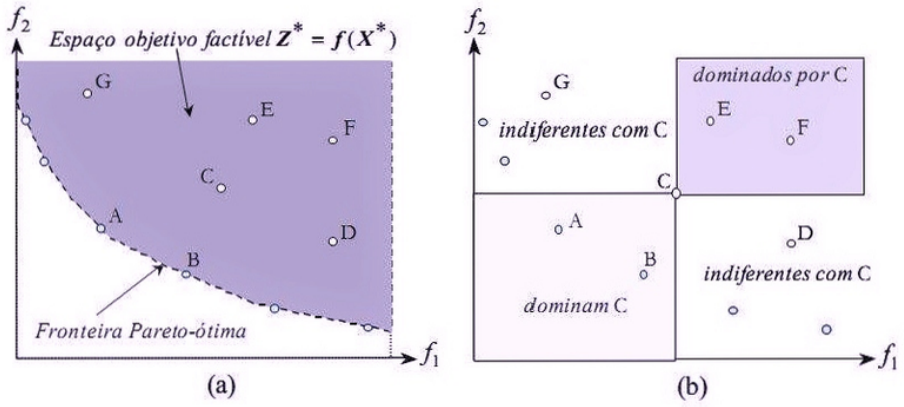


Figura 6 – ilustração do conceito de dominância de Pareto para minimização das duas funções objetivos

Segundo (GIAGKIOZIS; FLEMING, 2014), à medida que se aumenta o número de objetivos a serem otimizados, e também, ponderados os objetivos menos "bem comportados", o problema de buscar uma solução pareto-ótima torna-se complexo de forma rápida e progressiva. Em alguns casos, as funções objetivos poderiam ser otimizadas separadamente, uma a uma buscando um ganho no tempo computacional. Porém, soluções satisfatórias para o problema global raramente poderiam ser achadas neste sentido. Ou seja, uma solução ótima em um objetivo, se tal ótimo existe, comumente implica desempenho inaceitável em um ou mais dos outros objetivos, resultando na necessidade de se fazer um ajuste.

Os métodos de otimização convencional, como os baseados no *simplex*, adaptados aos problemas multiobjetivo apresentam restrições e tornam-se ineficientes (SANCHES

et al., 2012). Neste caso, para esta gama de problemas, é necessário se valer de métodos meta-heurísticos de otimização.

2.7.2 Métodos de Otimização Multiobjetivo

A solução dos problemas de otimização multiobjetivo consiste em determinar, no espaço objetivo factível, o conjunto eficiente, um subconjunto do conjunto eficiente ou ainda, conjuntos de soluções próximas da fronteira Pareto ótimo. O tamanho e a complexidade dos métodos de solução encontrados em grande parte dos problemas práticos exige a intervenção de um tomador de decisão. A definição dos critérios de busca pode ocorrer antes da execução da mesma, combinando os objetivos do problema em um único objetivo, segundo uma determinação de pesos de preferência do tomador de decisão (FONSECA et al., 2017). Como consequência, o problema acaba se constituindo na otimização de um único objetivo, que requer estratégias clássicas de otimização direta e exata.

Outra opção do tomador de decisão é classificar os objetivos em ordem de prioridade, efetuando a busca da solução ótima em etapas, iniciando a otimização do primeiro objetivo, sem considerar as demais, e seguindo para a otimização dos objetivos seguintes, considerando o valor ótimo anterior, até atingir o último objetivo. Nota-se que esta última estratégia não garante a obtenção da solução eficiente. Há também a classe de métodos de solução que envolve o mapeamento das soluções factíveis através de algoritmos evolucionários, que ao terminar as iterações, são disponibilizados ao tomador de decisão um conjunto de soluções aproximadas ou Pareto ótimo.

2.7.2.1 Otimização Combinatória Multiobjetivo

Em geral, ou o espaço objetivo compõe um conjunto finito ou, na prática, é possível limitar a busca dentro de um conjunto finito. Neste trabalho, são adotadas três dimensões do espaço de decisões, sendo que cada um corresponde a métrica que se deseja otimizar (disponibilidade, custo e exergia operacional), já quanto ao espaço objetivo são, teoricamente, finitos. O problema de Otimização Combinatória Multiobjetivo consiste em problemas que maximizam ou minimizam as funções objetivo envolvendo várias variáveis de decisão sujeito a dois tipos de restrições, as restrições de igualdades/desigualdades

e as restrições de integralidade sobre algumas ou todas as variáveis (TEGHEM, 2009). Resolve-se o problema combinatório como um método a posteriori, que consiste em realizar a busca antes da tomada de decisão. Com isto, encontra-se o conjunto das melhores soluções (fronteira de Pareto ótimo), e posteriormente o tomador de decisões escolhe a solução mais satisfatória dependendo de sua experiência, das necessidades e recursos disponíveis.

Portanto, existe uma problemática na Otimização Combinatória Multiobjetivo, trata-se do custo computacional para chegar no conjunto de soluções da fronteira Pareto ótimo, pois ele explora todo o espaço de decisão. A intensificação da busca deve ser devidamente avaliada pelo tomador de decisão, de maneira a lidar com o compromisso de aplicar este método em condições factíveis. Constituem problemas de Otimização Combinatória Multiobjetivo algumas variedades de planejamento espacial posicionamento de satélites, roteamento de veículos, alocação de trabalhadores, projetos de sistemas elétricos de *data center*, etc.

2.7.2.2 Algoritmo Genético de Ordenação das Soluções Não-Dominadas

Desde que surgiram, em 1985, os métodos de otimização Multiobjetivo (MO) baseados no Algoritmo Genético (AG) têm sido utilizados na resolução dos mais variados tipos de problema, os quais perpassam por áreas diversas como economia, meteorologia, saúde, arquitetura e engenharias, por exemplo. O primeiro deles, proposto por (SCHAFFER, 1985), denominado VEGA (Vector Evaluated Genetic Algorithm) é apenas uma extensão do AG convencional para problemas MO. Depois disso, muitas outras abordagens para o tratamento de problemas MO foram propostas. O Algoritmo Genético de Ordenação das Soluções Não-Dominadas (do inglês: *Non-dominated Sorting Genetic Algorithm* II - NSGA-II) proposto por (DEB et al., 2002) é uma dessas abordagens, cuja característica principal é a ordenação dos indivíduos por não-dominância. O fluxograma do algoritmo está ilustrado na Figura 7.

O algoritmo inicia com uma população aleatória de indivíduos. Em seguida são avaliadas as funções objetivos de cada um destes indivíduos. O terceiro passo é o ordenamento por não-dominância que consiste em utilizar o conceito de dominância para ranquear os indivíduos, formando os ranques. Assim, o ranque 1 é composto por todos

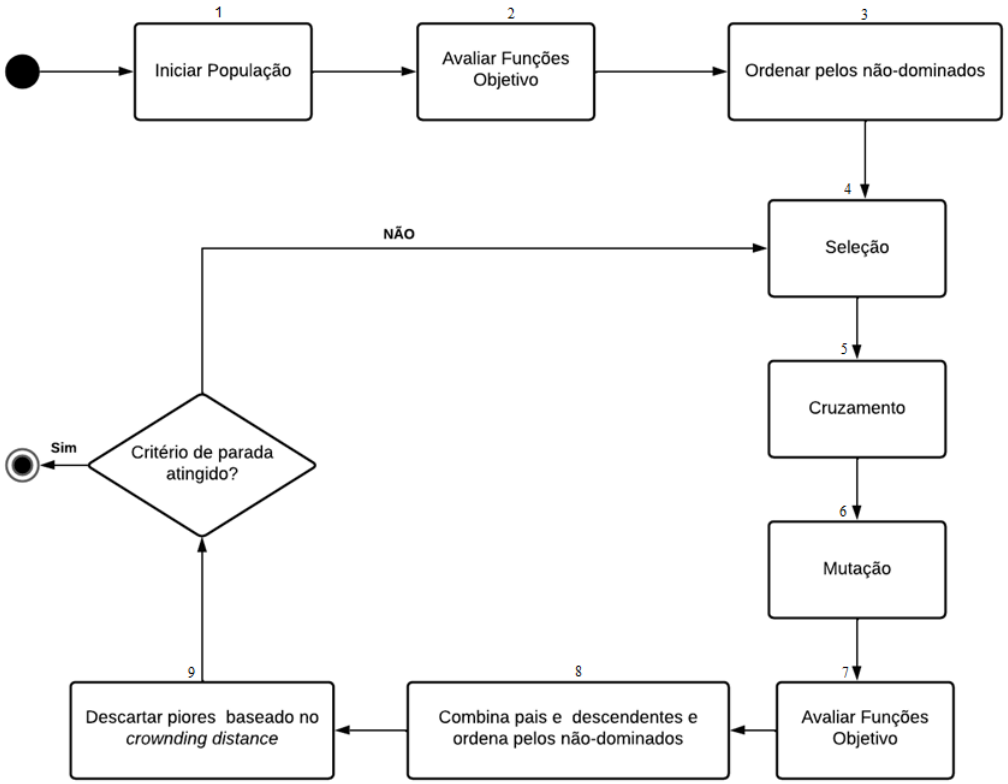


Figura 7 – Esquema do NSGA II

os indivíduos não-dominados dentro da população. O ranque 2, por sua vez, integram os indivíduos que são dominados apenas pelos indivíduos do ranque 1 e que dominam todos os n indivíduos dos ranques consecutivos. Os indivíduos pertencentes aos demais ranques são organizados seguindo a lógica tomada anteriormente.

Na quarta etapa do algoritmo NSGA-II são geradas duas populações, denotadas como pais e descendentes, ambas de tamanho N. Isto é feito após aplicar os operadores de genéticos de seleção dos melhores pais (mais bem ranqueados), em seguida é feito o cruzamento deste pais e, caso determinado percentual for atingido em um método aleatório ocorrerá a mutação. O resultado é uma população com pais e filhos. Na sétima etapa ocorre a avaliação das funções objetivos, na oitava o reordenamento por soluções não dominadas é feito. O preenchimento dos indivíduos que farão parte da próxima geração é feito na etapa nove. Nela as soluções não-dominadas pertencentes ao primeiro ranque (R1) da população combinada são as melhores entre todas as soluções em R, e são encaminhadas para a próxima geração (elitismo). Se o tamanho de R1 for menor que N, os membros remanescentes da próxima geração são preenchidos pelo conjunto de solução da segunda fronteira e, assim, sucessivamente até que se tem exatamente N soluções na

próxima geração. Caso o critério de parada tenha sido atingido se finaliza o algoritmo, caso contrário o processo é repetido a partir da quarta etapa.

De forma a garantir a diversidade dos indivíduos não-dominados, o NSGA-II emprega o cálculo da distância de multidão (ou *crowding distance*). Trata-se da média da distância das soluções adjacentes a um dado indivíduo para todos os objetivos considerados (GONG et al., 2015). A Figura 8 revela graficamente como essas distâncias são medidas. A vantagem do cálculo da distância de multidão diante de outros métodos que o antecederam reside no fato de não ser necessário a determinação de outros parâmetros além dos valores dos objetivos para cada indivíduo.

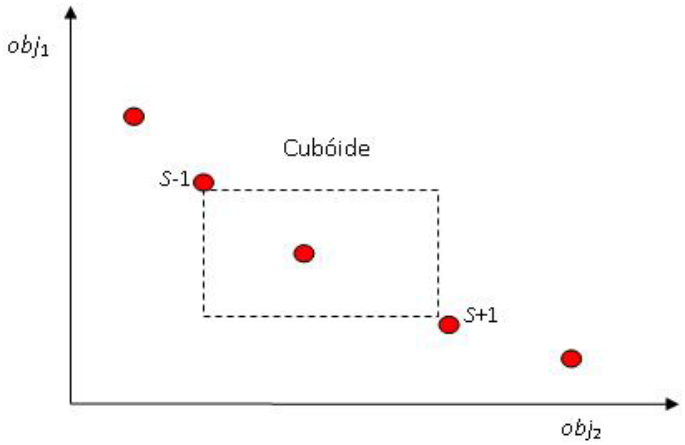


Figura 8 – Ilustração do cálculo da distância de multidão

Uma forma de visualizar a aplicação do *crowding distance* dentro do funcionamento do algoritmo NSGA-II é pelo esquema da Figura 9. Após o ranqueamento por soluções não dominadas, a População (pais mais descendentes), agora ordenada em ranques, precisa decidir quem passa para próxima geração. O primeiro critério leva em consideração os melhores indivíduos, aqueles que pertencem ao ranque 1. Avalia-se se a quantidade desses indivíduos é maior que N. Sendo maior, procede-se com o método *crowding distance* para selecionar aqueles que passarão à próxima geração. Realiza-se o cálculo da distância de multidão, escolhendo os indivíduos com os maiores resultados nesse parâmetro até completar o número N da população da Próxima Geração. Esse mecanismo favorece as primeiras fronteiras, estabelecendo um caráter elitista ao NSGA-II.

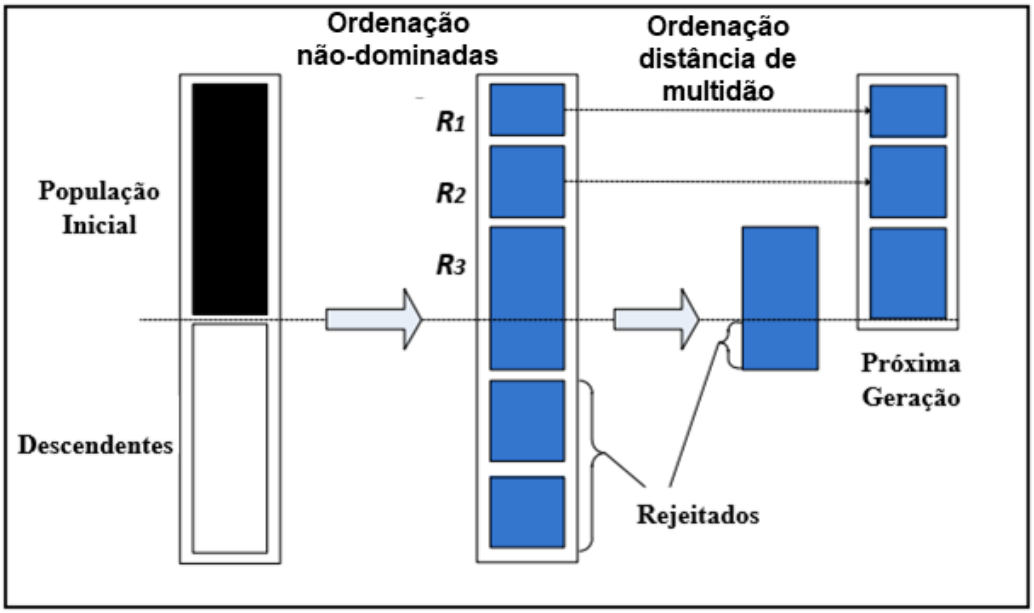


Figura 9 – Esquema NSGA II

2.7.2.3 Otimização Multiobjetivo por Enxame de Partículas

A otimização multiobjetivo por enxame de partículas (do inglês: *multi-objective particle swarm optimization* - MOPSO) proposta por (PATIL; DANGEWAR, 2015) é um algoritmo que usa o conceito de dominância de Pareto para encontrar soluções para problemas multiobjetivos. Também emprega a ideia da população ser guiada a partir das melhores posições dos indivíduos dentro do espaço de busca, armazenando as melhores soluções e orientando a busca em gerações futuras. O PSO (Otimização por Enxame de Partículas) proposto por (KENNEDY; EBERHART, 1995) é o algoritmo que o MOPSO se baseia. Nele a busca por soluções ocorre colocando um número de individuos (partículas) no espaço de busca do problema, onde cada elemento avalia a função de aptidão para sua posição atual. Cada partícula então calcula seu próximo movimento pelo espaço de busca combinando informações sobre sua atual e melhor solução encontrada até então, com informações oferecidas por outras partículas, além da aplicação de algumas perturbações aleatórias, tal qual a mutação no algoritmo NSGA-II. A próxima iteração tem início após todas as partículas terem sido movidas. Eventualmente, toda a população acaba se comportando como uma revoada de pássaros coletivamente procurando por comida, movendo-se próximos a uma área ótima da função de aptidão (*fitness*).

Muitos problemas encontrados na engenharia seriam elaborados de maneira mais adequada se descritos em termos de múltiplos objetivos a serem otimizados. Os sistemas

elétricos de potência são exemplos de campo que apresenta amplo espectro de aplicações de algoritmos de otimização multiobjetivo. Especificamente, em planejamento de sistema elétrico de *data center*, tal como abordado nesta dissertação, normalmente é importante minimizar o custo de operação do sistema ao mesmo tempo em que também se faz necessário otimizar a disponibilidade. Por outro lado, a melhora na disponibilidade costuma estar associada a custos e exergia operacional maiores, ou seja, objetivos conflitantes que devem ser otimizados simultaneamente.

Para se encontrar soluções que mitiguem os problemas de otimização multiobjetivo, é necessário aplicação de meta-heurísticas, tais como estratégias baseadas no MOPSO. A Figura 10 apresenta um fluxograma básico do funcionamento do algoritmo.

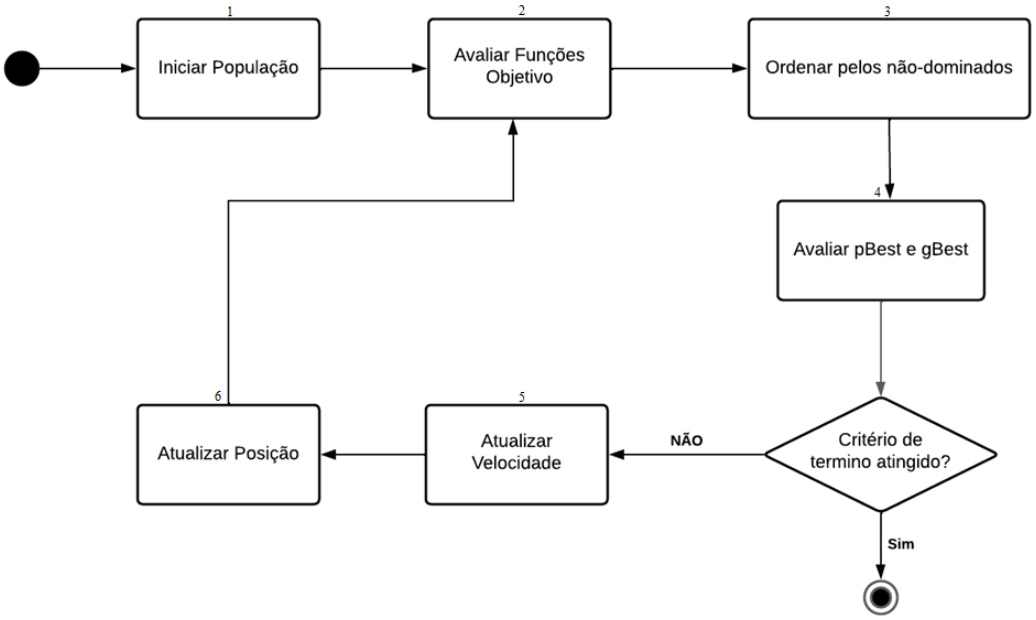


Figura 10 – Esquema do MOPSO

Na primeira etapa é inicializada a população de modo aleatório dentro do espaço de busca. Cada partícula (indivíduo) possui três atributos de N-dimensões, onde N é o número de dimensões do espaço de busca. Os atributos são: o vetor x_i representa a posição atual, p_i a última melhor posição, e v_i a velocidade. A segunda etapa do Algoritmo consiste na avaliação de cada partícula, aqui as métricas são calculadas e atribuídas para cada coordenada da posição, visto que a atual posição x_i é considerada como um conjunto de coordenadas que descrevem um ponto no espaço. Seguindo para etapa três as partículas são ordenadas utilizando o critério de soluções não dominadas. Em seguida, na etapa

seguinte, cada partícula comunica-se com outras partículas, e é afetada pelo melhor ponto encontrado pBest por qualquer membro da sua vizinhança. Já o gBest, que armazena a melhor posição para todas as partículas, é usado para fazer com que todas as partículas apontem para o máximo/mínimo global.

Enquanto o critério de parada não é atingido, duas fórmulas são usadas. Uma na quinta etapa do fluxograma, para atualizar a velocidade (ver Equação 2.13) e outra, na sexta etapa, para atualizar a posição da partícula no espaço de busca (ver Equação 2.14).

$$v(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest(t) - x_i(t)) + c_2 \cdot r_2 \cdot (gBest(t) - x_i(t)) \tag{2.13}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.14}$$

A velocidade adicionada à cada partícula durante o processo de evolução é ajustada para que cada partícula estocasticamente oscile dentre as regiões de pBest e gBest. O MOPSO possui alguns operadores básicos que são utilizados durante seu processo evolutivo. O primeiro deles é chamado o peso da inércia (w) para realizar a atualização da velocidade das partículas no espaço de busca. O fator de c_1 é uma constante denominada o peso cognitivo no intervalo [0, 2). O fator de r_1 é uma variável aleatória no intervalo [0, 1), que é maior ou igual a 0 e estritamente menor que 1. O pBest é a melhor posição da partícula encontrada até o momento. O $x_i(t)$ é a posição atual da partícula. O fator de c_2 é uma constante denominada o social no intervalo [0, 2). O fator de r_2 é uma variável aleatória no intervalo [0, 1).

Por fim, em cada iteração do algoritmo, a posição atual é avaliada como uma solução do problema. Caso essa posição seja a melhor que qualquer outra encontrada até o momento para aquela partícula, suas coordenadas são salvas no segundo vetor, pBest, que é usado para cálculo da velocidade. O objetivo é iterar este processo e encontrar melhores soluções, atualizando os valores de p_i. Assim sendo, cada partícula percorrerá o espaço de busca a uma determinada velocidade v_i atingindo a cada iteração (t) uma posição x_i.

2.8 Ferramenta *Stars*

A ferramenta *Stars* (Stochastic Tool for Availability and Reliability System analysis) proposta por (LEONARDO; CALLOU, 2021) surgiu com o objetivo de conectar uma

visão de alto nível representando a arquitetura de *data centers*, com os modelos formais responsáveis por avaliar as métricas como disponibilidade, custo e exergia operacional. A ferramenta foi desenvolvida em Java, utilizando as bibliotecas JAVA(Swing) e JGraphX. A Figura 11 fornece uma visão geral de como a ferramenta funciona e como ela se integra a outras ferramentas computacionais já existentes. Ressalta-se que a avaliação dos modelos é feita a partir de chamadas de métodos *solve* e *metric* nativas do ferramental Mercury (OLIVEIRA et al., 2017). Pode-se observar que o *Stars* foi feito em camadas, com as classes da visão de alto nível, o analisador (parser) e os módulos de otimização e de manutenção. Os algoritmos apresentados nesta dissertação foram codificados e integrados ao módulo de otimização do *Stars*.

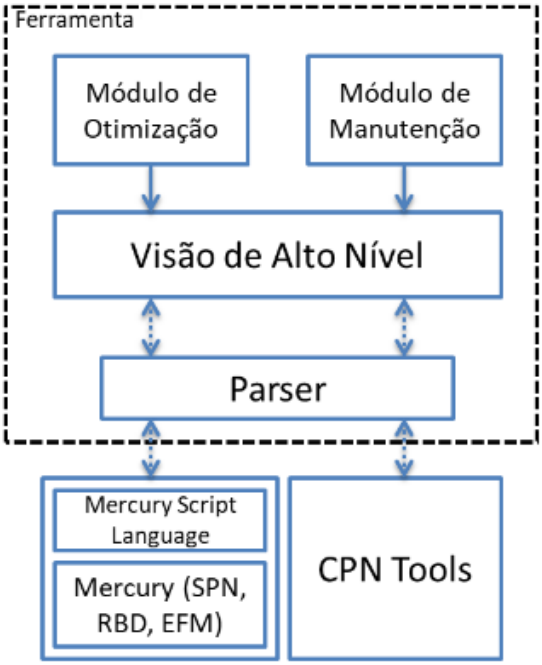


Figura 11 – Camadas do ferramental Stars

A camada do Parser é responsável pela conversão do modelo gerado pela ferramenta proposta para os modelos formais. Essa conversão é feita com base nas métricas que se deseja avaliar. Por exemplo, se um projetista deseja avaliar a disponibilidade de um modelo de sistema elétrico de *data center*, o Parser converte este modelo para os formalismos RBD ou SPN. Essa conversão é feita através da representação na linguagem de *script* do Mercury como foi explicado na Subseção 4.3. Essa linguagem de *script* permite a avaliação dos modelos SPN, RBD e EFM sem a necessidade de se chamar a interface gráfica. Um módulo de otimização também foi desenvolvido para poder realizar a otimização dos

modelos criados. Existe ainda o módulo de manutenção, criado para a partir dos modelos da visão de alto nível (MELO F., 2022). poder realizar ajustes e converter pelo Parser para modelos em SPN de manutenção preditiva e corretiva, por exemplo.

É importante mencionar a aplicabilidade do módulo de otimização que se vale de algoritmos multiobjetivo evolucionários (NSGA-II e MOPSO) propostos nesse trabalho, e do algoritmo genético (AG) (AUSTREGéSILO; CALLOU, 2019). Estas técnicas fazem uso de forma integrada dos formalismos de redes de Petri estocásticas (SPN), diagrama de bloco de confiabilidade (RBD) e modelo de fluxo de energia (EFM). A ferramenta desenvolvida conta com uma visão única da representação dos sistemas computacionais de interesse. A ênfase desta ferramenta consiste em utilizar diferentes formalismos para se avaliar diferentes métricas (custo, disponibilidade, consumo energético, etc.).

A Figura 12 apresenta um modelo da arquitetura elétrica de *data centers* na ferramenta *Stars*. Nela, cada um dos equipamentos é representado por um retângulo que possui seus atributos (MTTF, custo e eficiência energética) e as arestas representam as conexões entre eles. A funcionalidade de otimização (Mod. Otimização) quando acionada sugere ao projetista opções de qual técnica deseja utilizar para otimizar a arquitetura. Selecionada a técnica, uma caixa é apresentada para que sejam fornecidos os valores de entrada para rodar o algoritmo de otimização. É importante frisar que a ferramenta mantém uma base de dados de equipamentos. Ao final do processo é retornado um conjunto de indivíduos da arquitetura de entrada com suas métricas otimizadas.

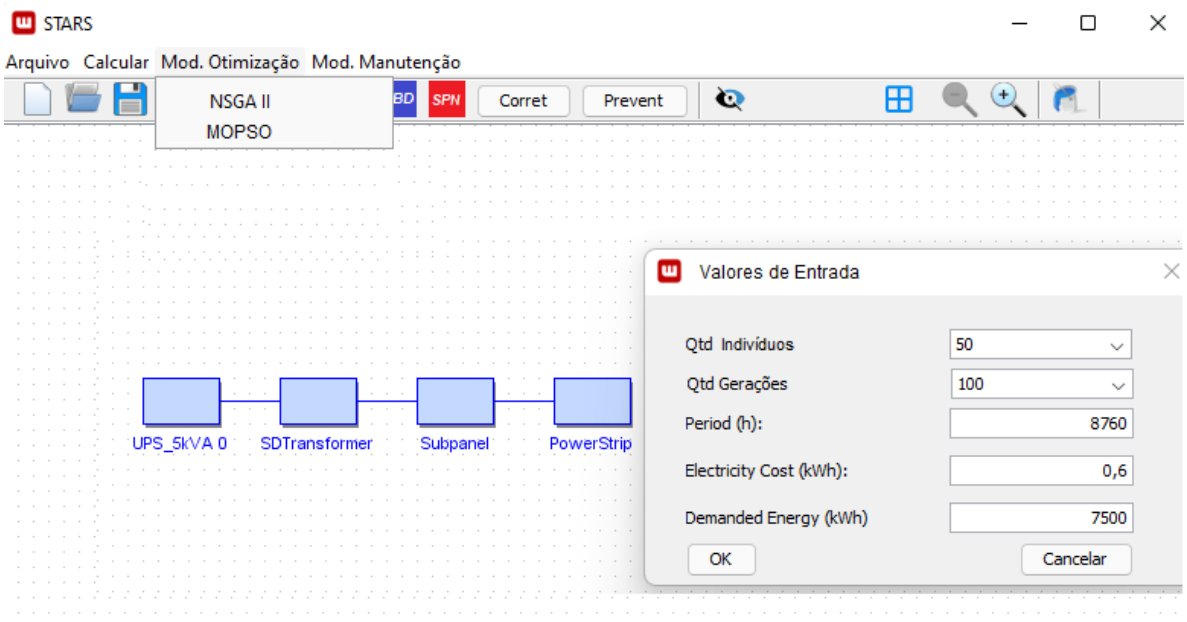


Figura 12 – Módulo de Otimização do *STARS*

3 Trabalhos Relacionados

Neste capítulo são apresentados alguns trabalhos que abordam uma estratégia de otimização multiobjetiva nos subsistemas elétricos de *data center*. Vários trabalhos vêm aplicando modelagens das infraestruturas supracitadas a partir de técnicas de otimização, buscando encontrar soluções sustentáveis e com alta disponibilidade. Os trabalhos relacionados foram encontrados na literatura ao se utilizar a seguinte string de busca: *("All Metadata":multi-objective optimization) AND ("All Metadata":data center) AND ("All Metadata":electrical infrastructure) AND ("All Metadata":availability) AND ("All Metadata":pareto front) AND ("All Metadata":exergy analysis)*.

Em (FERREIRA et al., 2019), os autores projetaram arquiteturas de *data center* que oferecem flexibilidade, automação, otimização e escalabilidade. De acordo com os resultados, a Programação Dinãmica (DP) foi a técnica de otimização que forneceu o mais alto nível de disponibilidade com o mais baixo custo. Entretanto, este trabalho não explorou a eficiência energética das arquiteturas analisadas.

Em (ROSENDO et al., 2019), os autores propuseram uma metodologia para estimar os dados de disponibilidade do *data center* com base nas informações de recursos de hardware fornecidas através da API Redfish. Os resultados da avaliação mostraram que a configuração de vPOD aumenta a disponibilidade quando comparada às estratégias convencionais. Contudo, não foram objetivos do trabalho a análise do custo e a exergia operacional do sistema.

(RAK, 2020) apresentaram um modelo formal que analisa o impacto da variação da temperatura sobre a disponibilidade e confiabilidade nos subsistemas de *data centers*. Os autores utilizaram a modelagem Production Trees (PT), que permite a criação dos modelos levando em conta os fluxos de ar e elétrico do *data center*. Apesar da abrangência, não há nesse trabalho referência a uma técnica de otimização.

Em (GONÇALVES et al., 2020), algoritmos meta-heurísticos foram utilizados para otimizar arquiteturas de *data center*. A estratégia proposta visou comparar qual técnica resulta na maior maximização da disponibilidade e com o mínimo custo de aquisição dos componentes. Ao final do estudo, a técnica que utilizava o algoritmo Differential Evolution (DE) obteve os melhores resultados. Embora tenham empregado técnicas de otimização visando maximizar a disponibilidade, os autores não analisaram a eficiência energética das

arquiteturas.

A abordagem de uma modelagem integrada para avaliar e otimizar a sustentabilidade, a disponibilidade e o custo de *data centers* proposta por (CALLOU et al., 2014) tem semelhança com este trabalho no que tange na utilização de três modelos formais stochastic Petri nets (SPN), Reliability Block Diagrams (RBD) e Energy Flow Model (EFM). Este último modelo é fundamental para a avaliação da métrica de exergia operacional. O algoritmo de otimização usado foi o Greedy Randomized Adaptive Search Procedure (GRASP). O trabalho, contudo, considera uma função objetivo mono-objetivo, e por isso não é capaz de encontrar a curva de Pareto.

O trabalho realizado por (HUANG et al., 2018) propuseram um algoritmo electricity cost optimization (ECO) baseado nos custos energético operacional do *data center*. Resultados experimentais demonstraram que o algoritmo proposto pode resultar em redução significativa do custo de energia, garantindo a qualidade do serviço e o tempo de resposta. No entanto, o trabalho se limita a analisar um equipamento (UPS).

Os autores em (ABUALIGAH; DIABAT, 2021) apresentaram um algoritmo de otimização híbrido para problemas de agendamento de tarefas multiobjetivo em ambientes de computação em nuvem. O algoritmo de otimização foi aprimorada pela utilização da evolução diferencial baseada na elite como uma técnica de busca local para melhorar sua capacidade de exploração e evite ficar preso em ótimos locais. No entanto, este trabalho focou somente em problemas de agendamento de tarefas não ampliando o estudo para os equipamentos que compõem a infraestrutura do ambiente de computação em nuvem.

Os autores em (LI et al., 2020) propuseram uma técnica para otimizar o controle de resfriamento do *data center* por meio da estrutura emergente de aprendizado por reforço profundo (DRL). Foi introduzido um mecanismo de validação de subestimação (DUE) para a rede crítica para reduzir a potencial subestimação do risco causado pela aproximação neural. O foco dos autores foi somente na infraestrutura de cooling e, mesmo assim, não houve cenários com foco na avaliação da disponibilidade e do custo.

Em (WANG et al., 2018) é apresentado um modelo de previsão de temperatura baseado em rede neural artificial para vários campos de fluxo. O modelo aproxima a solução dada por métodos de Dinâmica de Fluidos Computacional, e é usado para formular um problema de otimização não linear restrito com o objetivo de minimizar o consumo de energia do sistema de refrigeração. Esse trabalho apesar de abordar um otimizador

que ajusta as taxas de fluxo de racks individuais com base nas previsões das temperaturas deles baseia-se em uma estratégia mono-objetivo.

Os autores em (MASANET et al., 2020) fizeram uma análise das estimativas globais de uso de energia do *data center*. Este trabalho fornece aos formuladores de políticas e analistas de energia uma compreensão recalibrada do uso global de energia do *data center*, seus fatores e potencial de eficiência de curto prazo. Este trabalho apesar de demonstrar que o estudo do impacto do consumo de energia e formas de otimização devam ser adotadas em seus projetos, não detalha ou propõe um estratégia de otimização multiobjetivo que minimizem o impacto de consumo de energia, mantendo a disponibilidade dos *data centers*.

O trabalho realizado em (EVANS et al., 2020) desenvolveu uma estrutura de rede neural que aprende com dados de operações reais para modelar o desempenho da infraestrutura elétrica do *data center* (DC). O modelo foi amplamente testado e validado nos DCs do Google. Os resultados demonstraram que o aprendizado de máquina é uma maneira eficaz de aproveitar os dados de sensores existentes para modelar o desempenho de DC e melhorar a eficiência energética. Esse trabalho contudo não avaliam outras métricas como a disponibilidade e o custo da infraestrutura elétrica de *data center*.

Os autores em (CHONG et al., 2018) otimizaram a eficiência energética em *data centers*, além de seu dimensionamento tecnológico. Isso inclui a compreensão do espaço de carga de trabalho emergente para garantir a correspondência adequada do design da arquitetura com os requisitos do aplicativo; provisionar recursos de *data center* para cargas de trabalho e aplicativos futuros previstos e melhorar a proporcionalidade de energia de máquinas individuais para correlacionar o consumo de energia à carga; melhorar a integração vertical dentro das pilhas de software de comunicação, armazenamento e sistemas de tempo de execução; e estabelecer padrões para comunicações de hardware e software para permitir a integração de novas tecnologias e *designs* de componentes.

(STAMATESCU et al., 2019) propuseram um algoritmo de otimização que busca reduzir a energia consumida pelo subsistema elétrico de *data center*. Esta ação é realizada levando em consideração as diferentes arquiteturas dando variabilidade de equipamentos a serem utilizados. Apesar de o problema considerado utilizar técnica de otimização baseada em programação linear a métrica de disponibilidade não foi alvo deste estudo, assim sendo, não há uma otimização multiobjetivo neste trabalho.

O trabalho realizado por (SAKALKAR et al., 2020) projetou e implantou uma nova

arquitetura em hardware e software para melhorar significativamente o consumo excessivo de energia demandada pelo subsistema energético de *data center*. Mesmo que por um lado o trabalho tenha demonstrado que a estratégia adotada apresentou redução dos custos em até 25% e preservando a disponibilidade, por outro lado não foi explorada a variabilidade de equipamentos que pudessem gerar e explorar outras arquiteturas. Também não houve avaliação da exergia operacional neste trabalho.

(GHAREHPASHA; MASDARI, 2021) propuseram uma abordagem para otimização multiobjetivo das métricas custo e exergia operacional do *data center*. Eles usam uma combinação do Algoritmo de seno cosseno (SCA) e do Algoritmo de otimização baseado no comportamento de formigas-leão (ALO). O objetivo do modelo proposto foi minimizar o consumo de energia, reduzindo o desperdício de recursos. Apesar de apresentar uma abordagem de otimização por múltiplos critérios, os autores não avaliam a disponibilidade de outras possíveis arquiteturas, não permitindo inferir o comportamento do modelo proposto para outras arquiteturas elétricas de *data centers*.

(GREGORY; YUANSHUN, 2014) apresentaram técnicas para maximizar a disponibilidade a partir da utilização de redundâncias como a cold standby. Os autores propuseram modelos em SPN para estimar o cálculo da disponibilidade. Apesar de utilizar a técnica cold standby, não foi foco dos autores a proposição de estratégias de otimização que possibilitasse a minimizar o custo de infraestrutura elétrica do *data center*.

(AHAT et al., 2021) formularam um modelo de programação linear de números inteiros (MILP) para projetar de forma otimizada uma estrutura de computação multicamadas. Devido à questão da escalabilidade, um algoritmo heurístico baseado na relaxação Lagrangiana da formulação MILP foi proposto para resolver instâncias maiores. Adicionalmente, a fim de fornecer uma oportunidade para os operadores encontrarem uma solução viável em um tempo muito curto, uma abordagem heurística gulosa foi apresentada. Para avaliar o desempenho dos métodos propostos, experimentos computacionais são conduzidos em um amplo conjunto de topologias geradas aleatoriamente. Os resultados indicam que as abordagens propostas podem obter soluções de alta qualidade dentro do prazo determinado.

Em (YAO et al., 2021) foi apresentada uma técnica que aplica algoritmos de *Machine Learning* para solucionar problemas de distribruição de fluxo de energia. A estratégia é aplicada em *data centers* que possuam sistemas de baixa latência, alto

rendimento e escaláveis. A técnica se mostra promissora no balanceamento de carga do *data center* analisado principalmente em sistemas em atividades síncronas, contudo não foram explorados a disponibilidade, nem o custo do subsistema avaliado.

A Tabela 1 ilustra os trabalhos relacionados, ressaltando as diferenças entre o foco de cada pesquisa. Conforme pode ser observado, poucos trabalhos abordam otimização multiobjetivo de modelos formais na avaliação integrada da disponibilidade, custo e exergia, através de modelos em SPN, RBD, EFM no projeto das infraestruturas elétricas de *data centers*.

Tabela 1 – Resumo Comparativo dos Trabalhos Relacionados

Trabalhos Relacionados	Otimização	Modelagem	Métrica	Multiobjetivo
(FERREIRA et al., 2019)	DP	RBD, SPN	Disponibilidade e Custo	✓
(ROSENDO et al., 2019)	vPOD	RBD	Disponibilidade	
(RAK, 2020)	-	*Production Trees*	Disponibilidade e Exergia	
(GONÇALVES et al., 2020)	DE	SPN	Disponibilidade e Custo	✓
(CALLOU etal., 2014)	GRASP	SPN, RBD e EFM	Disponibilidade, Custo e Exergia	
(HUANG et al., 2018)	ECO	Modelo de filas	Custo	
(ABUALIGAH; DIABAT, 2021)	MALO	IaaS	Vazão (CPU)	
(LI et al., 2020)	DRL	*Cooling System Model*	Exergia	
(WANG et al., 2018)	Rede neural	Modelo não-linear	Custo	
(MASANET et al., 2020)	-	SPN RBD	Disponibilidade e Exergia	
(EVANS et al., 2020)	Rede neural	RBD	Exergia	
(CHONG et al., 2018)	EAC	-	Exergia	
(STAMATESCU et al., 2019)	-	RBD	Disponibilidade, Custo	
(SAKALKAR et al., 2020)	FFD	Monte Carlo	Disponibilidade, Custo	✓
(GHAREHPASHA; MASDARI, 2021)	SCA, ALO	-	Exergia	
(GREGORY; YUANSHUN, 2014)	-	SPN	Disponibilidade	
(AHAT et al., 2021)	Programação Linear	MILP	Disponibilidade	
(YAO et al., 2021)	*Machine Learning*	-	Exergia	
(Este Trabalho)	MOPSO, NSGA-II	RBD, SPN e EFM	Disponibilidade, Custo e Exergia	✓

4 Metodologia

Esse capítulo apresenta os passos adotados para realizar a otimização das arquiteturas elétricas de *data centers*. Essa metodologia leva em consideração desde o entendimento do sistema até a validação da estratégia adotada para otimizar as arquiteturas. A Figura 13 mostra uma visão geral da metodologia.

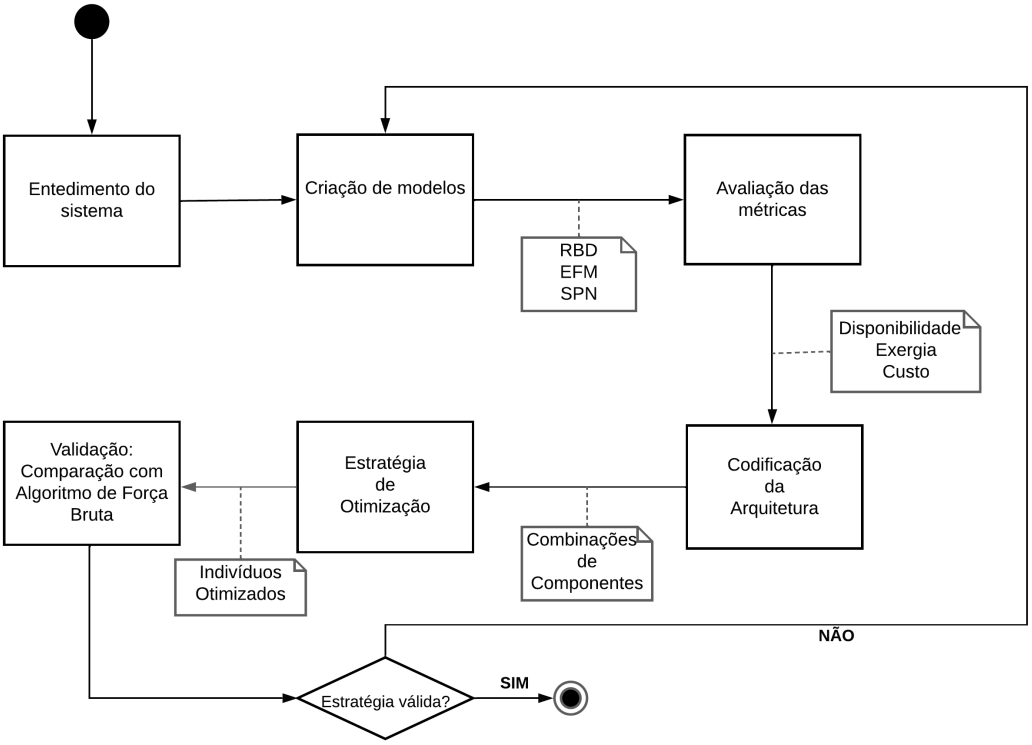


Figura 13 – Metodologia adotada

Vale salientar que duas estratégias de otimização são propostas neste trabalho, ambas baseadas em algoritmos evolucionários multiobjetivos. Além disso, uma técnica de otimização combinatória multiobjetivo é utilizada para validar as estratégias propostas. Trata-se do algoritmo de força bruto explicado na Subseção 5.3.

4.1 Entendimento do Sistema

A primeira fase da metodologia deste trabalho é o entendimento do sistema. Ele é essencial para compreender o funcionamento do sistema real e, a partir disso, ser capaz de representá-lo através de modelos. O subsistema elétrico de *data center* considerado para

este estudo é composto, basicamente, por UPS (*Uninterruptible Power Supplies*), SDT (*Step Down Transformer*), *subpanel* e *powerstrip* (TIA-942, 2005).

Na criação das arquitetura foram utilizados equipamentos redundantes, por isso foram necessários uso de comutadores STS (*static transfer switches*). Este equipamento é responsável em transferir cargas entre dois equipamentos independentes sem interrupção.

A Figura 14 apresenta uma das arquiteturas que foi levada em consideração para o estudo. Dois tipos de modelos de redundância estão presentes nesta arquitetura a *hot standby* e a *cold standby*.

Redundância *hot standby* é aquela em que o equipamento redundante sempre fica ativo. Assim sendo, quando houver falha no equipamento principal, esse pode ser substituído pelo equipamento redundante sem um atraso de tempo de ativação. O segmento principal do modelo da Figura 14 é $UPS1 \rightarrow STS1 \rightarrow SDT1 \rightarrow Subpanel1 \rightarrow Powerstrip1$ e está em redundância *hot standby* com o seguimento $UPS2 \rightarrow STS2 \rightarrow SDT2 \rightarrow Subpanel2 \rightarrow Powerstrip2$.

Redundância *cold standby* o componente redundante não está ativo. Ele espera para ser ativado quando o módulo principal falhar. Quando o módulo principal falha, a ativacão do módulo redundante ocorre em um certo período de tempo que é denominado de *Mean Time To Activate* (MTTA). O $UPS3$ está em *cold standby* (representado por sua ligação pontilhada), sendo sua ativação efetivada apenas nas falhas do $UPS1$ ou $UPS2$.

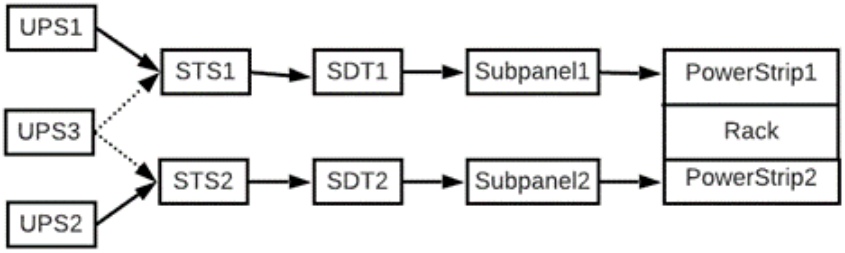


Figura 14 – Arquitetura elétrica de *data center*

4.2 Modelos

A segunda etapa da metodologia proposta corresponde à criação dos modelos (ex. RBD, SPN e EFM). Esses modelos são criados através da visão de alto nível oferecida pela ferramenta Stars (LEONARDO; CALLOU, 2021), que auxilia na integração da estratégia

de otimização com o ambiente de avaliação do Mercury (OLIVEIRA et al., 2017). Os parâmetros dos equipamentos como tempo médito de falha (MTTF), tempo médio de reparo (MTTR), custo e eficiência energética que compõem as arquiteturas modeladas neste trabalho foram extraídos de (GUIMARÃES; PEREIRA, 2020) e (CALLOU et al., 2014).

4.2.1 Modelagem em RBD

Definida a arquitetura, por exemplo a Figura 14, é necessária a criação de um modelo formal que permita representar relações interação entre componentes, como mecanismos de redundância, além de poder avaliar a disponibilidade deste modelo. Essas características são inerentes ao modelo em RBD. A Figura 15 mostra o modelo em RBD da arquitetura elétrica do *data centers* apresentado na Figura 14. A disponibilidade (D) pode ser obtida pela avaliação do comportamento do sistema através da expressão $D = (B1 \wedge STS1 \wedge SDT1 \wedge SUBPANEL1 \wedge POWERSTRIP1)|(B1 \wedge STS2 \wedge SDT2 \wedge SUBPANEL2 \wedge POWERSTRIP2)$.

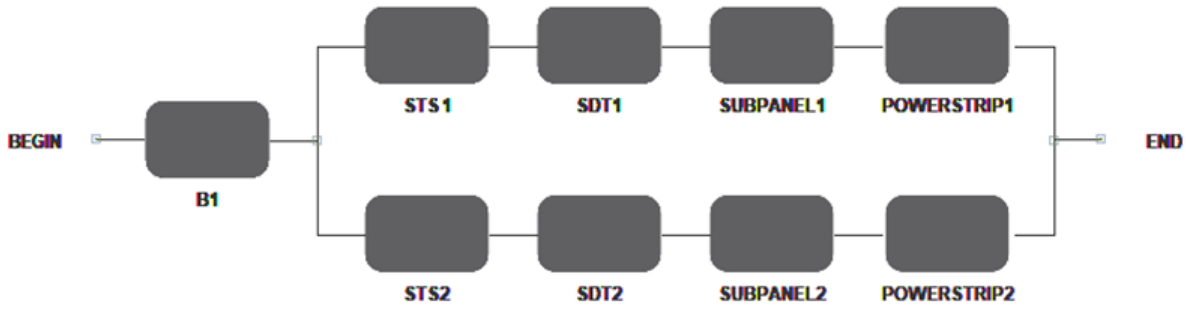


Figura 15 – Modelo RBD baseado na arquitetura da Figura 14

Com exceção dos UPSs cada equipamento foi modelado em um bloco RBD. A explicação para isto se deve ao fato de que nem todos os cenários é possível ser modelado em RBD de forma direta. Surge, então, a necessidade de uma abordagem híbrida, onde parte da disponibilidade do sistema avaliada em SPN e seu resultado passado para um bloco RBD. Isto ocorre com o bloco B1 do modelo da Figura 15 que representa a disponibilidade do sistema com os UPSs, conforme é detalhado a seguir.

4.2.2 Modelo Híbrido

Existem cenários em que apenas o RBD seria insuficiente para se avaliar a métrica da disponibilidade. Por exemplo, na arquitetura da Figura 14, o equipamento UPS3 só será ativado quando o UPS1 ou UPS2 estiverem em situação de falha. A Figura 16 apresenta o modelo SPN proposto para representar esse caso. Nesse modelo, a falha do UPS1 ou UPS2 faz com que um *token* seja gerado no lugar Controle, habilitando a transição T_ativ que é responsável pela ativação do UPS3. Após o tempo de ativação do UPS3, a ficha de Controle é consumida e o lugar UPS3_ON, inicialmente vazio, receberá um *token*. A disponibilidade (D) desse modelo é computada por: $D = P\{((\#UPS1\_ON = 1)OR(\#UPS3\_ON = 1))AND((\#UPS2\_ON = 1)OR(\#UPS3\_ON = 1))\}$.

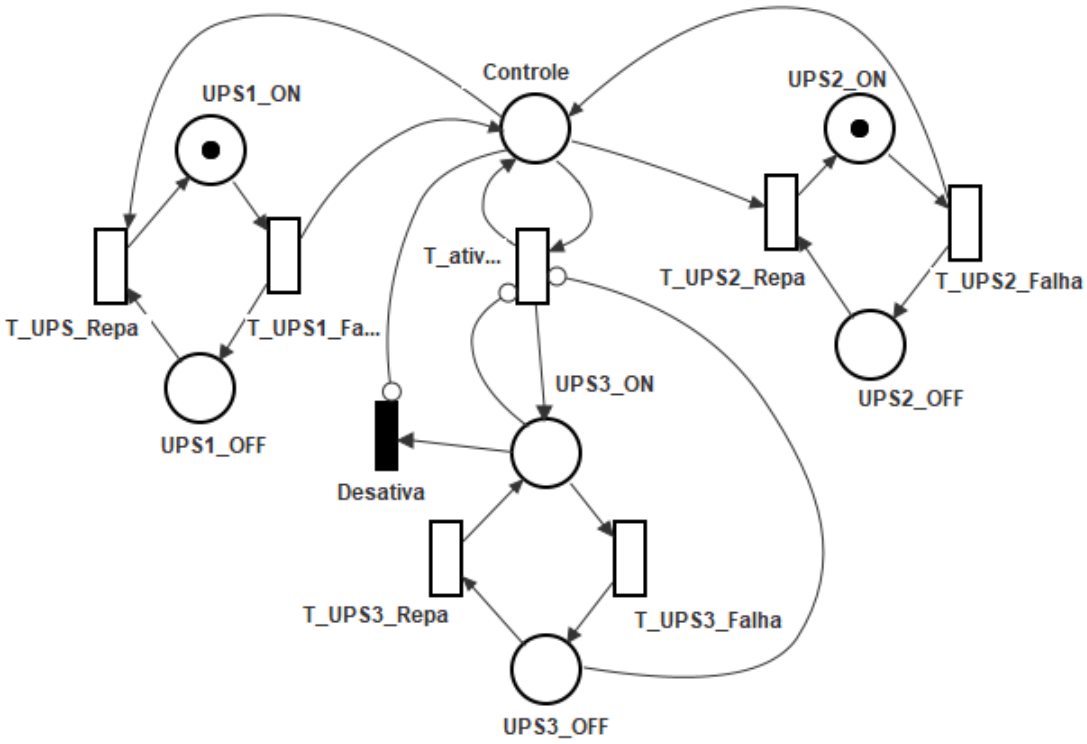


Figura 16 – Modelo SPN – UPS1, UPS2 e UPS3 (*cold standby*).

A Figura 17 apresenta o modelo híbrido, onde a parte em redundância *cold standby* é modelada em SPN. O resultado da avaliação da disponibilidade desse modelo é utilizado como parâmetro de entrada para um bloco RBD (B1), conforme mostrado na Figura 17 (c). Uma vez que se tem o bloco B1 parametrizado, pode-se avaliar todo a disponibilidade de toda arquitetura analisada.

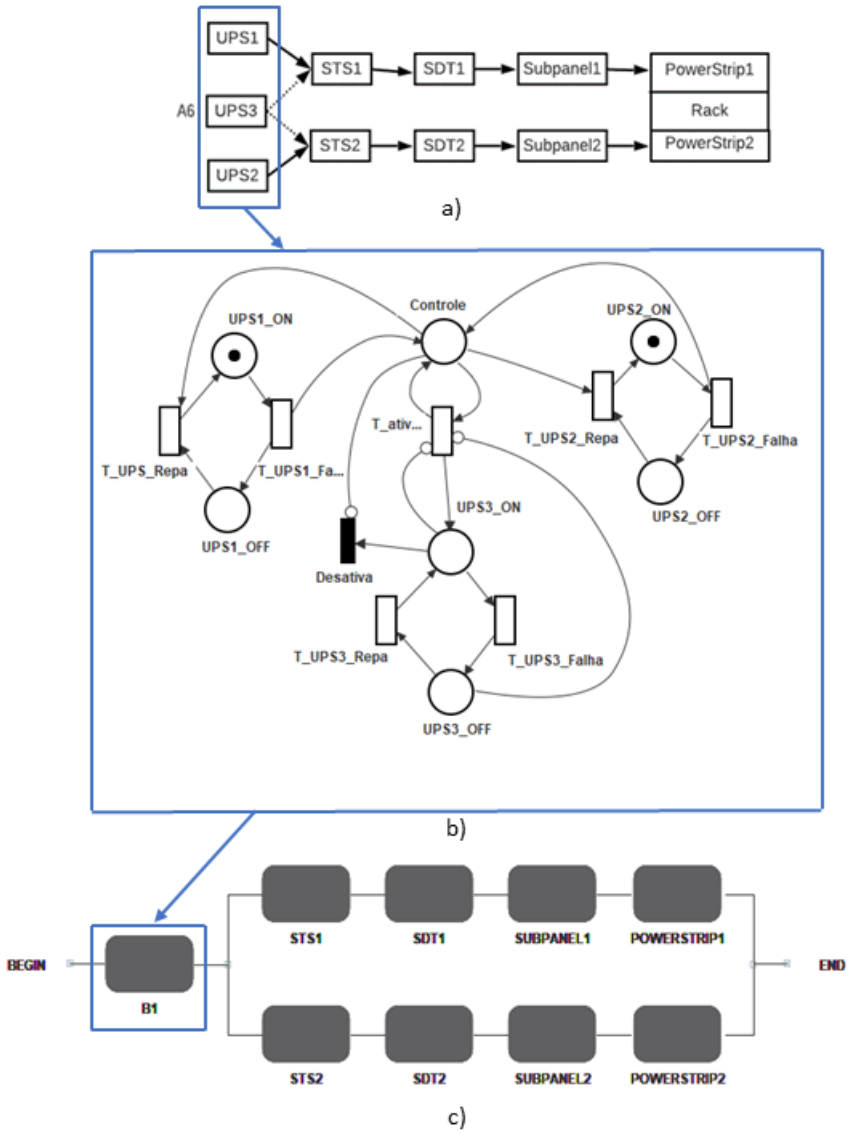


Figura 17 – Integração entre modelos. Modelagem Cold standby (UPS1, UPS2 e UPS3) em SPN convertida em um bloco RBD - B1

4.2.3 Modelo EFM

O modelo EFM representa o fluxo de energia entre os componentes do sistema do data center considerando a eficiência energética e a capacidade máxima de energia de cada componente. O sistema em avaliação pode estar disposto corretamente, no sentido de que os componentes necessários estejam devidamente conectados, mas podem não ser capazes de atender a demanda do sistema por energia elétrica ou carga térmica (CALLOU et al., 2014). O modelo EFM neste trabalho é responsável por avaliar as métricas de exergia operacional e custo do sistema modelado. Esses resultados auxiliam projetistas de *data centers* a propor um novo ambiente ou otimizar um já existente. A Figura 18 apresenta os

modelos EFM das arquiteturas baseadas na Figura 14.

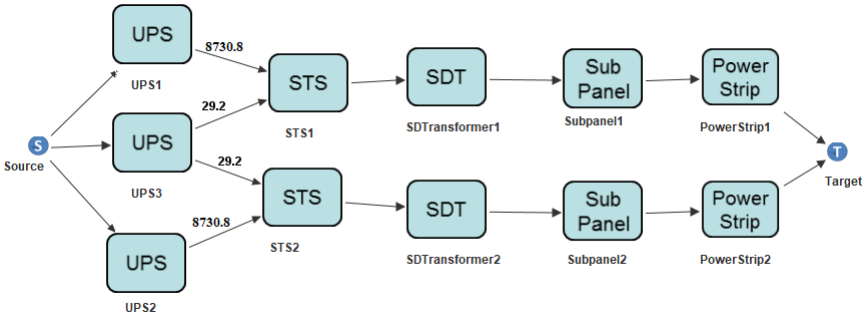


Figura 18 – Modelo EFM baseado na arquitetura da Figura 14

O EFM ilustrado possui pesos nas arestas de saída dos UPS. Isso ocorre pois, o UPS3 somente é acionado durante o período de falha de pelo menos um dos outros dois UPS (UPS1 ou UPS2). Sendo assim, considerando o período de 1 ano (8760h) o tempo de efetivo uso do UPS3 será de 29.2h. É importante destacar que foram adotados os valores de custo da energia elétrica (por KWh) de acordo com o padrão médio dos EUA (US$ 0.11), trata-se portanto, do custo operacional (ver Equação 2.4). Além disso, é pelo EFM que se avalia o custo de cada equipamento da arquitetura obtido por pesquisa de mercado.

4.3 Avaliação das Métricas: Conversão de Modelos em Linguagem de *Script*

Definidos os modelos é chegada a terceira etapa da metodologia que tem como objetivo avaliar as métricas de interesse. O uso da conversão dos modelos no padrão da linguagem de *script* faz-se necessário neste trabalho para auxiliar na integração entre da ferramenta *Stars* (LEONARDO; CALLOU, 2021), que inclui o módulo de otimização, e o Mercury que avalia os modelos criados. Além disso, o padrão de script do Mercury permite que se tenha parte do sistema modelado em um formalismo e o restante em outro formalismo. Os modelos RBD, SPN e EFM explorados nesta subseção são aqueles já explorados tomando por base a arquitetura da Figura 14. A métrica objetivo que se vale dos modelos elaborados em RBD e SPN é a disponibilidade. Já para avaliar a exergia operacional e o custo da referida arquitetura analisada é utilizado a conversão do modelo EFM.

A conversão dos modelos foi definida dentro de um único *script*.Para melhor a compreensão didática, a conversão dos modelos em linguagem de *script* foi dividida em

quatro partes, onde as três primeiras tratam das especificidades de cada modelo (SPN, RBD e EFM) e a última trata do método principal (*main*). Neste último, estão reunidas as métricas como disponibilidade, exergia operacional e custo. É importante frisar que o *script* segue uma numeração sequencial em relação ao modelo anterior.

A Figura 19 mostra a representação na linguagem de script do modelo em SPN *cold standby* da Figura 16. O *script* inicia com a declaração do tipo de modelo e seu nome. Das linhas 2 até 8, são declarados os lugares UPS1, UPS2 e o UPS3, e a quantidade de *tokens* de cada lugar. As transições imediatas (*immediateTransition*) e as transições temporizadas (*timedTransition*) são definidas das linhas 10 até 48. Destas, as linhas 10 até 12 representa a transição imediata, que difere das temporizadas (linhas 14 a 48) por terem delay associado. Por exemplo, o *delay* pode ser utilizado para representar o MTTF e o MTTR. Vale ressaltar que na definição de todas as transições temos também os lugares de entrada (*inputs*), e saída (*outputs*). Na linha 51 é definida a métrica *Disp*1, que é utilizada para o cálculo da disponibilidade e será passada para o bloco B1 do modelo RBD. É necessário definir o tipo de análise que está sendo feita, no caso análise estacionária.

A segunda parte do script é apresentado na Figura 20, e corresponde a conversão do modelo da Figura 15 em linguagem de *script*. Semelhantemente ao modelo anterior, o início é reservado para definir o tipo do modelo e o seu nome, isto é feito na linha 57. A linha 58 é onde ocorre a integração do modelo SPN *cold standby* com o modelo RBD. Nessa mesma linha, a disponibilidade (*Disp*1) que foi avaliada na Figura 19 é passada como parâmetro para o bloco B1 através do método *hierarchy*. Isto é feito com o método *solve* que tem como entrada o modelo e a métrica (SPNModel, Disp1) que será passada para o bloco B1.

Os demais blocos são definidos nas linhas 59 até 66. Cada bloco definido do RBD é construído seguindo a regra: bloco <nome_equipamento>, (MTTF_equipamento, MTTR_equipamento). As linhas 68 a 71 definem a estrutura do RBD (série e/ou paralelo). Neste exemplo, os blocos definidos em s2 estão conectados em série, o mesmo ocorre com os blocos definidos em s3. Já p1 define que os blocos presentes em s2 se encontram em paralelos aos blocos da estrutura s3. A linha 71 define s0 conectando B1 em série com p1. Vale ressaltar que o bloco B1 tem a disponibilidade do modelo dos UPS em redundância *cold standby*. A linha 73 define que s0 será o modelo inicial. E, finalmente, a linha 75 define a métrica av que calcula a disponibilidade do modelo.

```
SPN Model{
     place Controle;
     place UPS1_OFF;
     place UPS1_ON( tokens= 1 );
     place UPS2_OFF;
     place UPS2_ON( tokens= 1 );
     place UPS3_OFF;
     place UPS3_ON;

     immediateTransition Desativa(
          inputs = [UPS3_ON],
          inhibitors = [Controle]
     );
     timedTransition T_UPS1_Falha(
          inputs = [UPS1_ON],
          outputs = [UPS1_OFF, Controle],
          delay = mttf_ups1
     );
     timedTransition T_UPS2_Falha(
          inputs = [UPS2_ON],
          outputs = [UPS2_OFF, Controle],
          delay = mttf_ups2
     );
     timedTransition T_UPS3_Falha(
          inputs = [UPS3_ON],
          outputs = [UPS3_OFF],
          delay = mttf_ups3
     );
     timedTransition T_UPS_Repa(
          inputs = [UPS1_OFF, Controle],
          outputs = [UPS1_ON],
          delay = mttr_ups1
     );
     timedTransition T_UPS2_Repa(
          inputs = [UPS2_OFF, Controle],
          outputs = [UPS2_ON],
          delay = mttr_ups2
     );
     timedTransition T_UPS3_Repa(
          inputs = [UPS3_OFF],
          outputs = [UPS3_ON],
          delay = mttr_ups3
     );
     timedTransition T_ativacao(
          inputs = [Controle],
          outputs = [UPS3_ON, Controle],
          inhibitors = [UPS3_ON, UPS3_OFF],
          delay = 0.5
     );

     metric Disp1 = stationaryAnalysis(
          expression= "P {((#UPS1_ON =1)OR(#UPS3_ON = 1))AN D((#UPS2_ON = 1)OR(#UPS3_ON = 1))}");
}
```

Figura 19 – Conversão do SPN *cold standby* apresentado na Figura 16

A terceira parte do script representa o modelo EFM (ver Figura 21). Duas métricas deste modelo são utilizadas neste trabalho (custo total e exergia operacional). Não é por acaso que este será o terceiro script a ser estudado, mas sim pelo fato dele depender da disponibilidade do modelo (calculada no RBD), para enfim, poder calcular as outras duas métricas que o EFM contempla.

Semelhantemente aos dois outros modelos já apresentados, a linha 78 inicia o script com a declaração do tipo de modelo e seu nome. Em seguida, as linhas 79 a 166

```
57 RBD Model{
58    block B1( availability = solve (ModelSPN), Disp1);
59    block STS1( MTTF = mtt_sts1, MTTR = 6.0);
60    block SDT1( MTTF = mtt_sdt1, MTTR = 156.01);
61    block SUBPANEL1( MTTF = mtt_subpanel1, MTTR = 2.4);
62    block POWERSTRIP1( MTTF = mtt_powerstrip1, MTTR = 3.8);
63    block STS2( MTTF = mtt_sts2, MTTR = 6.0);
64    block SDT2( MTTF = mtt_sdt2, MTTR = 156.01);
65    block SUBPANEL2( MTTF = mtt_subpanel2, MTTR = 2.4);
66    block POWERSTRIP2( MTTF = mtt_powerstrip2, MTTR = 3.8);
67
68    series s2(STS1, SDT1, SUBPANEL1, POWERSTRIP1 );
69    series s3(STS2, SDT2, SUBPANEL2, POWERSTRIP2 );
70    parallel p1(s2, s3);
71    series s0(B1, p1 );
72
73    top s0;
74
75    metric av = availability;
76 }
```

Figura 20 – Representação do RBD da Figura 15 na linguagem de *script* do Mercury

apresentam as declarações dos componentes do modelo EFM. A criação de componentes segue a seguinte regra: palavra reservada *component* <nome_componente>, tipo do componente e seus parâmetros potência máxima, eficiência, preço de mercado e exergia (*maxPower*, *efficiency*, *retailPrice*, *embeddedExergy*).

Os arcos na linguagem de *script* são representados por *arc*. No modelo EFM, os arcos são direcionados para representar o fluxo de energia conforme pode ser visto das linhas 168 até 182. Partindo do *Source* e ligando aos 3 UPS. Os UPS1 e UPS3 têm ligações ao STS1, já o UPS2 e UPS3 têm ligações ao STS2. Vale salientar que os arcos possuem pesos, uma vez que o UPS3 está em *cold standby*, ou seja, ele só será acionado quando o UPS1 ou UPS2 estiverem em falha.

Isso faz com que os UPS1 e UPS2 tenham um maior fluxo energético que passa por tais dispositivos. Por fim, temos dois caminhos. O primeiro que trata das ligações saindo do STS1 parte para o SDT1, e este por sua vez liga-se ao *Subpanel*1, até por chegar ao *PowerStrip*1 que é o último componente da arquitetura que se liga ao *Target*. O segundo é similar ao primeiro, portanto, parte do STS2 até chegar ao *PowerStrip*2. Ambos os caminhos convergem no *Target*.

As linhas 184 a 189 são referentes as métricas do modelo EFM. Nota-se a dependência da disponibilidade como parâmetro para o cálculo das métricas tanto do custo operacional (oc), quanto para exergia operacional (oe), linhas 151 a 156. Neste trabalho, as métricas que serão utilizados como objeto de otimização são custo total (*metric* tc) e

```
78  EFM ModelEFM{
79       component Source(
80            type = "SourcePoint",
81            parameters = (
82                 efficiency = 100.0,
83                 retailPrice = 0.0));
84       component Target(
85            type = "TargetPoint",
86            parameters = (
87                 efficiency = 100.0,
88                 retailPrice = 0.0,
89                 demandedEnergy = de));
90       component UPS1(
91            type = "UPS_5kVA",
92            parameters = (
93                 maxPower = mp1,
94                 efficiency = ef1,
95                 retailPrice = rp1,
96                 embeddedEnergy = emb1));
97       component UPS2(
98            type = "UPS_5kVA",
99            parameters = (
100                maxPower = mp2,
101                efficiency = ef2,
102                retailPrice = rp2,
103                embeddedEnergy = emb2));
104      component UPS3(
105           type = "UPS_5kVA",
106           parameters = (
107                maxPower = mp3,
108                efficiency = ef3,
109                retailPrice = rp3,
110                embeddedEnergy = emb3));
111      component STS1(
112           type = "STS",
113           parameters = (
114                maxPower = mp4,
115                efficiency = ef4,
116                retailPrice = rp4,
117                embeddedEnergy = emb4));
118      component STS2(
119           type = "STS",
120           parameters = (
121                maxPower = mp5,
122                efficiency = ef5,
123                retailPrice = rp5,
124                embeddedEnergy = emb5));
125      component SDT1(
126           type = "SDTransformer",
127           parameters = (
128                maxPower = mp6,
129                efficiency = ef6,
130                retailPrice = rp6,
131                embeddedEnergy = emb6));
132      component SDT2(
133           type = "SDTransformer",
134           parameters = (
135                maxPower = mp7,
136                efficiency = ef7,
137                retailPrice = rp7,
138                embeddedEnergy = emb7));
139      component Subpanel1(
140           type = "Subpanel",
141           parameters = (
142                maxPower = mp8,
143                efficiency = ef8,
144                retailPrice = rp8,
145                embeddedEnergy = emb8));
146      component Subpanel2(
147           type = "Subpanel",
148           parameters = (
149                maxPower = mp9,
150                efficiency = ef9,
151                retailPrice = rp9,
152                embeddedEnergy = emb9));
153      component PowerStrip1(
154           type = "PowerStrip",
155           parameters = (
156                maxPower = mp10,
157                efficiency = ef10,
158                retailPrice = rp10,
159                embeddedEnergy = emb10));
160      component PowerStrip2(
161           type = "PowerStrip",
162           parameters = (
163                maxPower = mp11,
164                efficiency = ef11,
165                retailPrice = rp11,
166                embeddedEnergy = emb11));
167
168      arc Source -> UPS1;
169      arc Source -> UPS2;
170      arc Source -> UPS3;
171      arc UPS1 -> STS1( weight = 8730.8);
172      arc UPS3 -> STS1( weight = 29.2);
173      arc STS1 -> SDTransformer1;
174      arc SDTransformer1 -> Subpanel1;
175      arc Subpanel1 -> PowerStrip1;
176      arc PowerStrip1 -> Target;
177      arc UPS2 -> STS2( weight = 8730.8);
178      arc UPS3 -> STS2( weight = 29.2);
179      arc STS2 -> SDTransformer2;
180      arc SDTransformer2 -> Subpanel2;
181      arc Subpanel2 -> PowerStrip2;
182      arc PowerStrip2 -> Target;
183
184      metric ic = initialCost( eletricityCost = ec);
185      metric oc = operationalCost(
186                 eletricityCost = ec, availability = aval, time = period );
187      metric oe = operationalExergy( time = period, availability = aval);
188      metric tc(ic + oc);
189 }
```

Figura 21 – Representação do EFM (ver Fig. 18) na linguagem de script do Mercury

a exergia operacional (*metric* oe), além da disponibilidade já apresentada nos modelos anteriores.

A Figura 22 apresenta a parte final do script que inicia na linha 191 com a declaração do método principal (*main*). Este método é o responsável por acionar a execução dos demais métodos e, assim, ele reúne as métricas disponíveis nos modelos apresentados. As linhas 192 e 193 são utilizadas para acionar o modelo SPN, enquanto que as linhas 194 e 195 vão imprimir os resultados das avaliações. Na linha 197 o modelo RBD é acionado e a linha 198 imprime o resultado da disponibilidade. Similarmente, as linhas 200 a 210 acionam o modelo EFM para calcular as métricas de custo e exergia operacional.

```
main{
    Disp1=solve (ModelSPN, Disp1);

    print(Disp1);


    av = solve(ModelRBD, av);
    println("Availability: " .. av );

    ic = solve(ModelEFM, ic);
    println("Acquisition Cost: " .. ic);
    oc = solve(ModelEFM, oc);
    println("Operational Cost: " .. oc);
    tc = solve(ModelEFM, tc);
    println("Total Cost: " .. tc);

    println("");

    oe = solve(ModelEFM, oe);
    println("Operational Exergy: " .. oe);
}
```

Figura 22 – Função principal do script

4.4 Codificação da Arquitetura

É importante destacar que em cada arquitetura analisada iremos otimizar as métricas de disponibilidade, custo e exergia operacional. Cada arquitetura é composta por equipamentos, e a partir de uma lista de dispositivos disponíveis para cada equipamento, teremos os parâmetros que podem variar (MTTF, MTTR, custo de aquisição e eficiência energética). O Algoritmo 1 mostra como pegar um componente da base de dados.

Algoritmo 1 pegarComponente (linhaBaseDados)

1: *Componente comp*;
2: *comp.preencherParametros*(*linhaBaseDados*);
3: **retorno** *comp*;

A entrada para este Algoritmo é uma linha da base de dados de componentes. Essa linha contém os atributos, conforme ilustrado na Figura 23. Um objeto *comp* do tipo Componente é criado na linha 1, este tem por finalidade guardar os atributos que são preenchidos (linha 2) pelo método *preencherParametros*. Esse método lê uma linha da base de dados, onde os atributos estão separados por vírgulas, e passa para os atributos de *comp* (inicialmente vazios) os valores lidos. No fim, retorna o componente com seus respectivos atributos.

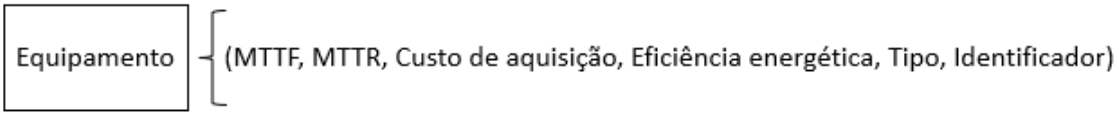


Figura 23 – Atributos dos Equipamentos da Infraestrutura Elétrica de *Data Center*

Vale ressaltar que para fins de organização de consulta a base de dados foi mapeada em um dicionário de dados como pode ser visto no Algoritmo 2. Este Algoritmo recebe a base de dados (.txt) onde cada linha tem os atributos de um componente de infraestrutura elétrica de *data center* (valores para MTTF, MTTR, Custo de aquisição, Eficiência energética, Tipo, Identificador). Na linha 1 é criada a estrutura *dicComp*. As linhas de 2 a 4 são responsáveis por realizar a leitura até o fim do arquivo (linha 3). Para cada linha é chamado o método *montarComponente* que traz um componente daquela linha e este é guardado dentro do dicionário, organizando-os a partir do seu tipo (linha 4). O retorno é um dicionário de componentes.

Algoritmo 2 mapearBase (baseDados)

1: $map < String, Componente >\ dicComp$;
2: **enquanto** baseDados.$temLinha()$ **faça**
3: $\quad linha \leftarrow baseDados.lerLinha()$;
4: $\quad dicComp.pegar(linha.tipo).colocar(montarComponente(linha))$;
5: **fim enquanto**
6: **retorno** $dicComp$;

A codificação de um indivíduo será feita tomando por base uma arquitetura que define a quantidade e posições dos componentes (TIA-942, 2005). Definido o padrão, segue-se para etapa de preenchimento do conjunto de equipamentos. Cada um desses equipamentos possui um tipo e a conexão entre eles seguem o padrão definido na arquitetura. A Figura 24 ilustra como é feita a construção de um indivíduo. Essa construção utiliza uma base de dados que é mapeada num dicionário usando como chaves *Strings* que se referem aos tipos de componentes existentes. Definida a arquitetura, os equipamentos que compõem o indivíduo são percorridos um a um. A partir desse mapeamento, uma função de escolha aleatória e sem repetição seleciona um equipamento do mesmo tipo e insere seus atributos (MTTF, MTTR, custo de aquisição e eficiência energética) no equipamento do indivíduo que está sendo percorrido no momento. Esse processo se repete até o último equipamento do indivíduo.

Definida a codificação do indivíduo, é preciso gerar uma população inicial. Este papel é desempenhado pelo Algoritmo 3. Ele recebe como entrada uma arquitetura (*arqModelo*), a base mapeado (*dicBase*) e o tamanho da população (*tamPop*). A linha 1 *popInicial* é instância de uma lista de indivíduos. As linhas 2 a 7 correspondem a um laço responsável por criar um indivíduo (*ind*). Em seguida, é setada a arquitetura deste

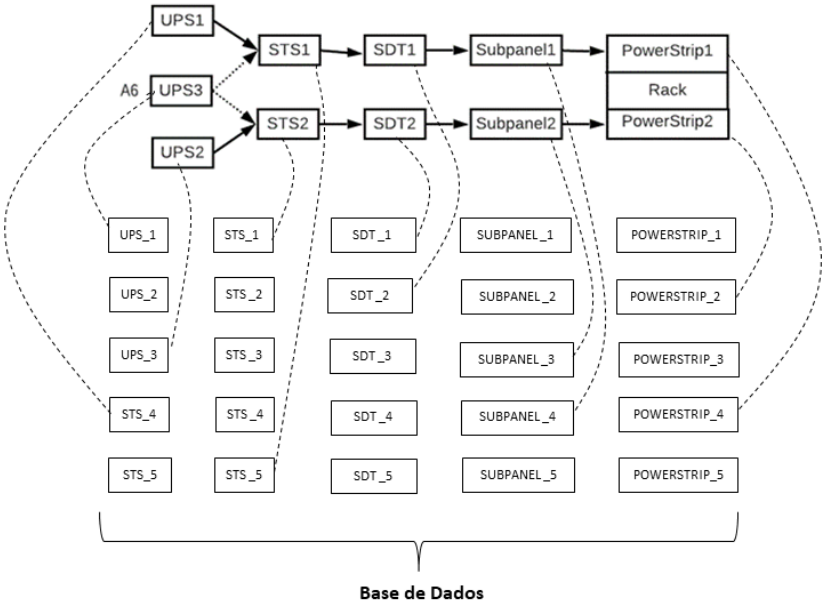


Figura 24 – Indivíduo Gerado pelo Mapeamento da Base

indivíduo (*arqModelo*). Logo depois, os equipamentos que compõe a arquitetura têm os seus parâmetros setados. O retorno é a população inicial (*popInicial*).

Algoritmo 3 gerarPopIncial(arqModelo, dicBase, tamPop)

```
1: Lista popIncial;
2: para i ← 1 até tamPop faça
3:     Individuo ind;
4:     ind.setArquitetura(arqModelo);
5:     ind.setParametroComponentes(dicBase);
6:     popIncial.adicionar(ind);
7: fim para
8: retorno popInicial;
```

Cada indivíduo desta população necessita ter suas métricas avaliadas, determinando os valores de sua disponibilidade, custo e exergia operacional. O Algoritmo 4 é responsável por computar o valor de cada uma das funções objetivos. Esse algoritmo recebe um indivíduo de entrada e, em seguida, as linhas 1 a 3 fazem os cálculos das funções objetivo. Com o método *solucao* são feitas as chamadas ao método *solve* do Mercury que recebe o modelo convertido em linguagem de *script* e retorna a métrica avaliada.

Algoritmo 4 avaliarMetrica(individuo)

```
1: individuo.setDisp ← solucao(RBD, disponibilidade);
2: individuo.setCusto ← solucao(EFM, custoTotal);
3: individuo.setExerOp ← solucao(EFM, exergiaOperacional);
```

5 Estratégias de Otimização

Este capítulo aborda as duas últimas fases da metodologia. Nele serão apresentadas as estratégias propostas baseadas em duas técnicas de otimização multiobjetivo aplicadas em sistema elétrico de *data center*. Na Seção 5.1 será apresentado uma adaptação do algoritmo de otimização baseada no NSGA-II, além das principais funções que auxiliam na construção da estratégia. De modo semelhante, é feito na Seção 5.2, sendo a técnica abordada, desta vez, a baseada no MOPSO. Por fim, na Seção 5.3 é apresentada a validação das estratégias propostas.

5.1 Estratégia baseada no NSGA II

O NSGA-II possui dois principais operadores: ordenação de indivíduos não dominados da população e uma métrica de promoção de diversidade da população, *Crowding Distance* (CD). A estratégia adotada é apresentada no Algoritmo 5. Suas entradas são o modelo de arquitetura que deve ser baseada nos padrões da (TIA-942, 2005), o tamanho da população, número de iterações do algoritmo e as funções alvo da otimização (disponibilidade, custo total e exergia operacional). A saída é um conjunto solução Pareto aproximado contendo arquiteturas otimizadas.

A linha 1 corresponde a inicialização de uma a população inicial. A população é composta por diversos indivíduos (lista de indivíduos). No caso deste trabalho, os modelos das arquiteturas dos subsistemas elétricos de *data center* representam a população. A linha 2 cria uma lista de listas de indivíduos. Isto é necessário para posterior análise estatística da estratégia proposta, pois a cada geração a população é salva em *popGer*. O laço das linhas 4 a 6 serve para avaliar as métricas de cada indivíduo. Isso é feito através da comunicação com o ferramental Mercury, especificamente pelos métodos *metric* e *solve*, ambos necessários para obter os resultados das funções objetivo. A linha 7 atribui à lista REP o resultado do método *ranquear* que tem a população inicial ranqueada por soluções não dominadas. O laço principal da estratégia inicia na linha 8, tendo como condição de parada o número de gerações. O laço interno, na linha 9, indica que é executado $\frac{N}{2}$ vezes, isso porque a cada chamada do *selTorneiroBinario* um par de pais são escolhidos baseado em elitismo. Uma vez definido os pais, na linha 11 é feito o cruzamento deles aplicando o

Algoritmo 5 Estratégia de Otimização Baseado no NSGA-II

Entrada: $N, g, arqModelo, dicBase$

1: $popInicial \leftarrow gerarPopInicial(arqModelo, dicBase, N)$;
2: $Lista\ popGer$;
3: $Lista\ REP$;
4: **para** Individuo ind : popInicial **faça**
5: ind.$avaliarMetricas()$;
6: **fim para**
7: $REP \leftarrow ranquear(popInicial)$;
8: **para** $(i \leftarrow 1$ **até** $g)$ **faça**
9: **para** $(j \leq \frac{N}{2})$ **faça**
10: $listPais \leftarrow selTorneioBinario(REP)$;
11: $listFilhos \leftarrow aplicarOperadoresGen(ListPais)$;
12: $descendentes.adionarVarios(listFilhos)$;
13: **fim para**
14: $REP.adionarVarios(descentedes)$;
15: $popGer.adicionar(ranquear(REP))$;
16: $REP \leftarrow descartarPiores(popGer.pegar(i))$;
17: **fim para**
18: **retorno** REP;

método *aplicarOperadoresGen*. Utiliza-se os operadores genéticos de cruzamento e mutação, onde através da seleção torneio um par de indivíduos são tomados. Então, a partir de um ponto de cruzamento, parte dos equipamentos de um indivíduo são trocados com o outro par. Em seguida pode haver mutação (taxa de 10%), onde um equipamento de um indivíduo pode ser mudado por outro do mesmo tipo, que esteja na base de dados. A linha 12 faz uma chamada ao método *adicionarVarios* que é o responsável por adicionar os descendentes gerados à população. Novamente há o ranqueamento (linha 15), por conta da inclusão de novos indivíduos na população e a geração atual tem sua população salva em *popGer*. Feito isto, é necessário descartar as piores arquiteturas baseado no método *crownding distance* (linha 16). Ao final de g gerações, o Algoritmo 5 traz como resposta um conjunto de indivíduos contendo a solução otimizada de forma multiobjetiva do modelo da arquitetura adotada, isto é, a fronteira de Pareto aproximada. Outros critérios de parada podem ser utilizados, por exemplo, um erro absoluto que compara a diferença absoluta entre duas iterações do algoritmo, checando se o valor do erro especificado pelo projetista de *data center* foi atingido. É importante frisar, que um erro de 10^{-5} pode ser razoável quando os valores da função objetivo são da ordem de 10, agora, para valores pequenos ou em termos percentuais, pode ser utilizado o erro relativo.

O Algoritmo 6 tem como objetivo fazer o ranqueamento das soluções não dominadas

e sua entrada para é uma população (lista de indivíduos). Na primeira linha, *proxRank* vai receber a população de entrada (*pop*). As linhas 3 e 4 criam, respectivamente, listas de indivíduos para guardar tanto os ranqueados como os não ranqueados. As linhas 4 e 5 inicializam as seguintes variáveis: n para representar a quantidade de indivíduos que dominam outro dentro da população, *cont* contador para percorrer toda a população e *rank* para indicar a qual ranque o indivíduo pertence. A linha 6 inicia o laço principal que percorrerá todos os indivíduos da população. Nas linhas 7 e 8 são feitas atribuições do *proxRank* para o *naoRanqueado* e o *proxRank* é esvaziado, recebendo uma lista inicialmente vazia. A linha 9 inicializa um laço interno para percorrer os indivíduos não ranqueados, logo em seguida a variável n é reinicializada com valor zero. O algoritmo segue com as linhas 11 a 16 que são utilizadas para verificar a dominância das arquiteturas (os indivíduos da população), isto é, verificar quais arquiteturas não são dominadas por nenhuma outra dentre os conjuntos de arquiteturas. Isso é feito da seguinte forma, se j domina o i , o valor de i é adicionado a *proxRank* e n é incrementado. Caso o indivíduo i que foi comparado a todos os outros da população for não dominado, o bloco das linhas 17 a 21 será executado. Esse bloco de código adiciona i à *popRanqueada*, incrementa o contador e aplica ao indivíduo qual é o seu rank. A linha 23 incrementa o ranque e o processo é repetido até que o último indivíduo tenha sido avaliado e atribuído seu *rank*. O algoritmo termina retornando a população ranqueada.

Após realizar o ranqueamento da população é necessário utilizar um método que ordene as soluções dentro do ranque. Isto é feito utilizando o operador que se baseia na distância em que cada função objetivo dos indivíduos possui dentro do ranque. O Algoritmo 7 trata exatamente disso.

Inicialmente, é importante destacar que para utilizar este operador a população já deve ter sido ranqueada, conforme apresentado no Algoritmo 6. Em seguida, uma variável n é criada para guardar quantos indivíduos existem dentro do ranque baseado no tamanho de cada ranque dentro da população ranqueada passada como parâmetro. O laço presente nas linhas 2 a 4 é responsável pela inicialização da variável *CD* de cada indivíduo com valor zero (0). Vale destacar que para cada função objetivo f, o ranque deve ser ordenado (ex., ranque de disponibilidade, custo e exergia operacional) isso é feito na linha 5. Em seguida, a variável *CD* dos extremos é preenchida com um valor elevado (infinito), e os demais seguem o cálculo da linha 10. O processo é repetido para se obter a ordenação

Algoritmo 6 ranquear (pop)

```
1: proxRank ← pop;
2: Lista naoRanquedo;
3: Lista popRanqueada;
4: n, cont ← 0;
5: rank ← 1;
6: enquanto cont < tamanho(pop) faça
7:    naoRaqueado ← proxRank;
8:    proxRank ← ∅;
9:    para i : naoRanqueado faça
10:      n ← 0;
11:      para j : naoRanqueado faça
12:         se j ≺ i então
13:            proxRank.adicionar(i);
14:            n + +;
15:         fim se
16:      fim para
17:      se n == 0 então
18:         popRanqueada.adicionar(i);
19:         cont + +;
20:         i.aplicar(rank);
21:      fim se
22:   fim para
23:   rank + +;
24: fim enquanto
25: retorno popRanqueada;
```

baseada no *CD*. Nesse caso, quanto maior o valor do *CD*, mais distante as soluções se encontram de suas vizinhas no espaço dos objetivos e, assim, maior é a sua preferência na seleção para a próxima geração da população (linha 13).

A estratégia de otimização proposta faz uso da seleção por torneio que é responsável por sortear pares de indivíduos (pais) ao acaso, comparar seus ranques e selecionar o mais bem ranqueado. Há casos em que os pais selecionados pertencem ao mesmo ranque, então será selecionado àquele com maior *CD*. O Algoritmo 8 trata da implementação da seleção torneio que tem como entrada uma população (*pop*).

A primeira linha mostra que este método recebe como parâmetro uma lista de indivíduos (população inicial, ver Algoritmo 3). Na linha 2 são criadas algumas variáveis de controle. A linha 3 inicia o laço que será executado duas vezes, fazendo a cada iteração um sorteio de 2 indivíduos aleatórios da população inicial. As linhas 4 e 5 fazem a escolha de dois números aleatórios dentro da população. As linhas 6 a 8 fazem uso de três condicionais aninhadas, afim de garantir que o índice do *pai1* não foi escolhido vazio. Depois disso,

Algoritmo 7 *Crownding Distance*(popRanqueada)

```
1: n ← tamanho(popRanqueada);
2: para Individuo ind : popRanqueada faça
3:    ind.CD ← 0;
4: fim para
5: para f : funcao faça
6:    popRanqueada ← ordenarCres(popRanqueada);
7:    popRanqueada[0].CD ← ∞;
8:    popRanqueada[n − 1].CD ← ∞;
9:    para i ← 1 até n − 1 faça
10:      popRanqueada[i].CD ← popRanqueada[i].CD + ((popRanqueada[i +
         1].pegar(f) − popRanqueada[i − 1].pegar(f))/(listIndividuo[n − 1].pegar(f) −
         popRanqueada[0].pegar(f)));
11:   fim para
12: fim para
13: popRanqueada ← ordenarCres(popRanqueada, CD);
14: retorno popRanqueada;
```

checa-se ambos valores aleatórios das linhas 4 e 5 são distinto. Em caso afirmativo, o terceiro condicionado é acionado para comparar os ranques do indivíduo da população cujo índice é o *aleatorio1* e o individuo cujo índice é o *aleatorio2*. As linhas 10 a 13 determinam qual será o índice do pai1 (vencendo aquele que possuir maior ranque). Caso não entre nos condicionais da linha 8 ou da linha 11, quer dizer que os indivíduos escolhidos através de índices aleatórios pertencem ao mesmo rank, então, as linhas 14 a 24 tratam do caso dos indivíduos pertencem ao mesmo ranque. Portanto, aquele de maior valor do *CD* será o índice do pai 1. As linhas 25 a 40 são executadas na segunda iteração deste Algoritmo, uma vez que o *indicePa1* já não é mais vazio. Então, a escolha do segundo pai será semelhante ao primeiro, onde se inicia com a verificação se os índices de cada individuo da população são distintos (linha 26). Posteriormente, verificar qual possui maior ranque (linhas 27-30), adicionando a *listPais* o indivíduo de maior ranque. Por fim, aqueles indivíduos com ranques iguais, comparar o operador *CD* (linhas 32-38), e aquele que com maior *CD* será adicionado a *listPais*. Por fim, o algoritmo termina retornando uma lista com dois indivíduos (*listPais*).

Após concluída a seleção dos pais, é preciso prosseguir com a geração dos descendentes. Para isso, é preciso se valer dos operadores genéticos de cruzamento e mutação. O Algoritmo 9 apresenta como é feita a geração de um descendente a partir da entrada de uma população. Na primeira linha, a função *selTorneiroBin* devolve para a lista *eleitos* o par de pais eleitos explicados no Algoritmo 8. Cada um deles é passado para

Algoritmo 8 seleçãoTorneio(*pop*)

```
1: Lista listPais;
2: i ← 0, aleatorio1, aleatorio2, indicePai1;
3: enquanto i < 2 faça
4:   aleatorio1 ← aleatorioInt(pop.tamanho);
5:   aleatorio2 ← aleatorioInt(pop.tamanho);
6:   se indicePai1 == ∅ então
7:     se aleatorio1 ≠ aleatorio2 então
8:       se (pop.pegar(aleatorio1).rank) < (pop.pegar(aleatorio2).rank) então
9:         listPais.adicionar(pop.pegar(aleatorio1));
10:        indicePai1 ← (aleatorio1);
11:      senão, se (pop.pegar(aleatorio2).rank) < (pop.pegar(aleatorio1).rank)
         então
12:        listPais.adicionar(pop.pegar(aleatorio2));
13:        indicePai1 ← (aleatorio2);
14:      senão
15:        se (pop.pegar(aleatorio1).CD) ≥ (pop.pegar(aleatorio2).CD) então
16:          listPais.adicionar(pop.pegar(aleatorio1));
17:          indicePai1 ← (aleatorio1);
18:        senão
19:          listPais.adicionar(pop.pegar(aleatorio2));
20:          indicePai1 ← (aleatorio2);
21:        fim se
22:      fim se
23:      i ← i + 1;
24:    fim se
25:  senão
26:    se aleatorio1 ≠ aleatorio2 & indicePai1 ≠ aleatorio2 & aleatorio1 ≠ indicePai1
       então
27:      se (pop.pegar(aleatorio1).rank) < (pop.pegar(aleatorio2).rank) então
28:        listPais.adicionar(pop.pegar(aleatorio1));
29:      senão, se (pop.pegar(aleatorio2).rank) < (pop.pegar(aleatorio1).rank)
         então
30:        listPais.adicionar(pop.pegar(aleatorio2));
31:      senão
32:        se (pop.pegar(aleatorio1).CD) ≥ (pop.pegar(aleatorio2).CD) então
33:          listPais.adicionar(pop.pegar(aleatorio1));
34:        senão
35:          listPais.adicionar(pop.pegar(aleatorio2));
36:        fim se
37:      fim se
38:      i ← i + 1;
39:    fim se
40:  fim se
41: fim enquanto
42: retorno listPais;
```

as variáveis arqPai1 e arqPai2 nas linhas 2 e 3. A linha 4 estabelece um ponto de corte que varia do segundo componente até o penúltimo que faz parte da arquitetura do indivíduo. As linhas 5 a 9 mostram como a primeira parte dos descendentes (*arqFilho1* e *arqFilho2*) é formada. A segunda parte (linhas 10 a 15) completam os descendentes, ficando no final o *arqFilho1* com a primeira metade vinda dos componentes pertencentes ao arqPai1 e a segunda parte com os respectivos componentes do arqPai2, semelhantemente ocorre com o arqFilho2. As linhas 16 a 20 tratam da mutação que ocorre a uma taxa de 10%. Caso ocorra a mutação, uma posição aleatória i é escolhida para representar o índice do arqFilho1 (ou arqFilho2, pois ambos tem a mesma arquitetura). Em seguida, é feita uma chamado ao método responsável por substituir o componente referente àquela posição, alterando assim apenas um componente do mesmo tipo da posição do índice selecionado. As linhas 21 e 22 avaliam as métricas de cada filho gerado, ambos são adicionados a uma mesma lista (*listFilhos*). O método retorna esta lista contendo dois filhos gerados por um par de pais selecionado da população (linha 25).

Algoritmo 9 operadorGenetico (pop)

```
1:  Lista eleitos ← SelTorneioBin(pop);
2:  arqPai1 ← eleitos.index(0);
3:  arqPai2 ← eleitos.index(1);
4:  pCorte ← aleatorioInt(1, tamanho(arqPai1) − 1);
5:  enquanto i < pCorte faça
6:      arqFilho1.add(arqPai1.pegar(i));
7:      arqFilho2.add(arqPai2.pegar(i)));
8:      i + +;
9:  fim enquanto
10: j ← pCorte;
11: enquanto j ≤ tamanho(arqPai2) faça
12:     arqFilho1.add(arqPai2.pegar(j));
13:     arqFilho2.add(arqPai1.pegar(j));
14:     j + +;
15: fim enquanto
16: se aleatorioInt(100) < 10 então
17:     i ← rand.nextInt(tamanho(desc1);
18:     arqFilho1.substituir(i, este.aplicarTurbulencia(arqFilho1.pegar(i)));
19:     arqFilho2.substituir(i, este.aplicarTurbulencia(arqFilho2.pegar(i)));
20: fim se
21: arqFilho1.calcularMetrica();
22: arqFilho2.calcularMetrica();
23: listFilhos.add(arqFilho1);
24: listFilhos.add(arqFilho2);
25: retorno listFilhos;
```

5.2 Estratégia baseada no MOPSO

A estratégia adotada é baseada no MOPSO (do inglês, *Multi-Objective Particle Swarm Optimization*), técnica utilizada para lidar com problemas de otimização multiobjetivo. O Algoritmo 10 apresenta uma adaptação do MOPSO que tem como entrada o tamanho da população, número de gerações, modelo da arquitetura, uma base de dados e os operadores de inerciais e constante cognitiva e social. É importante destacar que em cada indivíduo analisado da população serão otimizadas as métricas de disponibilidade, custo e exergia operacional. A codificação de cada equipamento segue o mesmo padrão adotado na Figura 24, porém há o acréscimo de mais dois atributos, além daqueles apresentados na Figura 23 que são a velocidade (*vel*) e a posição da arquitetura no espaço de busca.

Na linha 1 os indivíduos são constituídos pelo Algoritmo 3. Assim sendo, sabe-se que quando criado um indivíduo, e que pertence a um determinado padrão arquitetural, ele vai ter em sua composição a quantidade específica de equipamentos com posições pré-estabelecidas pelo modelo da arquitetura. Além de avaliar as métricas (linha 4), dois novos atributos são necessários para determinar novas posições dentro do espaço de busca. Um é o *pBest* que guarda a melhor posição obtida por um indivíduo (linha 5), o outro é a velocidade (linha 6) e serve para que o indivíduo saia de sua posição em direção a outra no espaço de busca. Vale ressaltar que os indivíduos resultantes das operações realizada dentro da estratégia proposta sempre são formados por equipamentos mapeados no dicionário da base de dados.

A linha 8 faz o ranqueamento da população inicial e atribui à lista de indivíduos (*REP*). Em sequência, na linha 9, o método *crownDist* é aplicada para o ranque 1 do REP, ou seja, é criada uma lista com apenas indivíduos do melhor ranque e ordenada pelo *CD*, segundo o Algoritmo 7. A linha 10 cria um *gBest* que é o melhor indivíduo global de cada iteração do algoritmo. A linha 11 inicia o *loop* principal da estratégia que consiste em pegar o mais bem posicionado via *CD* e atribuí-lo ao gBest (linha 12). Em seguida, na linha 13, foi adotada uma metodologia que pudesse conter a velocidade ao longo das iterações, uma forma de não ter o peso de inércia dinâmico. No laço, a partir da linha 14, são realizados os cálculos para atualizar a velocidade (linha 15). Esse cálculo faz uso da Equação 2.13 e, após atualizada a velocidade, é feita a atualização da posição na linha 16 (Equação 2.14), a partir da distância euclidiana para encontrar os componentes mais

Algoritmo 10 Estratégia de Otimização Baseado no MOPSO

Entrada: $N, g, arqModelo, dicBase, w\_min, w\_max, c1, c2$

1: $Lista\ populacao \leftarrow gerarPopInicial(arquitetura, dicBase, N)$;
2: $Lista\ popOtima$;
3: **para** Individuo ind : populacao **faça**
4: ind.$avaliarMetricas()$;
5: $ind.salvarPbest()$;
6: $ind.vel \leftarrow Aleatorio.proxValor()$;
7: **fim para**
8: $Arraylist < Individuo > REP \leftarrow ranquear(populacao)$;
9: $popOtima \leftarrow crownDist(pegarRank(1, REP))$;
10: $Individuo\ gBest$;
11: **para** $(i \leftarrow 1$ **até** $g)$ **faça**
12: $gBest \leftarrow pegarMelhor(popOtima)$;
13: $w \leftarrow w\_max - (\frac{w\_max - w\_min}{g}) \cdot i$;
14: **para** Individuo ind : populacao **faça**
15: $ind.atualizarVel(w, c1, c2, gBest)$;
16: $ind.atualizarPosicao(dicBase)$;
17: $ind.avaliarMetricas()$;
18: $ind.atualizarPbest()$;
19: **fim para**
20: $REP \leftarrow ranquear(populacao)$;
21: **para** $ind : pegarRank(1, REP)$ **faça**
22: $popOtima.adicionar(ind)$;
23: **fim para**
24: $popOtima \leftarrow crownDist(pegarRank(1, ranquear(popOtima))$;
25: **fim para**
26: **retorno** $popOtima$;

próximos contidos na base. A linha 17 avalia as métricas do indivíduo nesta sua nova posição e em seguida atualiza o *pBest*. Após cada indivíduo ter "caminhado"no espaço de busca, eles são ranqueados e passados para a lista *REP* (linha 20). As linhas 21 a 23 fazem o ranqueamento de *REP* e adicionam à população otimizada, que no fim é ordenada pelo *crownDist*, retornando assim a população ótima (*popOtima*).

O Algoritmo 11 demonstra que, caso haja a turbulência, um equipamento da arquitetura é sorteado para ser substituído por outro do mesmo tipo. A escolha do substituto é feita aleatoriamente através de um mapeamento na base de dados. Esse método realiza a modificação em um componente do indivíduo sem afetar no modelo da arquitetura.

A linha 1 escolhe um valor inteiro positivo aleatório de 1 a 100. Em seguida, a linha 2 verifica se o valor de *num* é menor que 10, se isso ocorrer o algoritmo segue para

Algoritmo 11 aplicarTurbulencia(comp, listaCompsTipo)

```
1: num ← aleatorioInt(100);
2: se (num < 10) então
3:    Componente novoComp ← comp;
4:    enquanto comp == novoComp faça
5:       novoComp ← sortearComp(listaCompsTipo);
6:    fim enquanto
7: fim se
8: retorno novoComp;
```

linha 3 onde um novo componente é criado. As linhas 4 a 6 são responsáveis por sortear o componente distinto daquele que deve ser modificado, isto evita repetição de componente, permitindo de fato a diversidade. O método finaliza com o retorno de um novo componente (linha 8).

Algoritmo 12 atualizarPosicao(dicBase)

```
1: Lista novoCompsArqt;
2: Lista tipo;
3: Componenete novoComp;
4: para comp : compsArqt faça
5:    tipo ← dicBase.pegar(comp.tipo);
6:    novosValores ← aplicarVelocidade(comp);
7:    novoComp ← obterCompCorresp(tipo, novosValores);
8:    novoComp ← aplicarTurbulencia(novoComp, tipo);
9:    novoComp.inserirDadosArqt(comp.pegarDadosArqt());
10:   novaCompArqt.adicionar(novoComp);
11: fim para
12: compsArqt ← novoCompsArqt;
```

A seguir, é apresentado o Algoritmo 12 que atualiza a posição do indivíduo. A linha 1 instância uma lista de componentes. A linha 2 cria uma outra lista de tipo de componentes. A linha 3 cria uma variável componente sem atribuir valor a ela. A linha 4 inicia o laço que percorrerá todos os componentes de uma dada arquitetura. A variável *tipo* armazena o tipo do componente (linha 5). A linha 6 recebe o valor do "passo" para onde o componente deveria de fato seguir. O método *obterCompCorrep* se valer da distância euclidiana para encontrar o equipamento do mesmo tipo que mais se aproxima da posição do indivíduo (linha 7). Sendo assim, o novo componente seria aquele com menor distância euclidiana. A linha 8 aplica a turbulência (ver Algoritmo 11). A linha 9 chama o método *inserirDadosArqt* para passar ao novo componente o modelo de arquitetura. Dessa forma o componente vai estabelecer as conexões dentro do indivíduo. A linha 10 adiciona este

novo componente dentro da lista *novoCompArqt* e o processo é repetido até que todos os componentes sejam readequandos. A linha 12 indica que a lista *compsArqt* recebe a lista *novaCompsArqt*, que representa a nova posição do indivíduo no espaço de busca.

5.3 Validação das Estratégias

A validação das estratégias propostas é feita comparando o conjunto solução da otimização baseada nas técnicas evolucionárias (NSGA-II ou MOPSO) com a curva de Pareto ótima (Algoritmo de Força Bruta). Este algoritmo compara todos os *trade-offs*, isto é, a partir de uma base de dados de equipamentos, e dada uma arquitetura como entrada, são geradas todas as combinações de equipamentos que compõem esta arquitetura e para cada combinação são avaliadas as métricas alvo deste trabalho (disponibilidade, custo e exergia operacional).

O Algoritmo 13 apresenta as etapas do força bruta que recebe como entrada uma arquitetura e uma base de dados. Na linha 1 a base é mapeada. A linha 2 estabelece a quantidade máxima de combinações possíveis. A linha 3 a variável *cont* é um contador que vai percorrer uma lista de componentes da arquitetura. Um array *vMax* é criado na linha 4 que vai armazenar para cada componente de uma dada arquitetura a quantidade máxima possível de variações, já o array da linha 5 é um contador dessas variações. A linha 6 inicia o laço para calcular a quantidade máxima de combinações possíveis por componente dentro da arquitetura, isso é feito verificando a quantidade de componentes por tipo (linhas 7 a 9). A quantidade máxima por tipo encontrada é armazenada na variável *tamChave* que por sua vez é passada na linha 10 para *tamanhoFinal* que computa a combinação de todos os tipos de componentes. A linha 11 cada posição do *vMax* recebe a quantidade máxima de combinações por equipamento. No segundo laço (linha 16) que será executado de 1 até o máximo de combinações possíveis (*tamanhoFinal*) uma variável booleana é criada (linha17) e um *indiceVert* (linha 18) que recebe o tamanho da arquitetura. As linhas 19 a 27 são para verificar se todos os componentes foram percorridos, fazendo isso da esquerda para direita. As linhas 28 e 29 criam o indivíduo e define uma arquitetura para ele. Nas linhas 30 a 34 o indivíduo criado (*ind*) é preenchido por componentes extraídos do dicionário de componentes (*dicCOmp*). A linha 35 incrementa o contado de combinações, seguindo para linha 36 o novo indivíduo tem suas métricas avaliadas e na linha 37 o indivíduo é

Algoritmo 13 Força Bruta (arquitetura, baseDados)

```
1: dicComp ← mapearBase(baseDados);
2: tamanhoFinal ← 1;
3: cont ← 0;
4: Array vMAX ← Array[Arquitetura.tamanho)];
5: Array v ← Array[arquitetura.tamanho)];
6: para comp : arquitetura.componentes faça
7:    tipo ← comp.tipo;
8:    auxComp ← dicComp.pegar(tipo);
9:    tamChave ← tamanho(auxComp);
10:   tamanhoFinal ← tamanhoFinal · tamChave;
11:   vMAX[cont] ← tamChave;
12:   v[cont] ← 1;
13:   cont + +;
14: fim para
15: Lista todaCombinacoes;
16: para j ← 1 até tamanhoFinal faça
17:   verficador ← VERDADEIRO;
18:   indiceVert ← arquitetura.tamanho − 1;
19:   enquanto verificador == VERDADEIRO faça
20:     se v[indiceVert] ≤ vMAX[indiceVert] então
21:       verificar ← FALSO;
22:     senão
23:       v[indiceVert] ← 1;
24:       v[indiceVert − 1] + +;
25:     fim se
26:     indiceVert − −;
27:   fim enquanto
28:   Individuo ind;
29:   ind.definirArquitetura(arqModelo);
30:   para i ← 0 até (arquitetura.tamanho − 1) faça
31:     listComp ← dicComp.pegar(ind.arquitetura.indice(i).tipo);
32:     comp ← listComp(v[i] − 1);
33:     ind.preencherArquitetura(i, comp);
34:   fim para
35:   v[arquitetura.tamanho − 1] + +;
36:   ind.avaliarMetricas();
37:   todaCombinacoes.adicionar(ind);
38: fim para
39: FB ← crowDist(raquear(todaCombinacoes, 1));
40: retorno FB;
```

adicionado numa lista chamada *todasCombinacoes*. O processo repete até todos os ranques. A linha 39, por fim, realiza o ranqueamento da lista *todaCombinacao* salvando em *FB* a fronteira de Pareto ótimo ordenada pelo *crowding distance*.

6 Estudos de Caso

Este capítulo apresenta três estudos de caso. O primeiro estudo utiliza a estratégia de otimização multiobjetivo baseada no NSGA-II para trazer um conjunto solução com máxima disponibilidade, minimizando ao mesmo tempo custo e exergia operacional das infraestruturas elétricas de *data centers*. O segundo estudo de caso tem como objetivo otimizar os mesmos critérios do primeiro, contudo com a estratégia de otimização multiobjetivo baseada no MOPSO. Ambos estudos de caso utilizam o *Pareto Subset* (*ps*) como métrica de comparação de algoritmos multiobjetivo (VELDHUIZEN, 2000). O *ps* é a quantidade de elementos da curva Pareto ótimo (gerada pelo força bruta) que dominam os elementos da curva de Pareto aproximado. O terceiro estudo de caso faz uso dos modelos TIERs (UPTIME, 2021) e amplia a base de dados, este cenário analisa o comportamento das estratégias propostas nos dois primeiros estudos de caso. Como há neste último estudo de caso um aumento exponencial de combinações, faz com que a utilização do algoritmo de força bruta seja computacionalmente inviável.

6.1 Estudo de caso I

Essa seção apresenta um estudo de caso que busca otimizar as métricas de disponibilidade, custo e exergia nas infraestruturas elétricas de *data centers* utilizando a estratégia apresentado no Algoritmo 5. Além disso, essa seção também apresenta os modelos adotados e análise estatística do algoritmo de otimização proposto em relação ao algoritmo de força bruta que retorna o conjunto solução ótimo. Para cada indivíduo gerado no força bruta foram avaliadas suas funções objetivo de acordo com o Algoritmo 13. Vale frisar que foram feitas normalizações dos dados das funções, uma vez que possuem ordem de grandeza distintas. A Equação 6.1 apresenta como é feita a normalização das funções objetivo.

$$f(x)_{normal} = \frac{f(x) - f(x)_{min}}{f(x)_{max} - f(x)_{min}} \tag{6.1}$$

A Tabela 2 apresenta os parâmetros de entrada (MTTF, MTTR, eficiência energética (EE) e custo de aquisição) adotados com base em (GUIMARÃES; PEREIRA, 2020) e

(MELO et al., 2021). Intervalos (min e máx.) foram definidos a partir dos valores bases dessa tabela para criação da base de equipamentos utilizados para a otimização. Sendo assim, para a eficiência energética, foi definida uma variação de 15%; para o MTTF foi adotada a variação de 50%; e considerando o custo dos equipamentos se variou 15%. Esse estudo adotou o MTTR de cada equipamento constante (8h). Ao todo foram gerados 10 equipamentos de cada tipo, que se combinados na arquitetura mais simples (A1) geram 10000 combinações possíveis. A estratégia multiobjetivo proposta e o algoritmo de força bruta foram aplicados nas 6 arquiteturas. O objetivo é maximizar a disponibilidade, diminuir a exergia operacional e o custo. O computador utilizado para rodar os modelos teve a seguinte configuração Win10, Intel i5 1.80 GHz, 8GB RAM, 1TB HD.

Tabela 2 – Valores de intervalo de referência para os dispositivos

Equipamento	MTTF(hs)	Preço(US$)	EE(%)
AC	[2619, 6205]	[11724,74, 18517,98]	[90,35, 98,9]
Generator	[1120 - 2700]	[60679.75, 74628.79]	[23,83, 25,89]
UPS	[25000, 75000]	[43558,31, 72597,19]	[90,535, 99,9]
STS	[24038, 72114]	[2323,11, 3871,85]	[94,525, 99,9]
ATS	[306749, 646790]	[608,11, 968,85]	[95,25, 99,9]
Subpanel	[152000, 456000]	[580,78, 967,96]	[94,905, 99,9]
SDT	[141290,5, 423871,5]	[1597,14, 2661,90]	[93,575, 99,9]
Powerstrip	[199576,60, 313455,75]	[681,45, 898,27]	[86,575, 97,575]

A ferramenta utilizada para a modelagem foi a *Stars* (LEONARDO; CALLOU, 2021), que possibilita a avaliação integrada dos modelos RBD, SPN e fluxo de energia (EFM). A estratégia de otimização evolucinária foi implementado na linguagem Java e integrada no ferramental *Stars* como funcionalidade de otimização de modelos. É importante destacar que a ferramenta se comunica com o *Mercury* (OLIVEIRA et al., 2017), para realizar a avaliação das métricas de cada indivíduo modelado de acordo com a Figura 25.

6.1.1 Arquiteturas

A Figura 25 apresenta as seis arquiteturas das infraestruturas elétricas de *data center* adotadas nesses estudos (TIA-942, 2005). Essas arquiteturas são compostas por: fontes de alimentação sem interrupção (UPS), transformador (SDT), chaveador de transferência estática (STS), quadro de energia (subpanel) e réguas de energia elétrica (*powerstrip*).

A Figura 25(a) ilustra a arquitetura A1 que é a base para as outras arquiteturas, apesar de não ter nenhum equipamento redundante. Esta arquitetura inicial é composta por UPS, SDT, SubPanel e Powerstrip. A Figura 25(b) mostra a arquitetura A2, que é semelhante a arquitetura A1 com a adição de uma replicação para o UPS. A Figura 25(c) ilustra a arquitetura A3, com os mesmos componentes presentes na arquitetura A1, com a diferença na replicação do UPS e SDT, como também a adição de uma chave de transferência estática (STS). A Figura 25(d) apresenta a arquitetura A4, onde temos os mesmos componentes presentes na arquitetura A1, com SubPanel adicional. A Figura 25(e) ilustra a arquitetura A5, que corresponde a arquitetura A1 com todos os equipamentos replicados. Vale ressaltar que essa arquitetura não tem a necessidade do equipamento STS. A Figura 25(f) apresenta a arquitetura A6, que corresponde a arquitetura A5 com adição de um ups (UPS3) em *cold standby* aos UPS1 e UPS2, além da adição de mais dois STS que comutam para o UPS de reserva na ocorrência de uma falha.

6.1.2 Modelos

Nesta seção são apresentados os modelos propostos. Cada arquitetura teve sua modelagem elaborada a partir do modelo da visão de alto nível da ferramenta *Stars*, conforme a Figura 26. A modelagem adotada integra os modelos formais. O modelo em RBD, usado no cálculo da disponibilidade. O modelo em EFM utilizados para obtenção do custo e exergia operacional. O SPN para criação das redundâncias *coldstandby*. Para todas as arquiteturas (A1, ..., A6) foi aplicada a estratégia apresentada no Algoritmo 5 baseado no NSGA-II e o Algoritmo 13 de Força Bruta .

Em seguida, assim como explicado na Subseção 4.3, ocorre a conversão desses modelos em linguagem de *script* compatível com o Mercury, o método apresentado no Algoritmo 4 é chamado para avaliar as métricas a serem otimizadas (disponibilidade, custo e exergia operacional). A Figura 27 apresenta os modelos RBD proposto para representar

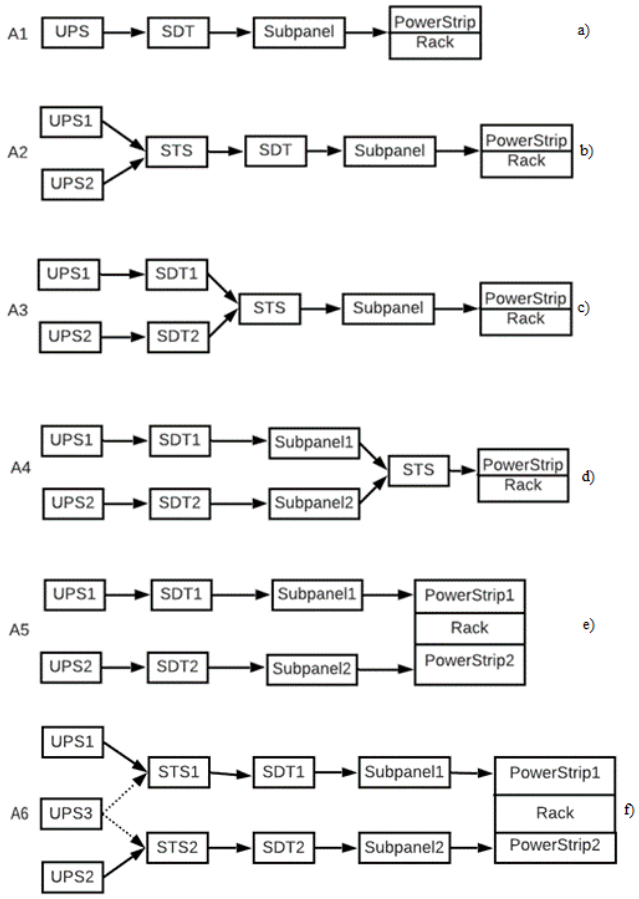


Figura 25 – Arquiteturas elétricas de *data center*

as arquiteturas (A1,...,A6), detalhe para a arquitetura A6 com o bloco B1, resultante da disponbilidade do modelo SPN *coldstadby* apresentado na Figura 16.

A Figura 28 apresenta o modelo EFM proposto para representar as arquiteturas (A1,...,A6). Esse modelo representa o fluxo de energia elétrica entre os equipamentos. A exergia operacional e o custo são computados, segundo o Algoritmo 4 e os parâmetros de entrada de cada equipamento que compõe sua respectiva arquitetura foi extraído da Tabela 2. Vale salientar que a arquitetura A6 existem pesos nas arestas que partem dos UPS, isto se dá ao fato da modelagem *coldstanby*, que é apresentado na Figura

6.1.3 Resultados

O algoritmo adotado, além de receber uma arquitetura (Figura 25), teve, por questões de limitação computacional, uma população inicial de 50 indivíduos, 100 gerações e foi executada 10 vezes (a partir de 10 execuções o algoritmo não apresentava mudança estatísticamente relevante nas funções objetivo da sua população), por se tratar de uma

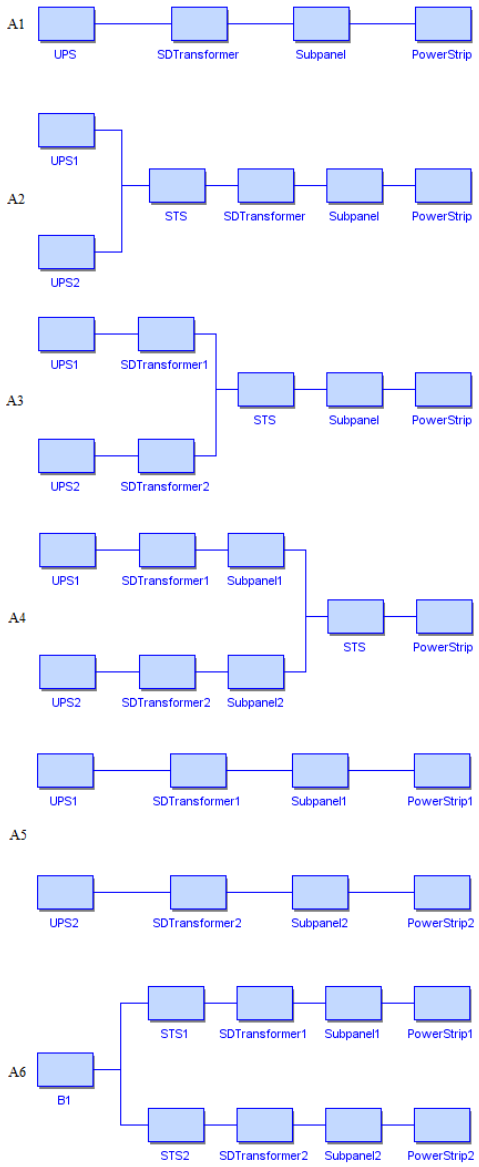


Figura 26 – Modelos das Arquiteturas na Visão da ferramenta *Stars*

solução estocástica. Os resultados obtidos no gráfico apresentado na Figura 29, onde é possível comparar a estratégia multiobjetivo proposta baseada no NSGA-II com o algoritmo da força bruta da arquitetura A6. Pelo gráfico é possível observar que as soluções geradas pela estratégia de multiobjetivo proposta se aproximam bastante das soluções ótimas (fronteira de Pareto ótimo). Analisando os resultados obtidos pode ser observado que a disponibilidade da solução por meio de força bruta e da solução otimizada foram bem próximas, apresentando uma diferença dentro do intervalo de confiança de 95%. Além disso, devemos observar que o tempo de execução entre as duas abordagens apresentadas são bem distintos.

É importante destacar que foram computados os tempos de execução demandados

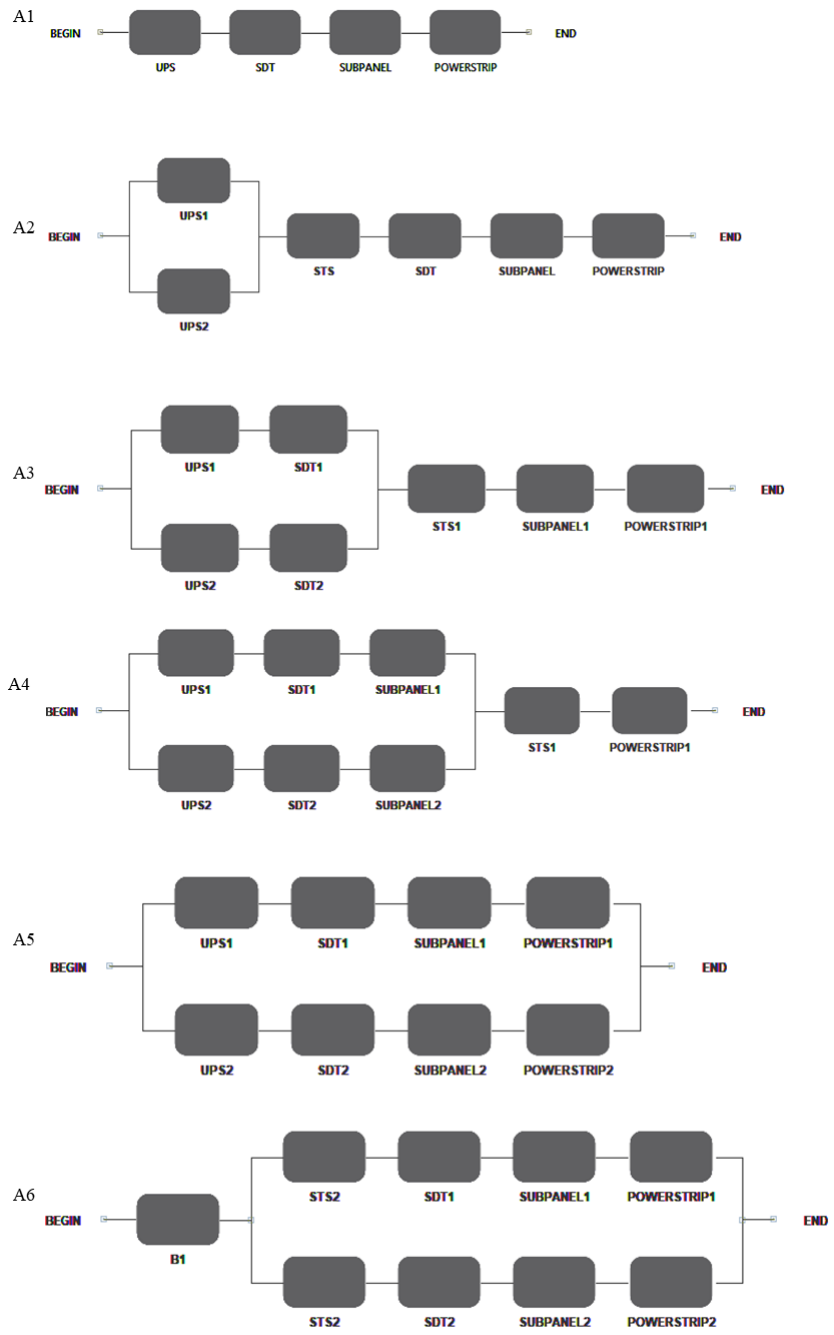


Figura 27 – Modelos Arquiteturas em RBD

para cada abordagem. Por exemplo, assumindo a arquitetura A1, o tempo com o algoritmo de otimização proposto demandou em média aproximadamente 11s em comparação com os 621s necessários utilizando a força bruta. Já para a arquitetura A6, o tempo demandado médio foi de 734s para o algoritmo de otimização proposto versus 432.896s no algoritmo de busca exaustiva. É considerável a diferença entre o tempo gasto pelos algoritmos, principalmente quando são arquiteturas mais complexas (589 vezes). Enquanto o algoritmo

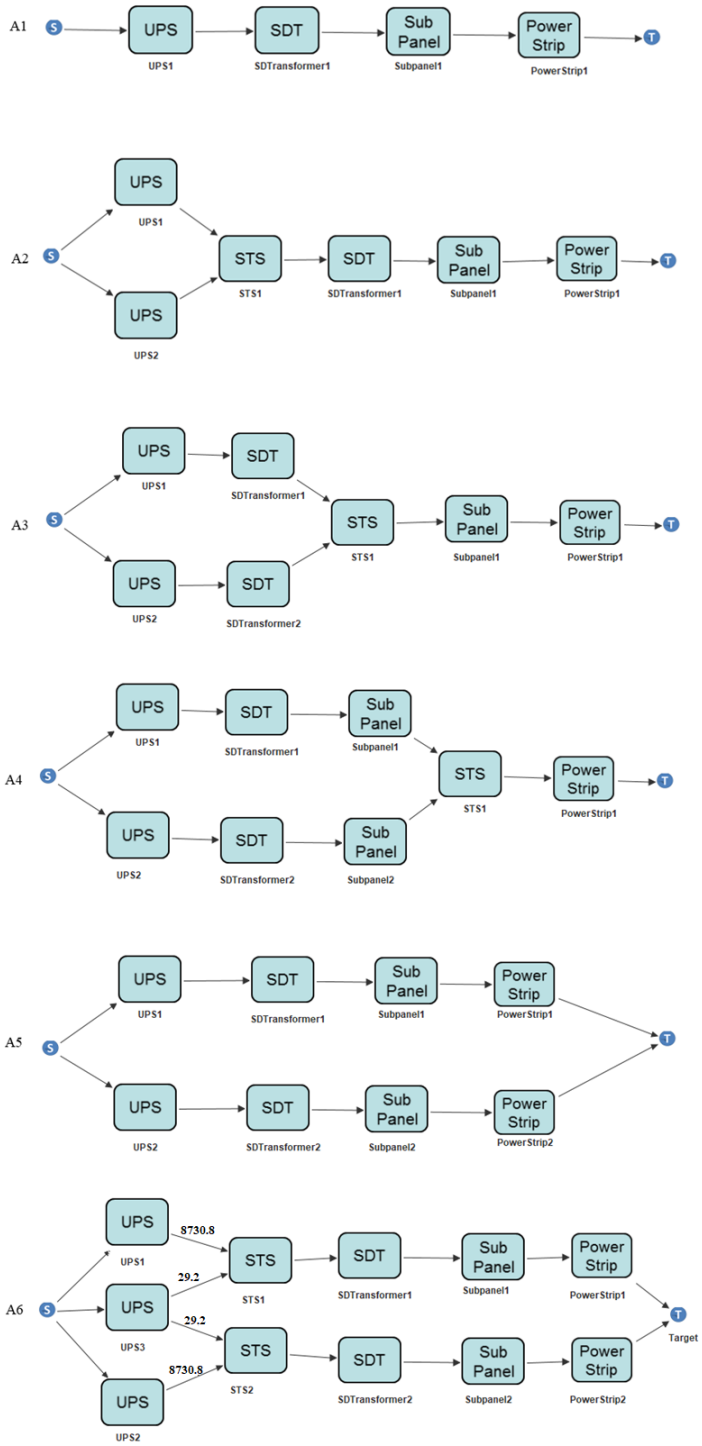


Figura 28 – Modelos em EFM das arquiteutras propostas

baseado no NSGA-II tem crescimento linear do tempo, na abordagem utilizando força bruta o tempo de execução tende a crescer de modo exponencial, de acordo com a arquitetura analisada.

Nos gráficos da Figura 30 é possível comparar as funções por três perspectivas. Na primeira (Figura 30 (a)) são apresentados os resultados da otimização na perspectiva

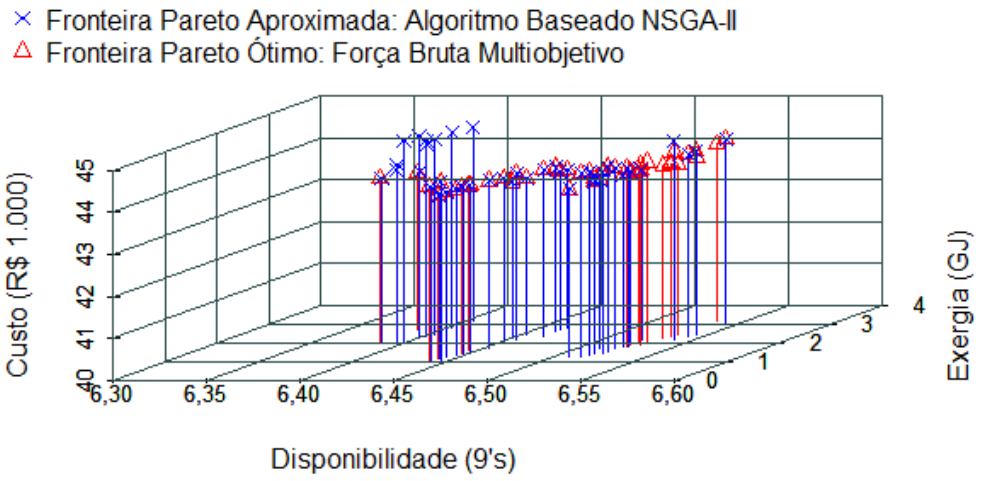


Figura 29 – Gráfico conjunto Pareto aproximado Algoritmo Proposto x conjunto Pareto ótimo Força Bruta. (Arquitetura A6)

Disponibilidade x Custo. Nele é possível observar, tomando por base as médias de cada função objetivo, levando em consideração o *ps* a máxima diferença entre o conjunto solução e a curva de Pareto ótimo é de 1,42%. A segunda (Figura 30 (b)) traz a perspectiva da disponibilidade em função da exergia, onde a diferença máxima entre as curvas ficam em 2,5%. A Figura 30 (c) traz a terceira perspectiva (Custo x Exergia), onde a diferença máxima entre as curvas ficam em 3,1%.

A Figura 31 apresenta o gráfico que permite comparar as média de cada métrica otimizada pelo força bruta com àquelas otimizadas utilizando a estratégia baseada no NSGA-II. É importante destacar que também foi realizado o cálculo do desvio padrão para a arquitetura A6.

A Tabela 3 apresenta os resultados obtidos ao realizar a otimização das seis arquiteturas. Nessa tabela é possível inferir de forma quantitativa a convergência do conjunto solução da última geração contendo os 50 indivíduos para cada uma das seis arquiteturas do problema comparando a estratégia baseada no NSGA-II com a fronteira ótima de Pareto obtida pelo Força Bruta. Os dados computados levam em consideração a média, o desvio padrão e o intervalo de confiança de 95%. Vale salientar que para cada arquitetura (A1,...,A6) foi realizado o cálculo da média e desvio padrão de cada métrica otimizada de suas populações finais. É possível constatar, a partir dos dados obtidos, que os intervalos de confiança nas médias das métricas do algoritmo proposto baseado no

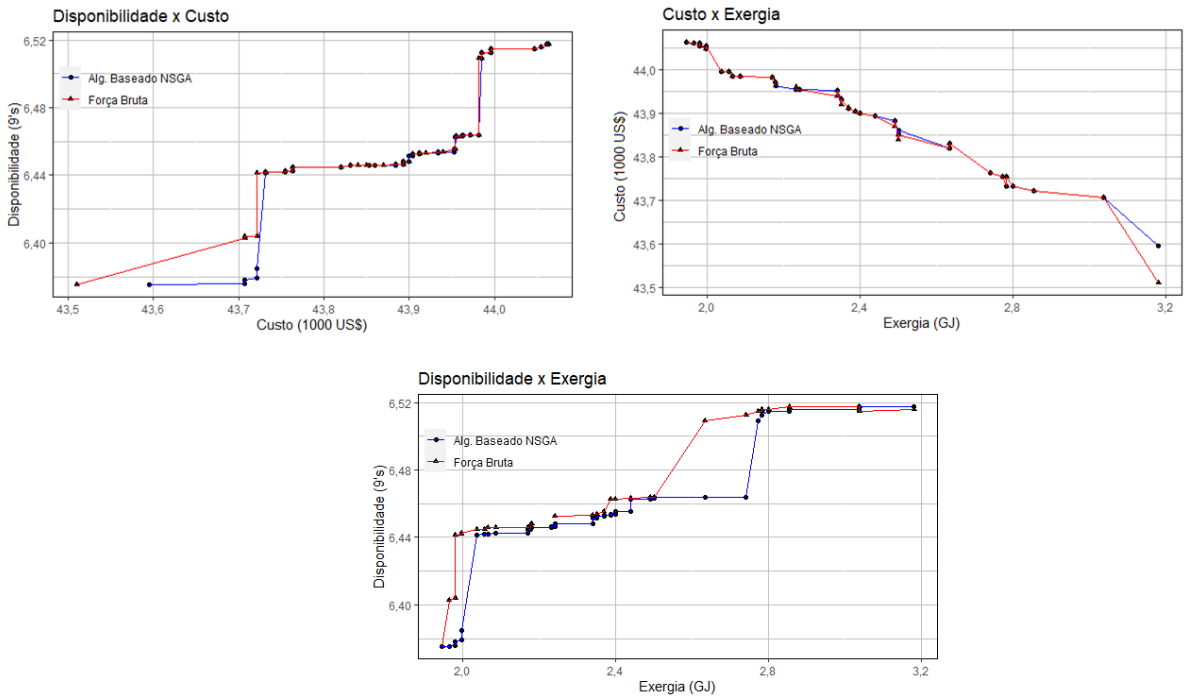
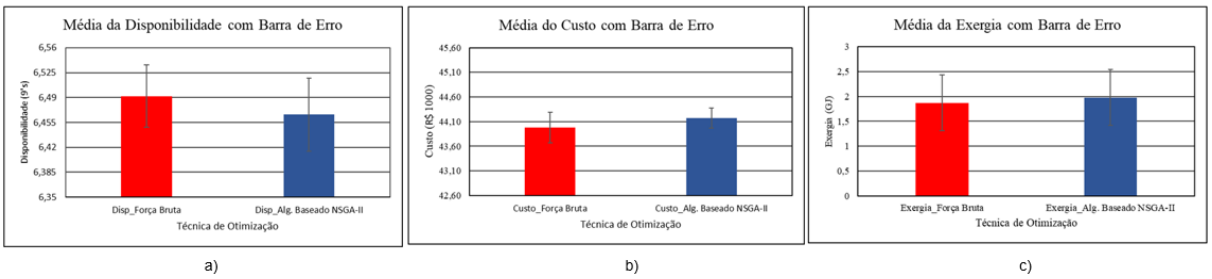


Figura 30 – Gráficos Comparativos em Três Perspectivas (arq. A6)



a) b) c)

Figura 31 – Gráficos Média com Barra de Erro por Funções Objetivo (arq. A6)

NSGA-II sobrescrevem as médias do Algoritmo de Força Bruta. Dessa forma, pode-se concluir que os resultados obtidos acabam por validar a estratégia proposta, tendo em vista a proximidade com as soluções ótimas de Pareto.

6.2 Estudo de caso II

Este segundo estudo de caso difere do primeiro, pois adota a estratégia de otimização baseada no MOPSO (ver Algoritmo 10), afim de maximizar a disponibilidade, minimizar custo e exergia operacional das infraestruturas elétricas de *data centers*. Para que haja validação da estratégia, será apresentada uma comparação entre o conjunto solução da última geração da estratégia proposta e o conjunto solução ótimo obtido pelo Algoritmo de Força Bruta. Vale destaque para o método 12 que foi utilizado para cada componente

Tabela 3 – Comparativo entre Força Bruta e Estratégia Baseada no NSGA-II

		Força Bruta				Estratégia baseada no NSGA-II			
Arquitetutura	Métrica	Média	Desv. Pad.	IC	Tempo (s)	Média	Desv. Pad.	IC	Tempo (s)
A1	Disp.(9's)	3,155	0,0217	[3,1492; 3,1622]		3,105	0,221	[3,038; 3,172]	
	Exergia (GJ)	1,129	0,322	[1,032; 1,225]	621	1,198	0,341	[1,096; 1,301]	11
	Custo (US$)	19515,14	904,87	[194879,60; 19542,33]		20745,04	1230,91	[19514,13; 21975,95]	
A2	Disp.(9's)	3,643	0,024	[3,635; 3,650]		3,6123	0,112	[3,578; 3,646]	
	Exergia (GJ)	1,427	0,407	[1,304; 1,549]	1367	1,515	0,432	[1,385; 1,645]	20
	Custo (US$)	20118,70	932,86	[20090,68; 20146,73]		20732,82	1008,12	[20107,88; 21357,76]	
A3	Disp.(9's)	3,880	0,026	[3,872; 3,888]		3,764	0,134	[3,723; 3,884]	
	Exergia (GJ)	1,502	0,428	[1,373; 1,631]	23184	1,4782	0,421	[1,351; 1,604]	117
	Custo (US$)	22108,46	102,51	[22077,67; 22139,26]		23336,11	1412,84	[22011,65; 24660,58]	
A4	Disp.(9's)	4,078	0,028	[4,070; 4,087]		4,040	0,426	[3,912; 4,168]	
	Exergia (GJ)	1,564	0,446	[1,430; 1,698]	147332	1,548	0,441	[1,415; 1,680]	237
	Custo (US$)	23829,64	108,59	[23487,36; 23552,61]		23336,11	966,52	[23539,27; 24120,02]	
A5	Disp.(9's)	5,934	0,040	[5,922; 5,946]		5,841	0,299	[5,741; 5,941]	
	Exergia (GJ)	1,758	0,501	[1,607; 1,908]	285396	1,781	0,508	[1,628; 1,933]	415
	Custo (US$)	33432,12	155,13	[33385,52; 33478,73]		33972,28	765,65	[33742,25; 34202,31]	
A6	Disp.(9's)	6,492	0,043	[6,478; 6,505]		6,482	0,062	[6,464; 6,501]	
	Exergia (GJ)	1,975	0,563	[1,806; 2,1449]	432896	1,774	0,811	[1,530; 2,018]	734
	Custo (US$)	44080,53	204,33	[44019,13; 44141,94]		44387,23	1099,68	[44056,84; 44717,60]	

Legenda: Disp.(9's) é a Disponibilidade (em números de noves); Desv. Pad. é o desvio padrão considerando os 50 individuos da população otimizada após 100 gerações; [IC] - [Intervalo de Confiança com nível de significância de 95%]; Tempo (s) - Tempo de Execução da Estratégia de Otimização.

avaliado pela estratégia no espaço de busca (base de dados dos equipamentos), a fim de indicar qual seu substituto baseado na próximidade de seus atributos (distância euclidiana). Ressalta-se ainda, que as arquiteturas (ver Figura 25) e os modelos em Figura 27 e Figura 28 adotados no primeiro estudo são os mesmos para esse estudo.

6.2.1 Resultados

A estratégia de otimização multiobjetivo foi aplicada às 6 arquiteturas partindo da mesma população inicial gerada no primeiro estudo de caso. O objetivo continua em ser maximizar a disponibilidade, diminuir a exergia operacional e o custo. O computador utilizado para rodar os modelos teve a seguinte as mesmas configurações do estudo de caso anterior.

O algoritmo adotado teve como entrada uma população inicial de 50 indivíduos, 100 gerações e foi executado 10 vezes, por se tratar de uma solução estocástica. O número de execuções do algoritmo levou em consideração um intervalo de confiança de 95%,

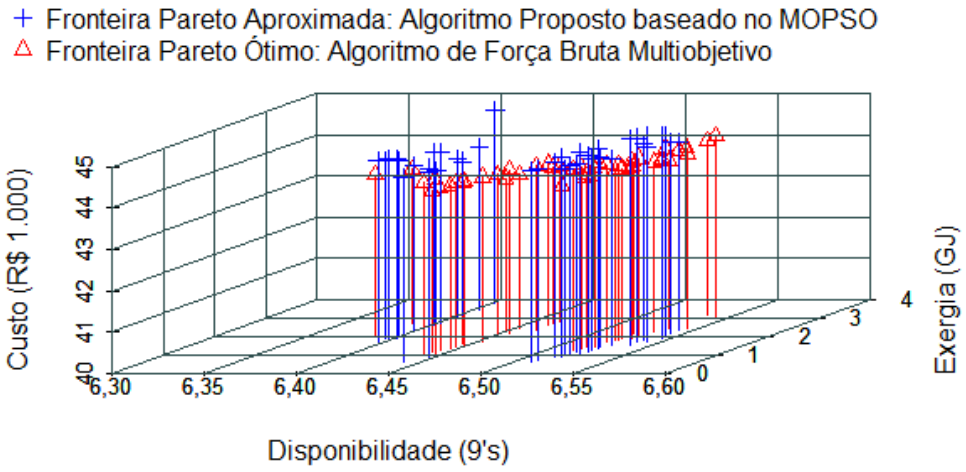


Figura 32 – Gráfico Pareto Ótimo: Força Bruta x Algoritmo Proposto Baseado no MOPSO (arq. A6)

revelou não haver diferença, estatisticamente relevante, após a décima execução. Os resultados obtidos aplicando a estratégia de otimização na arquitetura A6 são mostrados na Figura 32. Analisando os resultados apresentados na Tabela 4 pode ser observado que a disponibilidade da solução por meio de força bruta e da solução otimizada apresentaram uma diferença inferior a 4%, no pior caso. Foram computados os tempos de execução demandados para cada abordagem. Por exemplo, assumindo a arquitetura A1, o algoritmo de otimização baseado no MOPSO demandou em média aproximadamente 20s, tempo bem inferior em comparação com os 621s necessários utilizando a força bruta. Já para a arquitetura A6, o tempo demandado médio foi de 862s para o algoritmo de otimização baseado no MOPSO contra 432.869s no algoritmo de busca exaustiva. Quando, portanto, são comparados os tempos de execução das duas técnicas para otimizar a arquitetura mais complexa, conclui-se que aquela baseada no MOPSO leva mais tempo que a estratégia do primeiro estudo de caso, em cerca de 17%.

A Figura 33 apresenta o gráfico que permite comparar as médias de cada métrica otimizada pelo força bruta com aquelas otimizadas utilizando a estratégia baseada no MOPSO. Esses dados foram computados para a arquitetura A6, uma vez que se trata da arquitetura mais complexa deste estudo. Pode-se inferir, através da sobreposição das médias pela barra de erro, que a estratégia tem sua similaridade com os resultados obtidos pelo algoritmo de força bruta.

Nos gráficos da Figura 34 é possível comparar as funções por três perspectivas.

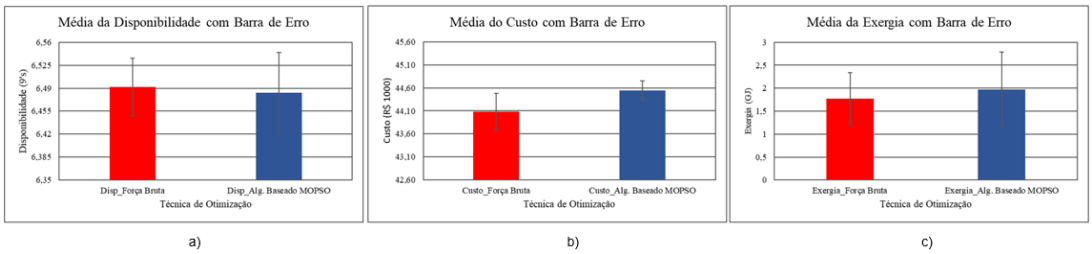


Figura 33 – Gráficos Média com Barra de Erro por Funções Objetivo (arq. A6)

Na primeira (Figura 34 (a)) são apresentados os resultados da otimização na perspectiva Disponibilidade x Custo. Nele os pontos dispostos representam os indivíduos otimizados pelo Algoritmo Força Bruta e os otimizados pela estratégia baseada no MOPSO. Há 50 indivíduos de cada técnica, sendo que alguns deles estão sobrescritos. Levando em consideração o *ps*, pode-se constatar que a máxima diferença entre o conjunto solução e a curva de Pareto ótimo é de 2,38%, isto é feito comparando as médias da disponibilidade otimizada pela estratégia baseada no MOPSO com o força bruta. A segunda (Figura 34 (b)) traz a perspectiva da disponibilidade em função da exergia, onde a diferença máxima entre as curvas ficam em 3%. A Figura 34 (c) traz a terceira perspectiva (Exergia x Custo), onde a diferença máxima entre as curvas ficam em 3,64%.

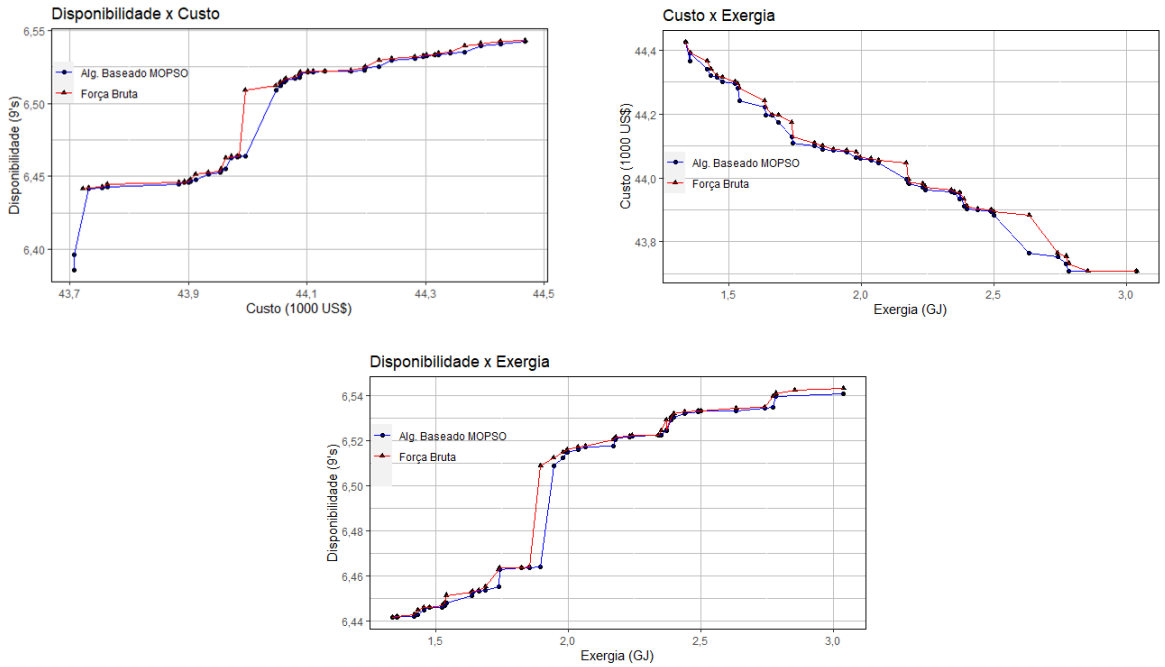


Figura 34 – Gráficos Comparativos em Três Perspectivas (arq. A6)

A Tabela 4 apresenta os resultados dos valores das médias de cada métrica analisada.

Além disso, essa tabela apresenta também os desvios padrões (Desv. Pad.) e o intervalo de confiança (IC) da última geração da arquitetura A6 do algoritmo proposto e também do força bruta (que traz o conjunto solução ótimo). Dessa forma, pode-se concluir que os resultados obtidos acabam por validar a estratégia proposta, tendo em vista que as médias de suas funções objetivo tem proximidade com as soluções ótimas de Pareto dentro do intervalo de confiança de 95%.

Tabela 4 – Comparativo entre Força Bruta e Estratégia Baseada no MOPSO

		Força Bruta				Estratégia baseada no MOPSO			
Arquitetutura	Métrica	Média	Desv. Pad.	IC	Tempo (s)	Média	Desv. Pad.	IC	Tempo (s)
A1	Disp.(9's)	3,155	0,0217	[3,1492; 3,1622]		3,123	0,1211	[3,087; 3,159]	
	Exergia (GJ)	1,129	0,322	[1,032; 1,225]	621	1,162	0,331	[1,063; 1,262]	23
	Custo (US$)	19515,14	904,87	[194879,60; 19542,33]		86771,45	18170,79	[19475,79; 21154,73]	
A2	Disp.(9's)	3,643	0,024	[3,635; 3,650]		3,624	0,063	[3,605; 3,644]	
	Exergia (GJ)	1,427	0,407	[1,304; 1,549]	1367	1,469	0,419	[1,344; 1,595]	29
	Custo (US$)	20118,70	932,86	[20090,68; 20146,73]		20343,57	797,11	[20104,09; 20583,05]	
A3	Disp.(9's)	3,880	0,026	[3,872; 3,888]		3,850	0,101	[3,819; 3,880]	
	Exergia (GJ)	1,502	0,428	[1,373; 1,631]	23184	1,442	0,411	[1,318; 1,565]	137
	Custo (US$)	22108,46	102,51	[22077,67; 22139,26]		23023,76	106,75	[22091,68; 23955,83]	
A4	Disp.(9's)	4,078	0,028	[4,070; 4,087]		4,058	0,066	[4,038; 4,078]	
	Exergia (GJ)	1,564	0,446	[1,430; 1,698]	147332	1,517	0,432	[1,387; 1,648]	375
	Custo (US$)	23519,98	108,59	[23487,36; 23552,61]		23790,98	814,02	[23546,42; 24035,54]	
A5	Disp.(9's)	5,934	0,040	[5,922; 5,946]		5,899	0,131	[5,860; 5,938]	
	Exergia (GJ)	1,758	0,501	[1,607; 1,908]	285396	1,705	0,486	[1,559; 1,851]	647
	Custo (US$)	33432,12	155,13	[33385,52; 33478,73]		33776,45	958,46	[33488,50; 34064,41]	
A6	Disp.(9's)	6,492	0,043	[6,478; 6,505]		6,492	0,052	[6,478; 6,535]	
	Exergia (GJ)	1,975	0,563	[1,806; 2,1449]	432896	2,051	0,735	[1,946; 2,122]	862
	Custo (US$)	44080,53	204,33	[44019,13; 44141,94]		44350,58	4260,45	[44070,53; 44819,75]	

Legenda: Disp.(9's) é a Disponibilidade (em números de noves); Desv. Pad. é o desvio padrão considerando os 50 individuos da população otimizada após 100 gerações; [IC] - [Intervalo de Confiança com nível de significância de 95%]; Tempo (s) - Tempo de Execução da Estratégia de Otimização.

6.3 Estudo de Caso III

Esse estudo difere dos anteriores realizados por realizar otimização em arquiteturas segundo o padrão de classificação baseado em TIERs (UPTIME, 2021). A palavra *tier* tem origem inglesa e significa "camada" ou "nível". Por isso, na classificação TIER, existem diferentes níveis de certificação: o TIER I, TIER II, TIER III e TIER IV. Tratam-se, portanto, de arquiteturas que possuem mais equipamentos o que torna inviável o uso do

força bruta devido a grande quantidade de combinações. Serão analisados, então, as duas estratégias evolucinárias propostas e validadas pelos outros estudos de caso (Estudos de Caso I e II).

6.3.1 TIER I

O TIER I é o mais simples dos níveis da classificação (N). Utilizado basicamente na operação de servidores que seguem critérios básicos de conformidade, sem apresentar componentes redundantes. Nesta classificação Tier existe previsão de um nível mínimo de distribuição de carga com pouca ou nenhuma redundância. Neste caso uma falha ou uma parada para manutenção pode ocasionar a interrupção parcial ou total da operação.

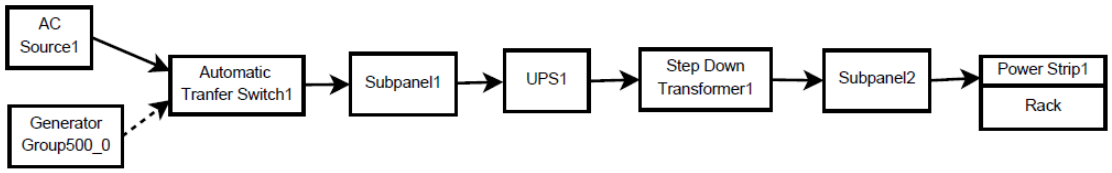


Figura 35 – Arquitetura TIER I

Definida a arquitetura TIER I, o modelo de visão de alto nível é apresentado na Figura 36. Este modelo foi elaborado na ferramenta *Stars*. É importante destacar que, a partir deste modelo, os demais são obtidos, através da conversão da linguagem de *script*, tal qual foi apresentado na Seção 4.3. O passo seguinte corresponde a chamada ao método *solve* que avalia a métrica de disponibilidade do modelo.

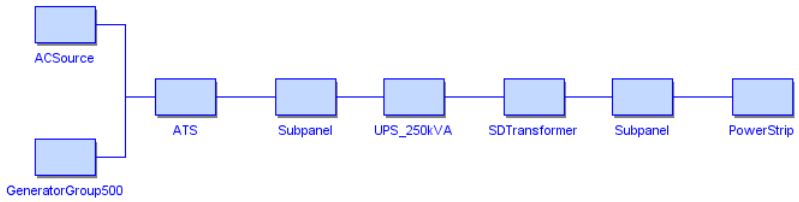


Figura 36 – Modelo Visão Alto Nível. *Stars* - TIER I

O único mecanismo de tolerância a falha deste *tier* é a presença de um gerador como fonte de energia. O gerador está em redundância do tipo *cold standby* cujo modelo SPN está representado na Figura 37. Neste modelo AC_ON representa que o fornecimento da concessionária de energia está ativo. Caso haja falha no fornecimento de energia pela concessionária, o lugar AC_OFF recebe um *token* e a transição temporalizada (T_atv)

ficará ativa e após seu *delay* acionará o gerador (o lugar GER_ ON recebe um *token*). A disponibilidade, para este modelo, é obtida pela expressão: P{(#AC_ ON = 1) OR (#GEN_ON = 1)}. A disponibilidade do modelo *cold standby* representado pelo Figura 37 é passada para o bloco AC_GEN do modelo RBD apresentado na Figura 38.

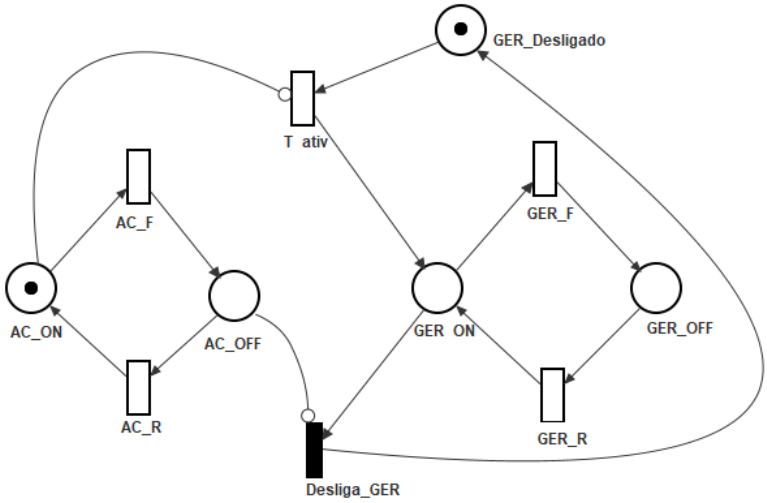


Figura 37 – Modelo SPN *cold standby* AC/GER TIER I

Figura 38 – Modelo RBD TIER I

A disponibilidade do sistema avaliado é passada como parâmetro para avaliar as outras duas métricas (exergia operacional e custo) ambas obtidas através do modelo EFM da Figura 39. O leitor deve observar os pesos das arestas da concessionária (AC), principal fonte de energia e o gerador (GEN). Como o gerador só é acionado na falha deste fornecimento de AC, então o seu peso na aresta é muito menor, uma vez que o tempo de *downtime* da concessionária (MTTF) em 8760 horas é de apenas 2 horas.

6.3.2 TIER II

A arquitetura do TIER II tem complexidade maior que a TIER anterior, mesmo continuando com caminho único de fornecimento de energia. Nesta arquitetura é necessário prover módulos UPS (*Uninterruptible Power Supply*) redundantes para N+1, isto é, além

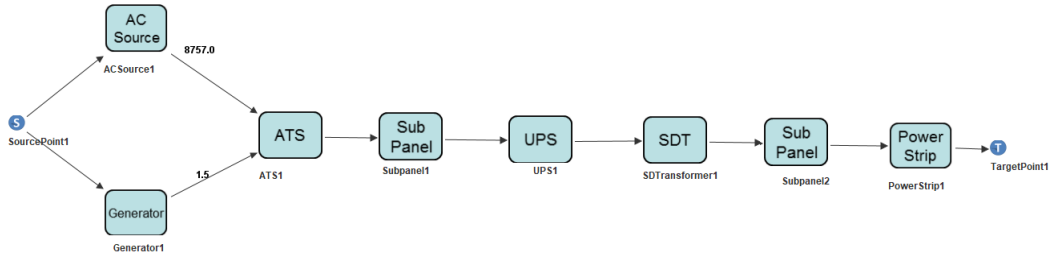


Figura 39 – Modelo EFM TIER I

do principal um em redundância *hot standby* e outro em *cold standby*. Além disso, um sistema de gerador elétrico para suprir a carga, não é necessário, todavia, redundância na fonte de energia (outro ACSourse). A principal diferença para o TIER I está na maior redundância de alguns componentes que reduz, mas não elimina, a possibilidade de interrupção por conta de uma falha de um equipamento não redundante. A arquitetura TIER II é representado pela Figura 40.

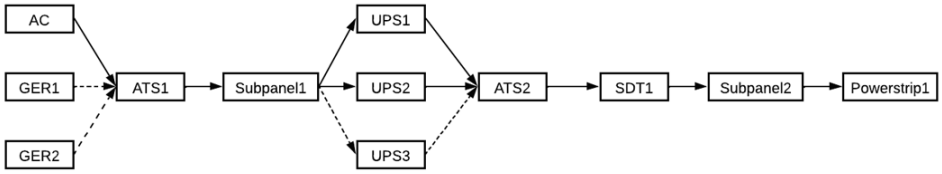


Figura 40 – Arquitetura TIER II

O modelo de visão de alto nível foi construído utilizando o ferrametal *Stars*, e é apresentado na Figura 41. Esse ferramental traduz o modelo de alto nível para os formalismos RBD, SPN e EFM no padrão adotado pela linguagem de *script* do Mercury. Então, trata-se de uma modelagem integrada, mas não uma visão isolada, de um único formalismo. Ela inclui os formalismo em um único *script*, facilitando, por exemplo, a codificação de quais componentes estão em redundância *cold standby*.

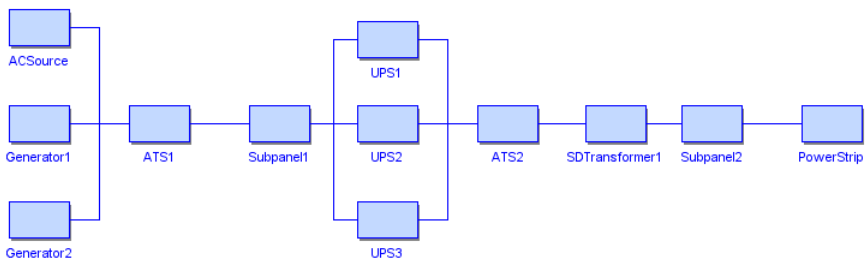


Figura 41 – Modelo Visão Alto Nível. *Stars* - TIER II

É importante frisar, que dois modelos SPN de redundâncias são construídos. Um

no nível de fonte (AC/GER1/GER2) e outro no nível dos UPS (UPS1/UPS2/UPS3). O primeiro modelo está representado na Figura 42. Nesse modelo, o AC está em redundância *cold standby* com os geradores (GER1/GER2). O lugar AC_ON inicia com um *token*, apresentado que a concessionária está fornecendo energia e a transição AC_F está ativa. Caso haja falha no fornecimento de energia, a transição AC_F consome o *token* que segue para o lugar AC_OFF indicando a falha da concessionária. O primeiro gerador (GER1) é acionado após o tempo de ativação (T_ativGER1) consumir o *token* do lugar GER1_D e transferi-lo para o lugar GER1_ON, definindo que o primeiro gerador está ativo. Quando o primeiro gerador falhar, ou seja, a transição GER1_F consumir o *token* do lugar GER1_ON e gerar um token no lugar GER1_OFF. Dessa forma, tanto o lugar AC_OFF quanto o lugar GER1_OFF terão um *token*, a transição T_ativGER2 será habilitada e após seu disparo o *token* de GER2_D é transferido para GER2_ON acionando com isso o segundo gerador (GER2). A disponibilidade do modelo é calculada por: $P\{(\#AC\_ON = 1)\ OR\ (\#GER1\_ON = 1)\ OR\ (\#GER2\_ON = 1)\}$.

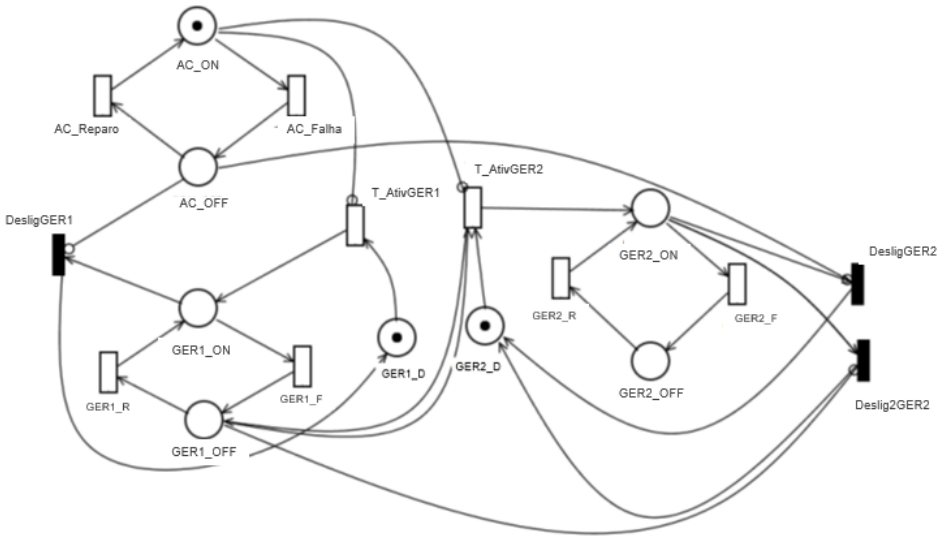


Figura 42 – Modelo SPN *cold standby* AC/GER1/GER2

O segundo modelo, representado pela Figura 43, tem como objetivo demonstrar o comportamento do sistema com a redundância UPS1/UPS2/UPS3. Nesse caso, o equipamento UPS3 só será ativado quando o UPS1 ou UPS2 estiverem em situação de falha. Nesse modelo, inicialmente com os UPS1 e UPS2 estão ativos, apresentando uma redunância *hot standby*, caso a transição UPS1_F e UPS2_F sejam disparadas o *token* será transferido para o lugar UPS1_OFF e UPS2_OFF respectivamente, indicando falha nos

UPS1 e UPS2. Isso faz com que a transição T_ativ que é responsável pela ativação do UPS3 fique habilitada. Após o tempo de ativação do UPS3, a ficha de Controle é consumida e o lugar UPS3_ON, inicialmente vazio, receberá um *token*. A disponibilidade desse modelo é computada por: $P\{(\#UPS1\_ON = 1)OR(\#UPS2\_ON = 1)OR(\#UPS3\_ON = 1)\}$.

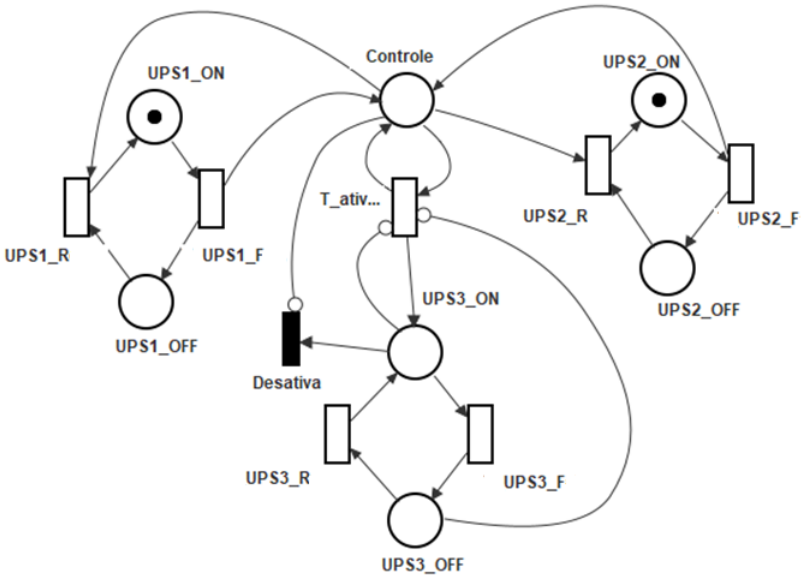


Figura 43 – Modelo SPN *cold standby* UP1/UP2/UP3

A disponibilidade obtida pela avaliação do modelo da Figura 42 é passada para um bloco RBD AC_GER. Por sua vez, a disponibilidade do modelo da Figura 43 é atribuída ao bloco UPS_CONJ, ambos são conectados aos demais que compõe o modelo RBD do TIER II, conforme ilustra a Figura 44.

Figura 44 – Modelo RBD TIER II

Ainda que, apresentados separadamente (para fins didáticos), vale salientar que todos os modelos estão representados em um *script* único contendo os três modelos formais (SPN, RBD e EFM). Todavia, a métrica da disponibilidade do modelo é necessária para avaliar o custo e exergia operacional. Isso é possível através do modelo EFM, conforme apresentado na Figura 45

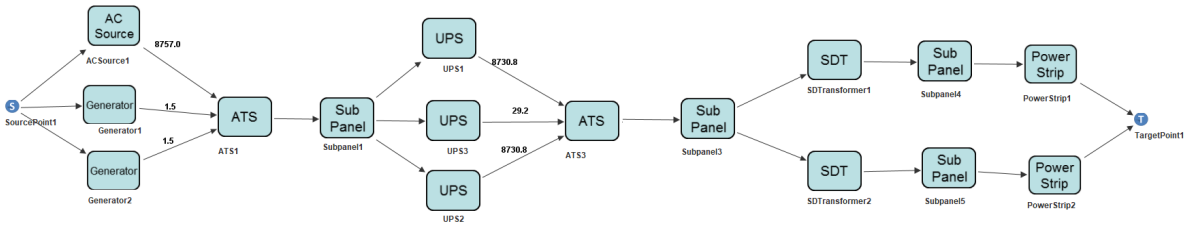


Figura 45 – Modelo EFM TIER II

6.3.3 TIER III

A arquitetura TIER III deve ser atendida por no mínimo duas concessionárias de energia. Nesta classificação TIER, a redundância é N+1, que possui os equipamentos principais e mais dois redundantes. Além disso, um novo caminho é adicionado, através de uma nova fonte de fornecimente de energia. Esta inclusão de uma fonte redundante permite que a manutenção seja realizada no primeiro caminho, sem interrupção do serviço. A Figura 46 ilustra a arquitetura TIER III.

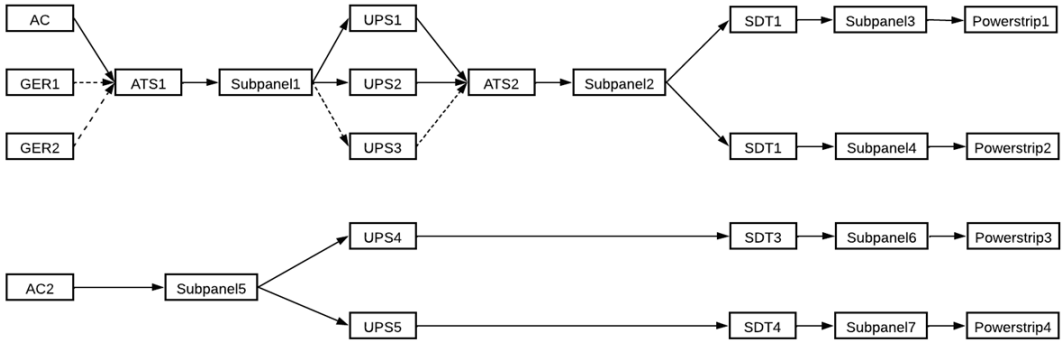


Figura 46 – Arquitetura TIER III

O modelo de alto nível feito no *Stars* pode ser visto na Figura 47. O modelo em RBD da arquitetura resultante da conversão do modelo de visão de alto nível é apresentado na Figura 48. Vale destacar que este modelo utiliza a modelagem híbrida nas redundâncias AC/GER1/GER2 que avalia a disponibilidade em SPN e atribui ao bloco AC_GER, segundo a Figura 42. O mesmo ocorre com a redundância UPS1/UPS2/UPS2, onde a disponibilidade é passada para o bloco UPS_CONJ de acordo com a Figura 43.

A partir do resultado da disponibilidade é possível, então, avaliar as outras duas métricas alvo deste estudo. A exergia operacional e o custo são obtidos a partir da avaliação do modelo EFM da Figura 49.

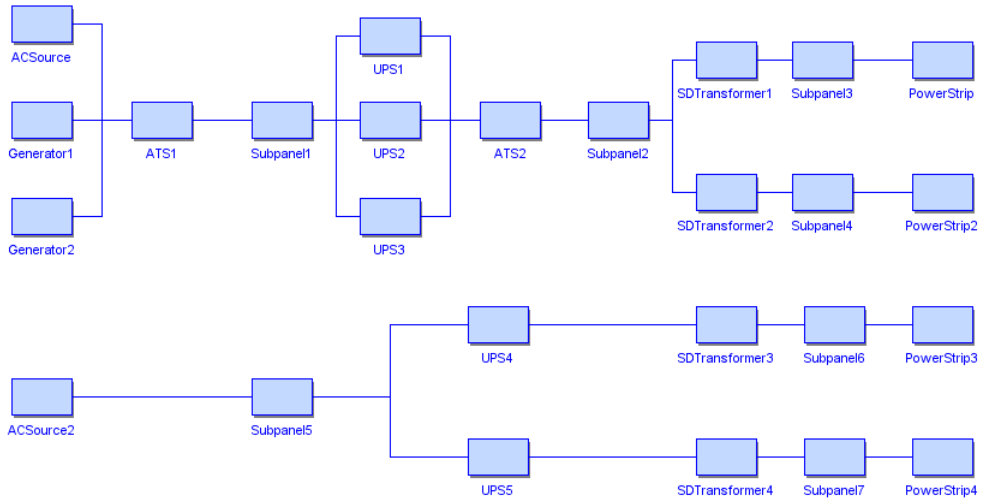


Figura 47 – Modelo Visão Alto Nível. *Stars* - TIER III

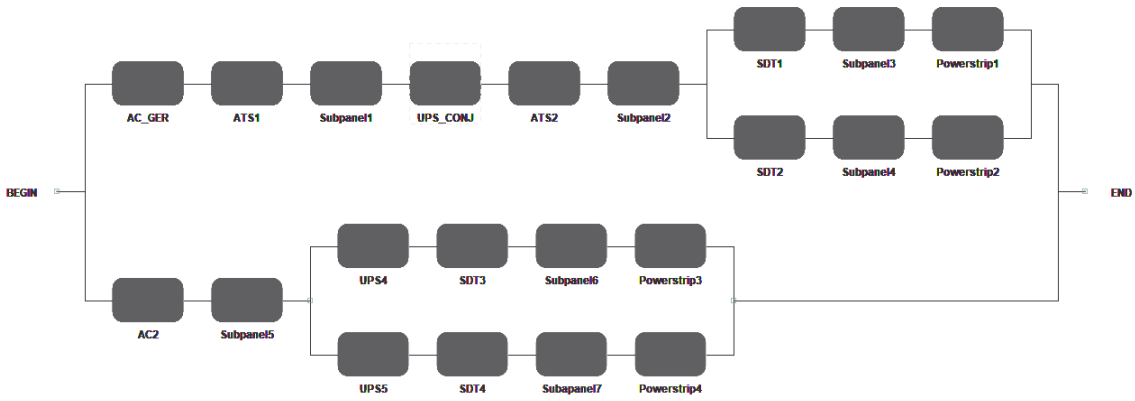


Figura 48 – Modelo RBD TIER III

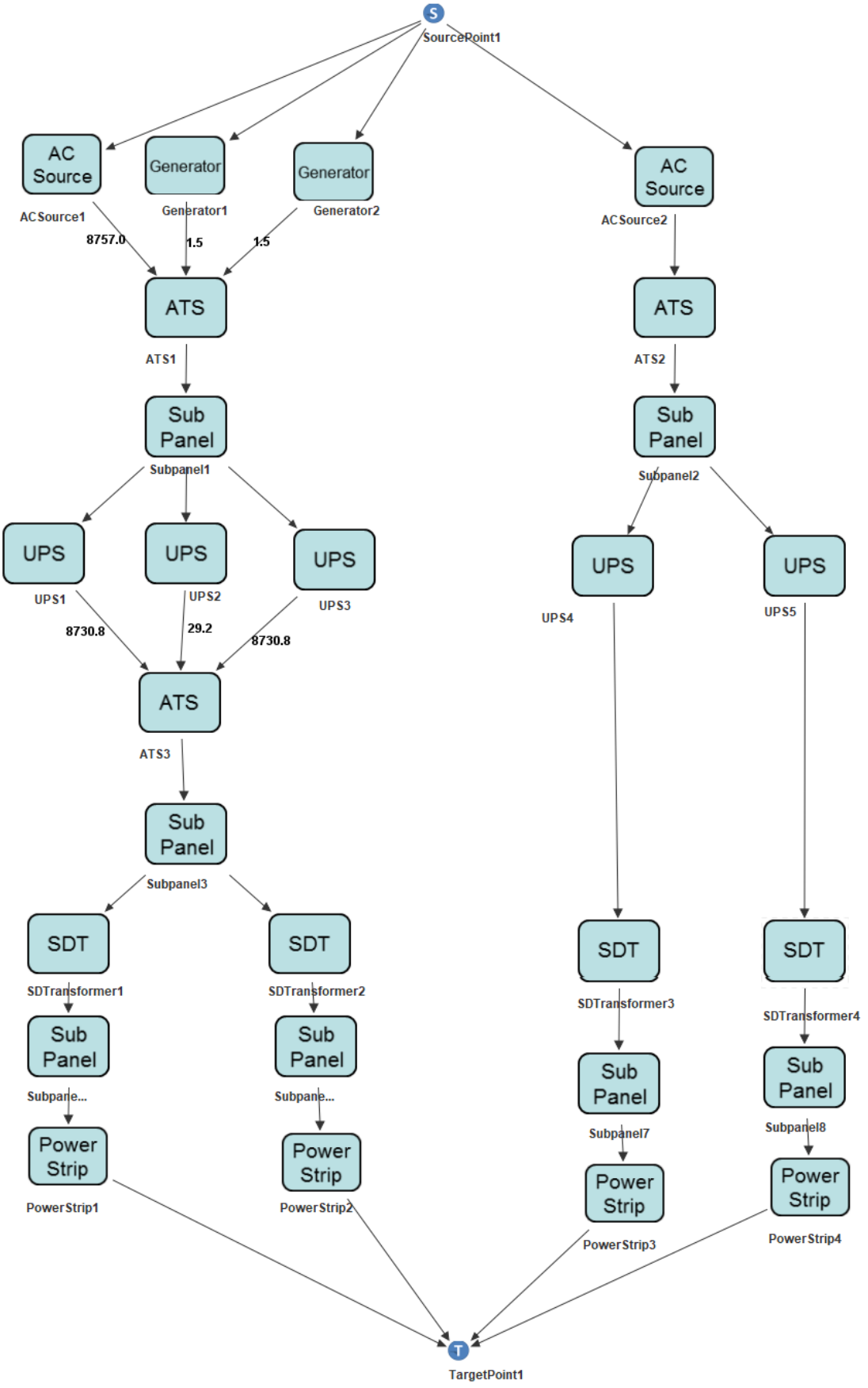


Figura 49 – Modelo EFM TIER III

6.3.4 TIER IV

A arquitetura TIER IV é do tipo 2(N+1), onde deve haver duas fontes de alimentação distintas para o sistema elétrico do *data center*, além de, no mínimo, um gerador para cada uma delas. Esta arquitetura possui dois caminhos de distribuição de energia, sendo que os dispositivos com função semelhante têm separação física. Isso implica dizer que mesmo em caso de falha ou manutenção de uma unidade, módulo, caminho ou sistema, as operações não são interrompidas. A Figura 50 apresenta a arquitetura do TIER IV.

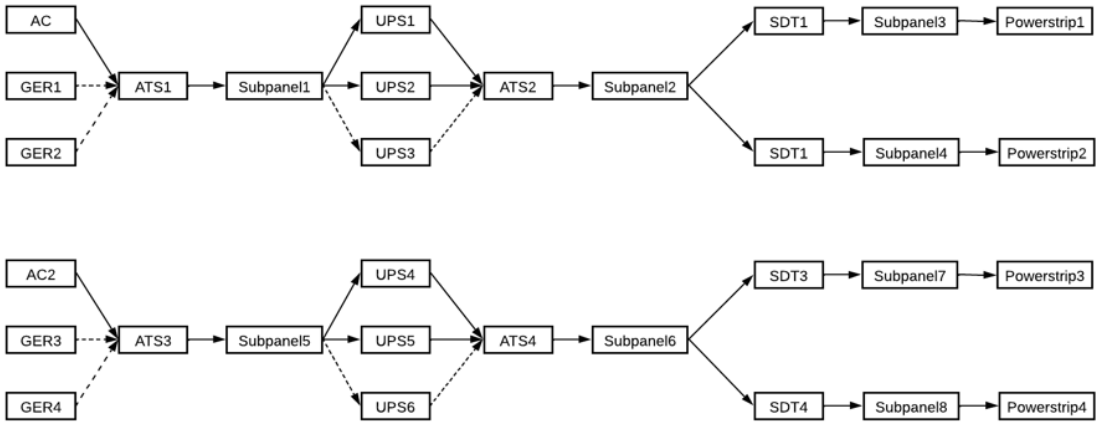


Figura 50 – Arquitetura TIER IV

Como mencionado nos modelos dos TIERs anteriores, a modelagem foi elaborada no *Stars*, tendo em vista a integração com os modelos formais. Isso permite a conversão um único modelo de *script* contendo RBD, SPN e EFM. O modelo TIER IV é mostrado na Figura 51. A representação em RBD da arquitetura é apresentado na Figura 52. A disponibilidade dos modelos SPN utilizados para representar as redundâncias são obtidas pela avaliação daquele modelo apresentado na Figura 42, sendo assim a disponibilidade AC1/GER1/GER2 é passada para o bloco AC_GER1 e a disponibilidade de AC2/GER3/GER4 é atribuida ao bloco AC_GER2. Semelhantemente como ocorre na Figura 43, os UPS1/UPS2/UPS3 têm sua disponibilidade passada para o bloco RBD UPS_CONJ1 e os UPS4/UPS5/UPS6 têm atribuida sua disponibilidade ao bloco UPS_CONJ2.

Obtida a disponibilidade do sistema é possível avaliar a exergia operacional e o custo através do modelo EFM da Figura 53. Esta avaliação é feita, tal qual foi explicado na Seção 4.3 (ver Figura 21).

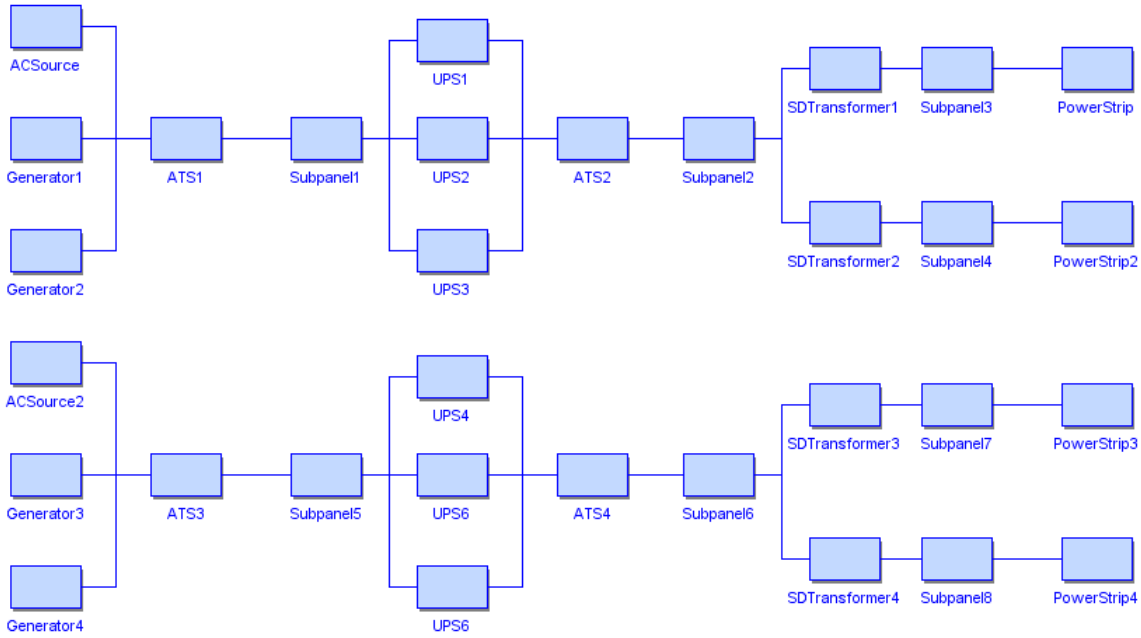


Figura 51 – Modelo Visão Alto Nível. *Stars* - TIER IV

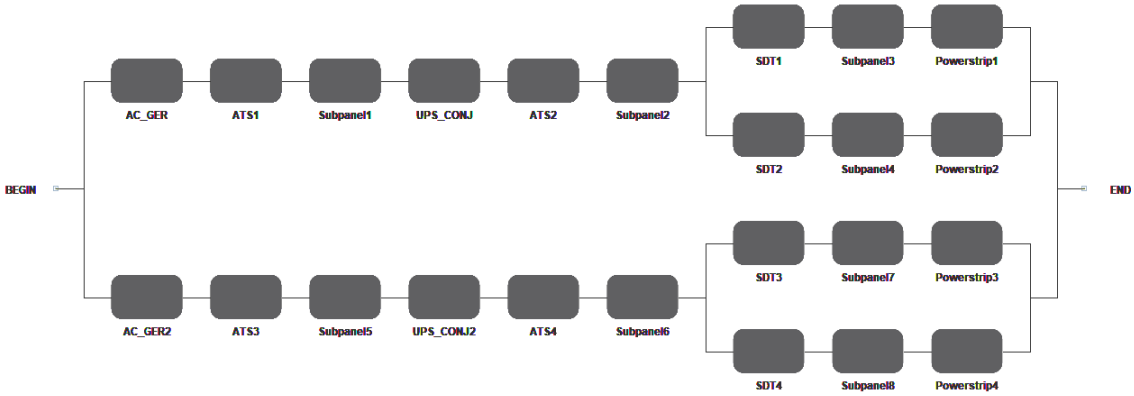


Figura 52 – Modelo RBD TIER IV

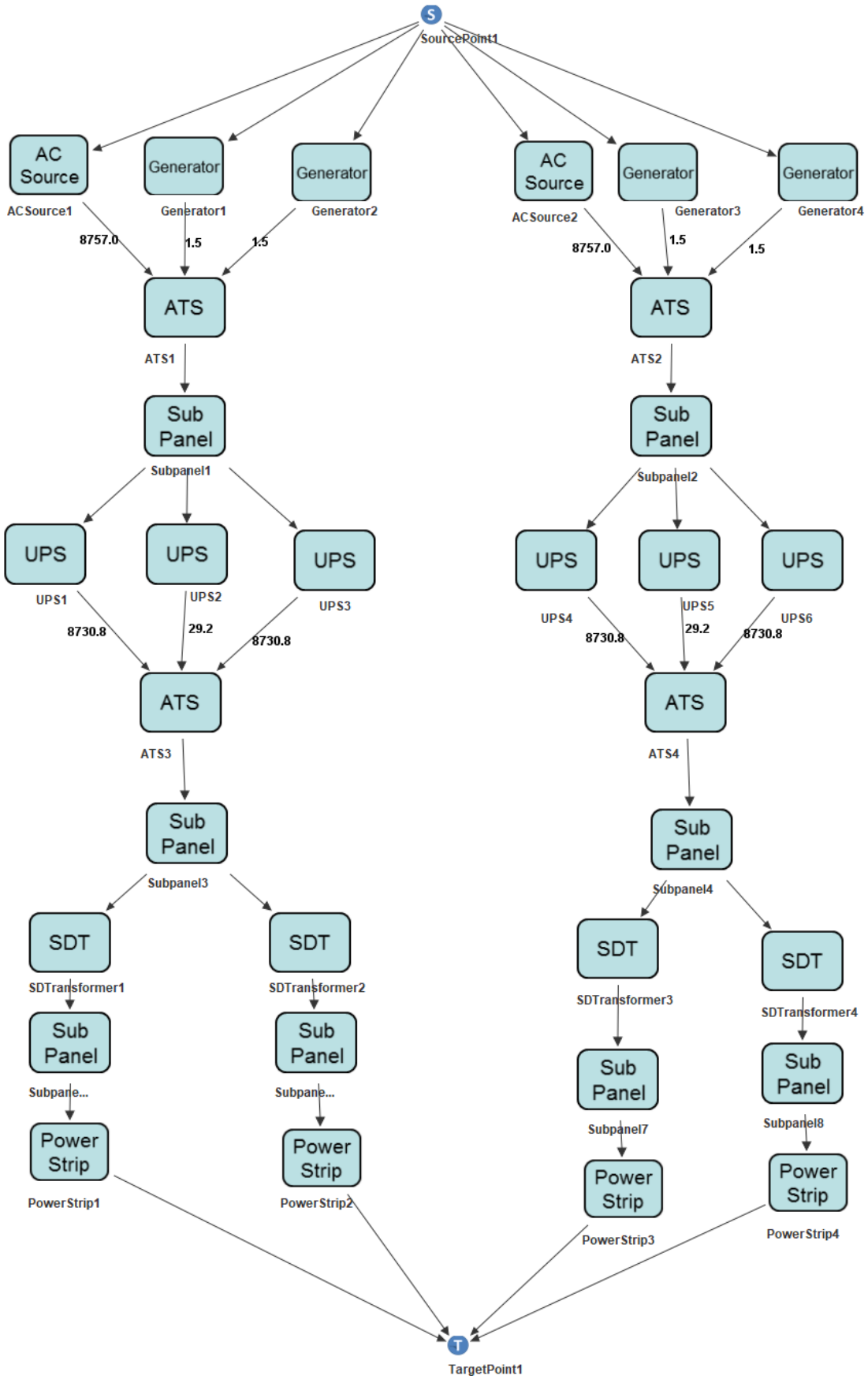


Figura 53 – Modelo EFM TIER IV

6.3.5 Resultado

Os resultados e análise apresentado neste estudo de caso se valem das estratégias de otimização validadas nos Estudos de Caso I e II. É importante destcar que devido à complexidade dos modelos analisados não é factível a utilização do algoritmo de força bruta, dado o seu tempo para realizar a avaliação destes modelos. Sendo assim, nesta seção serão apresentados os resultados dos quatro modelos baseados nos TIERs.

- Tier I

  Foi aplicado o teste t-pareado com $\alpha = 0,05$ para analisar a diferença estatística das médias das métricas antes de aplicar as estratégias de otimização e após aplicar em 100 gerações. Verificou-se o valor-p $< 0,05$ é possível concluir que existe uma diferença na média da disponibilidade na geração 1 e depois de aplicar a estratégia de otimização. O mesmo teste foi aplicado para a segunda estratégia baseada no MOPSO, comparando a média antes aplicar a otimização e 100 gerações após. O estudo estatístico teste t-pareado resulta em valor-p $< 0,05$ indica diferença significativa. A Figura 54 apresenta a evolução da otimização das estratégias evolucionárias. A estratégia baseada no NSGA-II se mostrou superior, contudo apresentou as médias de suas métricas otimizadas dentro do intervalo de confiança de 95% (ver Tabela 5) das métricas otimizadas pela estratégia baseada no MOPSO.

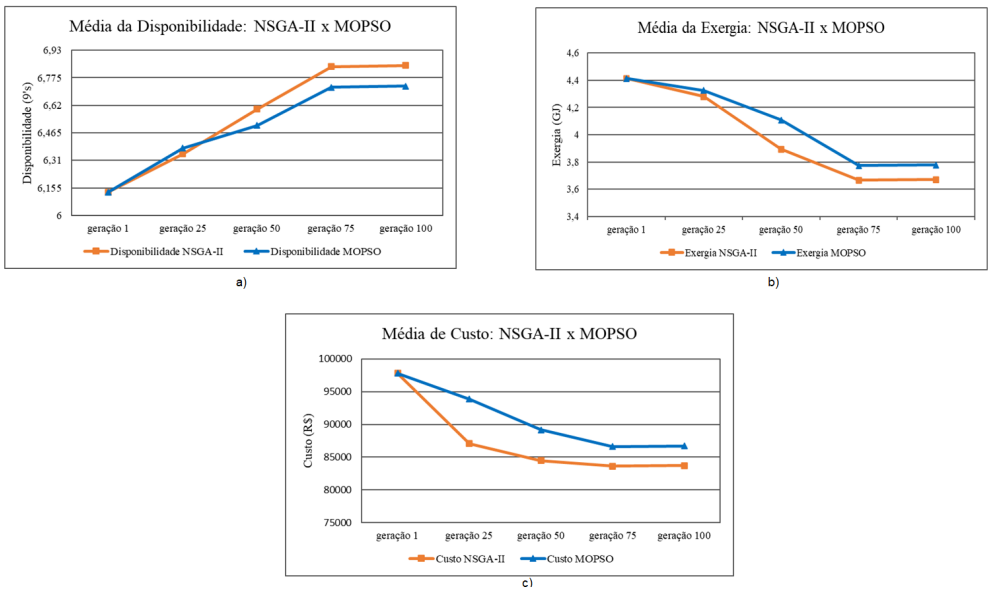


Figura 54 – Comparação NSGA-II x MOPSO - TIER I

- Tier II

  Os resultados obtidos após aplicar as estratégias evolucionárias indicam diferença significativa quando analisadas a população inicial e final. Isso pode ser constatado pelo teste-t pareado, onde para todos os cenários se teve seu valor-p $< 0{,}05$. Todavia, a disponibilidade até a geração 50 tinha no Algoritmo baseado no MOPSO melhor performance quantitativa, sendo superada pela Algoritmo baseano no NSGA-II entre as gerações 50 e 75. Semelhantemente ocorre com a exergia que até a geração 25 o MOPSO se mostrava mais eficiente quando comparado ao NSGA-II, contudo ao se aproximar da geração 50 foi superado pelo NSGA-II. A Figura 55 apresenta a evolução da otimização das estratégias evolucionárias. A estratégia baseada no NSGA-II tem as médias de suas métricas otimizadas dentro do intervalo de confiança de 95% (ver Tabela 5) das métricas otimizadas da estratégia baseada no MOPSO.

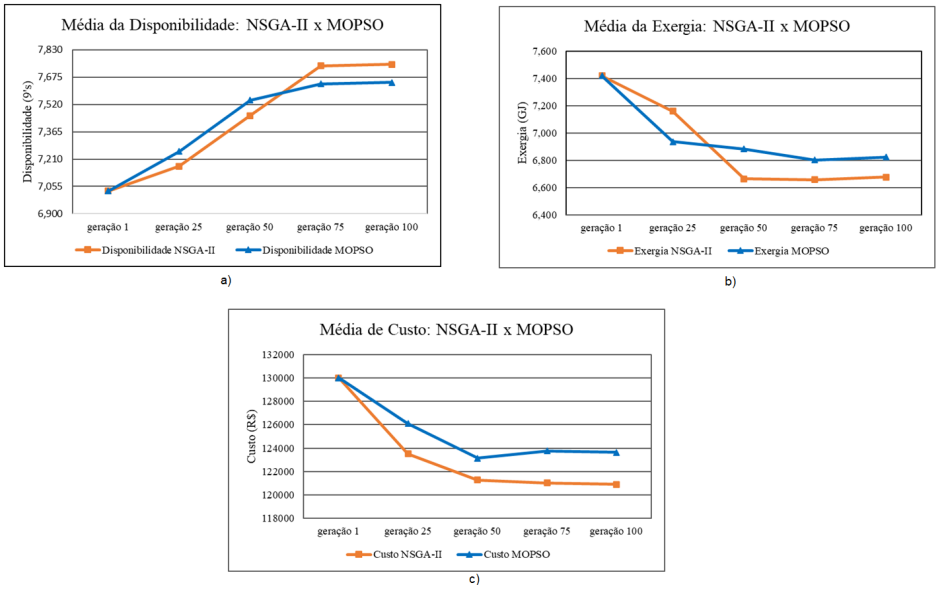


Figura 55 – Comparação NSGA-II x MOPSO - TIER II

- Tier III

  Os resultados obtidos após aplicar as estratégias evolucionárias indicam diferença significativa quando analisadas a população inicial e final. Isso pode ser constatado pelo teste-t pareado, onde para todos os cenários se teve seu valor-p $< 0{,}05$. Além disso, foi verificado também que tanto o comportamento das médias da disponibilidade, quanto do custo ao longo das gerações não houve diferença significativa, quando comparam-se as duas estratégias evolucionárias. Todavia a exergia não apresenta este comportamento em todas as geração. Na geração 50, por

exemplo, o valor-p $< 0{,}05$ apresentando com isso diferença significativa. Porém, entre as gerações 75 a 100 ambas possuem similaridade, mesmo com melhor performance quantitativa do NSGA-II. O desempenho do MOPSO até próximo a geração 75 era melhor quando comparado ao NSGA-II, contudo da geração 75 a 100 o NSGA-II é superior quantitativamente. A Figura 56 apresenta a evolução da otimização das estratégias evolucionárias. A estratégia baseada no NSGA-II tem as médias de suas métricas otimizadas dentro do intervalo de confiança de 95% (ver Tabela 5) das métricas otimizadas da estratégia baseada no MOPSO..

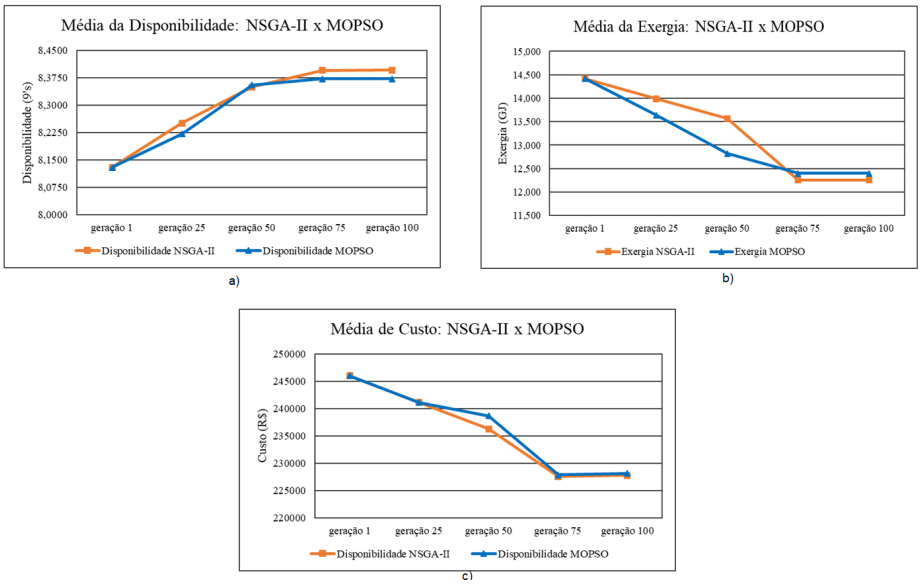


Figura 56 – Comparação NSGA-II x MOPSO - TIER III

- Tier IV

  Os resultados obtidos após aplicar as estratégias evolucionárias indicam diferença significativa quando analisadas a população inicial (antes de aplicar a estratégia de otimização) e final (após aplicar a estratégia). Isso pode ser constatado pelo teste-t pareado, onde para todos os cenários se teve seu valor-p $< 0{,}05$. Ao verificar os dados de cada geração foi possível constatar que a disponibilidade não possui diferença estatisticamente significava quando se usa NSGA-II ou MOPSO, o que não ocorre com a exergia que tem na geração 25 maior diferença, que só vem a ser superada a partir da geração 50 onde os dados quantitativamente diferem no máximo de 0,5%. A Figura 57 apresenta a evolução da otimização das estratégias evolucionárias. A estratégia baseada no NSGA-II tem as médias de suas métricas otimizadas dentro do intervalo de confiança de 95% (ver Tabela 5) das métricas otimizadas da estratégia

baseada no MOPSO.

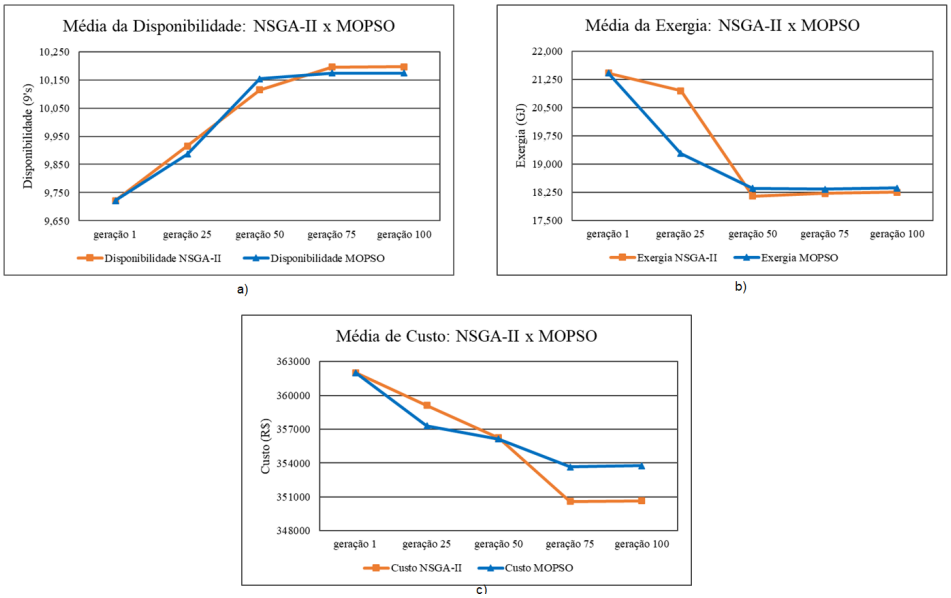


Figura 57 – Comparação NSGA-II x MOPSO - TIER IV

Tabela 5 – Comparativo das Estratégias de Otimização Aplicadas às Arquiteturas TIER

		Estratégia baseada no NSGA-II				Estratégia baseada no MOPSO			
Arquitetutura	Métrica	Média	Desv. Pad.	IC	Tempo (s)	Média	Desv. Pad.	IC	Tempo (s)
Tier I	Disp.(9's)	6,843	0,148	[6,843; 6,873]		6,728	0,593	[6,601; 6,846]	
	Exergia (GJ)	3,671	0,798	[3,512; 3,831]	357	3,777	0,735	[3,591; 3,964]	245
	Custo (US$)	83726,08	11617,95	[81408,92; 86043,24]		86771,45	18170,79	[83087,86; 90335,03]	
Tier II	Disp.(9's)	7,747	0,209	[7,705; 7,788]		7,645	0,807	[7,485; 7,805]	
	Exergia (GJ)	6,676	0,794	[6,520; 6,834]	486	6,822	0,748	[6,674; 6,971]	407
	Custo (US$)	120912,27	11649,42	[118600,78; 123223,77]		123650,05	12182,97	[120438,69; 126861,73]	
Tier III	Disp.(9's)	8,396	0,042	[8,387; 8,404]		8,372	0,121	[8,347; 8,396]	
	Exergia (GJ)	12,256	0,752	[12,106; 12,405]	721	12,401	0,96	[12,209; 12,590]	645
	Custo (US$)	227768,38	15451,42	[224702,48; 230834,27]		228110,03	18740,59	[224391,60; 231828,45]	
Tier IV	Disp.(9's)	10,197	0,052	[10,186; 10,207]		10,174	0,129	[10,149; 10,201]	
	Exergia (GJ)	18,251	0,859	[18,081; 18,422]	1218	18,364	0,877	[18,190; 18,538]	965
	Custo (US$)	350643,74	14260,45	[347814,63; 353473,16]		353768,34	15451,41	[350702,93; 356834,74]	

Legenda: Disp.(9's) é a Disponibilidade (em números de noves); Desv. Pad. é o desvio padrão considerando os 50 individuos da população otimizada após 100 gerações; [IC] - [Intervalo de Confiança com nível de significância de 95%]; Tempo (s) - Tempo de Execução da Estratégia de Otimização.

A Tabela 5 apresenta os resultados da otimização das arquiteturas TIERs. Os resultados mostram que a estratégia baseada no NSGA-II teve, em todos os cenários, seu tempo de execução maior que aquele baseado no MOPSO. Contudo, a estratégia baseada NSGA-II apresentou, em todas as TIERs, maior disponibilidade, menores custos e exergia.

7 Conclusão

Este trabalho de pesquisa desenvolveu estratégias de otimização multiobjetivo baseadas em técnicas evolucionárias com objetivo de maximizar a disponibilidade e ao mesmo tempo minimizando custo e exergia operacional dos subsistemas elétricos de *data center*. Neste capítulo são apresentadas as contribuições, as limitações encontradas e os trabalhos futuros.

7.1 Considerações

Duas estratégias de otimização são apresentadas neste trabalho. Uma é baseada no algoritmo elistista (NSGA-II) que através de adaptações em sua função de ranqueamento foi possível ordenar as soluções pelo conjunto de indivíduos não dominados, e assim sendo, aplicar operadores genéticos (cruzamento e mutação) naqueles mais bem ranqueados. Outra adaptação está na chamada do operador *crowding distante* permitindo que se exclua os piores indivíduos de uma determinada arquitetura. A outra estratégia é baseada no algoritmo de otimização de enxame de partículas (MOPSO), onde cada partícula foi mapeada como um conjunto de equipamentos que faz parte de uma arquitetura. Diferentemente da primeira, esta última estratégia atualiza os dados da partícula a partir de sua posição no espaço de busca. Cada eixo do espaço de busca é um atributo do componente (MTTF, eficiência energética, custo).

Estudos de caso evidenciaram a validade da abordagem proposta, onde ambas estratégias de otimização fornecem resultados semelhantes para um intervalos de confiança de 95%, ou seja, os intervalos de confiança das métricas otimizadas pela estratégia proposta sobrescrevem as médias de cada métrica otimizadas pelo Força Bruta. Além disso, houve importantes redução do número de cenários avaliados e o tempo de execução teve sua redução em até 589 vezes. No Estudo de caso I, a utilização da otimização baseada no NSGA-II trouxe resultados constatando que a curva Pareto Aproximada ficou próximo a curva Pareto ótimo gerado pelo Força Bruta com a diferença máxima de 3,1%. No Estudo de Caso II foi posta a prova a estratégia baseada no MOPSO comparando o conjunto de soluções Pareto Aproximado com a curva Pareto ótimo gerada pelo força bruta no Estudo de Caso I. Além disso, foram realizadas análise estatística (com base em erro

padrão) para comparar as métricas utilizando mesmo intervalo de confiança do primeiro estudo de caso. Com as estratégias validadas foi possível realizar um terceiro estudo de caso usando modelos as arquiteturas TIER (infraestrutura elétrica de *data centers* mais complexas). Neste cenário, as estratégias tiveram desempenho semelhantes, em alguns casos com menos gerações o MOPSO teve melhor desempenho que o NSGA-II. Porém, ao final de 100 gerações, a estratégia baseada no NSGA-II se mostrou quantitativamente superior a estratégia baseada no MOPSO. Por outro lado, os estudos estatísticos aplicados demonstram que ambas estratégias são válidas para otimização multiobjetivo e ao final das 100 gerações não houve diferença, estatisticamente significativa, entre aquela baseada no NSGA-II ou MOPSO.

7.2 Contribuições

Esta seção apresenta todas as principais contribuições deste trabalho. Abaixo seguem os resultados mais relevantes para esta pesquisa:

- Os modelos híbridos SPN/RBD para redundância *cold standby*
  - A proposta de adotar um modelo híbrido foi a solução encontrada para modelos em que o RBD não consegue avaliar por si só. Assim, usou-se o SPN que é capaz de avaliar a disponibilidade de um modelo redundância *cold standby*, e após isso, atribuir esta disponibilidade para um bloco RBD. A utilização desta técnica se mostrou eficiente tendo em vista que o tempo de execução foi reduzido em aproximadamente 30% quando comparado a modelagem exclusivamente feita por SPN.
- Desenvolvimento de duas estratégias de otimização baseado em técnicas evolucionárias:
  - Com a utilização da estratégia proposta de otimização é possível realizar experimentos mais direcionados, por exemplo, dando peso aos critérios avaliados, mudança de infraestrutura, ampliação de arquiteturas.
  - Ao realizar a proposta de otimização foi possível obter os resultados em um menor tempo de execução. Se observar os tempos de execução dos experimentos no Estudo de caso II, pode-se perceber que a estratégia baseada no MOPSO tem uma diminuição considerável no tempo de execução do experimento feito

no Estudo de caso I.

- Funcionalidade otimização na ferramenta *Stars*:
  - As estratégias apresentadas foram implementadas e já encontram-se no módulo otimização da ferramenta *Stars*. Este modelo possibilita gerar os casos de teste a avaliar outras métricas e tam. Vale ressaltar que a o cálculo realizado pelos métodos *solve* e a linguagem de *script* são feitas através da interação da aplicação desenvolvida com a ferramenta Mercury. O caminho do projeto pode ser consultado em <http://callou.pythonanywhere.com/stars>

7.3 Limitações e Trabalhos Futuros

Como limitação desse trabalho, temos que foi avaliado um modelo EFM de forma estática, sem levar as falhas na simulação do EFM para obter os resultados. Outra limitação, pode ser quanto ao alto tempo de avaliação por ter sido realizado em um desktop. Diversos direcionamentos podem ser seguidos e propostos a partir deste trabalho. Abaixo são enumeradas alguns possíveis trabalhos futuros desta pesquisa.

- As estratégias propostas podem ser consideradas para avaliar outros cenários ampliando o escopo do estudo para outros tipos de infraestruturas de *data center* como TI e refrigeração.
- Aplicar a estratégia para otimizar serviços em nuvem, por exemplo, um problema de caminho mais curto em grafos onde vários critérios são considerados simultaneamente.
- Analisar cenários de balanceamento de carga de servidores, avaliar novas métricas de desempenho e outras ferramentas que modelem formalismos (RBD, SPN e EFM).
- Aplicar as estratégias em modelos de desempenho analisando a complexidade computacional e o tempo de execução. Além disso, utilizar novos testes estatísticos, que possibilitem obter conclusões seguras na comparação entre as estratégias de otimização.
- Aplicar a mesma estratégia utilizando outro contexto, por exemplo, computação em nuvem como cenário de avaliação. Podendo aplicar as técnicas de otimização, inclusive, para melhorar um balanceador de carga visando a redução de custo e consumo de energia, por exemplo.

Referências

ABUALIGAH, L.; DIABAT, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. **Cluster Computing**, p. 205–223, 2021.

AHAT, B.; BAKTıR, A. C.; ARAS, N.; ALTıNEL, K.; ÖZGöVDE, A.; ERSOY, C. Optimal server and service deployment for multi-tier edge cloud computing. **Computer Networks**, v. 199, p. 108393, 2021. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128621003716>.

ARROYO, J. E. C.; ARMENTANO, V. A. Genetic local search for multi-objective flowshop scheduling problems. **European Journal of Operational Research**, v. 167, n. 3, p. 717–738, 2005.

AUSTREGéSILO, M. S. S.; CALLOU, G. Stochastic models for optimizing availability, cost and sustainability of data center power architectures through genetic algorithm. **Revista de Informática Teórica e Aplicada**, v. 26, n. 2, p. 27–44, Aug. 2019. Disponível em: <https://seer.ufrgs.br/index.php/rita/article/view/RITA_VOL26_NR2_27>.

BEZERRA, T. V.; LEONARDO, W. d. S.; JúNIOR, G. A. d. A.; CALLOU, G. R. d. A. Dimensioning the relationship between availability and data center energy flow metrics. v. 27, p. 63–78, dez. 2020. Disponível em: <https://seer.ufrgs.br/index.php/rita/article/view/RITA_VOL27_NR4_63>.

CALLOU, G. Planning of sustainable data centers with high availability. In: . [S.l.: s.n.], 2013.

CALLOU, G.; FERREIRA, J.; MACIEL, P.; TUTSCH, D.; SOUZA, R. An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost. **Energies**, Multidisciplinary Digital Publishing Institute, v. 7, n. 1, p. 238–277, 2014.

CHADLI, M. Multiobjective optimisation and control. liu, g.p., yang, j.b., whidborne, j.f. **Control Engineering Practice - CONTROL ENG PRACTICE**, v. 15, p. 273–274, 02 2007.

CHALISE, S.; GOLSHANI, A.; AWASTHI, S. R.; MA, S.; SHRESTHA, B. R.; BAJRACHARYA, L.; SUN, W.; TONKOSKI, R. Data center energy systems: Current technology and future direction. In: IEEE. **2015 IEEE Power & Energy Society General Meeting**. [S.l.], 2015. p. 1–5.

CHONG, F. T.; HECK, M. J. R.; RANGANATHAN, P.; SALEH, A. A. M.; WASSEL, H. M. G. Data center energy efficiency:improving energy efficiency in data centers beyond technology scaling. **IEEE Design Test**, v. 31, n. 1, p. 93–104, 2018.

COELLO, C. A. An updated survey of ga-based multiobjective optimization techniques. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 32, n. 2, p. 109–143, jun 2000. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/358923.358929>.

DAYARATHNA, M.; WEN, Y.; FAN, R. Data center energy consumption modeling: A survey. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 732–794, 2016.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.

EHRGOTT, M.; IDE, J.; SCHöBEL, A. Minmax robustness for multi-objective optimization problems. **European Journal of Operational Research**, v. 239, n. 1, p. 17–31, 2014. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221714002276>.

ENDO, P. Analyzing the impact of power subsystem failures and checkpoint mechanisms on availability of cloud applications. **IEEE Latin America Transactions**, v. 18, n. 1, p. 138–146, Mar. 2020. Disponível em: <https://latamt.ieeer9.org/index.php/transactions/article/view/826>.

EVANS, R. A.; GAO, J.; RYAN, M. C.; DULAC-ARNOLD, G.; SCHOLZ, J. K.; HESTER, T. A. **Optimizing data center controls using neural networks**. [S.l.]: Google Patents, 2020. US Patent 10,643,121.

FERREIRA, J.; CALLOU, G.; MACIEL, P.; TUTSCH, D. An algorithm to optimise the energy distribution of data centre electrical infrastructures. **International Journal of Grid and Utility Computing**, v. 11, n. 3, p. 419–433, 2020. Disponível em: <https://www.inderscienceonline.com/doi/abs/10.1504/IJGUC.2020.107625>.

FERREIRA, L.; ENDO, P. T.; GONCALVES, G.; ROSENDO, D.; SANTOS, G. L.; MOREIRA, A.; KELNER, J.; SADOK, D.; WILDEMAN, M.; MEHTA, A. Maximizing the availability of composable systems of next-generation data centers. In: **2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)**. IEEE Press, 2019. p. 573–580. Disponível em: <https://doi.org/10.1109/SMC.2019.8914382>.

FONSECA, L. P. G.; NUNES, V. D. L.; SANTANA, L. O.; CARLO, J. C.; CÉSAR, K. M. L. Otimização multiobjetivo das dimensões dos ambientes de uma residência unifamiliar baseada em simulação energética e estrutural. **Ambiente Construído**, SciELO Brasil, v. 17, p. 267–288, 2017.

GHAREHPASHA, S.; MASDARI, M. A discrete chaotic multi-objective sca-alo optimization algorithm for an optimal virtual machine placement in cloud data center. **Datacenter: projeto, operação e serviços-Unisul Virtual**, v. 12, 2021.

GIAGKIOZIS, I.; FLEMING, P. J. Pareto Front Estimation for Decision Making. **Evolutionary Computation**, v. 22, n. 4, p. 651–678, 12 2014. ISSN 1063-6560.

GONÇALVES, G.; GOMES, D.; LEONI, G.; ROSENDO, D.; MOREIRA, A.; KELNER, J.; SADOK, D.; ENDO, P. Optimizing the cloud data center availability empowered by surrogate models. In: . [S.l.: s.n.], 2020.

GONG, W.; DUAN, Q.; LI, J.; WANG, C.; DI, Z.; YE, A.; MIAO, C.; DAI, Y. Multi-objective adaptive surrogate modeling-based optimization for parameter estimation of large, complex geophysical models. **Water Resources Research**, v. 52, 12 2015.

GREGORY, X. L. L.; YUANSHUN, D. Cold vs. hot standby mission operation cost minimization for 1-out-of-n system. **European Journal of Operational Research**, v. 1, p. 155–162, 2014.

GUIMARÃES, A. P.; PEREIRA, A. Análise de aspectos de dependabilidade em sistemas de data centers integrando as infraestruturas de comunicação, de potência e de refrigeração. **Revista Brasileira de Administração Científica**, 2020.

HUANG, Y.; LI, G.; WANG, P.; CHANG, F.; LI, J. Electricy cost optimization of data center interactive services with ups. In: **2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)**. [S.l.: s.n.], 2018. p. 181–184.

HUO, J.; LIU, J.; GAO, H. An nsga-ii algorithm with adaptive local search for a new double-row model solution to a multi-floor hospital facility layout problem. **Applied Sciences**, v. 11, n. 4, 2021. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/11/4/1758>.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4.

KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-objective optimization using genetic algorithms: A tutorial. **Reliability Engineering System Safety**, v. 91, n. 9, p. 992–1007, 2006. ISSN 0951-8320. Special Issue - Genetic Algorithms and Reliability. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0951832005002012>.

LEI, N.; MASANET, E. R. Global data center energy demand and strategies to conserve energy. **Data Center Handbook: Plan, Design, Build, and Operations of a Smart Data Center**, Wiley Online Library, p. 15–26, 2021.

LEONARDO, W. de S.; CALLOU, G. Stars: um ambiente integrado para avaliação de disponibilidade, custo e consumo de energia de sistemas. **Revista Brasileira de Computação Aplicada**, v. 13, n. 3, p. 10–21, set. 2021. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/11595>.

LI, Y.; WEN, Y.; TAO, D.; GUAN, K. Transforming cooling optimization for green data center via deep reinforcement learning. **IEEE Transactions on Cybernetics**, v. 50, n. 5, p. 2002–2013, 2020.

LOMAN, J.; WANG, W. On reliability modeling and analysis of highly-reliable large systems. In: **Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No.02CH37318)**. [S.l.: s.n.], 2002. p. 456–459.

MARSAN, M. A. Stochastic petri nets: An elementary introduction. In: _____. **Advances in Petri Nets 1989**. Berlin, Heidelberg: Springer-Verlag, 1990. p. 1–29. ISBN 0387524940.

MASANET, E.; SHEHABI, A.; LEI, N.; SMITH, S.; KOOMEY, J. Recalibrating global data center energy-use estimates. **Science**, v. 367, n. 6481, p. 984–986, 2020.

MASHWANI, W. K.; SALHI, A. Multiobjective evolutionary algorithm based on multimethod with dynamic resources allocation. **Appl. Soft Comput.**, Elsevier Science Publishers B. V., NLD, v. 39, n. C, p. 292–309, feb 2016. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2015.08.059>.

MELO F., A. E. C. G. Optimization of electrical infrastructures at data centers through a doe-based approach. **The Journal of Supercomputing**, The Journal of Supercomputing, v. 78, p. 406–439, 2022.

MELO, F. F. de L.; CALLOU, G. R. de A.; SOUSA, E. T. G. de. Sensitivity analysis of data center electrical infrastructures considering aspects of dependability and cost. **IEEE Latin America Transactions**, IEEE, v. 19, n. 02, p. 235–242, 2021.

MISHRA, S. K.; P, G.; MEHER, S.; MAJHI, R. A fast multiobjective evolutionary algorithm for finding wellspread pareto-optimal solutions. In: **In KanGAL Report No. 2003002,lndian Institute Of Technology Kanpur**. [S.l.: s.n.], 2002.

NGUYEN, T. A.; MIN, D.; CHOI, E.; TRAN, T. D. Reliability and availability evaluation for cloud data center networks using hierarchical models. **IEEE Access**, v. 7, p. 9273–9313, 2019.

OLIVEIRA, D.; MATOS, R.; DANTAS, J.; FERREIRA, J. a.; SILVA, B.; CALLOU, G.; MACIEL, P.; BRINKMANN, A. Advanced stochastic petri net modeling with the mercury scripting language. In: . New York, NY, USA: Association for Computing Machinery, 2017. (VALUETOOLS 2017), p. 192–197. ISBN 9781450363464. Disponível em: <https://doi.org/10.1145/3150928.3150959>.

PATIL, D.; DANGEWAR, B. Multi-objective particle swarm optimization (mopso) based on pareto dominance approach. **International Journal of Computer Applications**, v. 107, p. 975–8887, 01 2015.

RAK, T. Modeling web client and system behavior. **Information**, Multidisciplinary Digital Publishing Institute, v. 11, n. 6, p. 337, 2020.

REIS, A. C. B.; DOURADO, L. dos S.; NÓBREGA, F. F. da. Análise de decisão para selecionar uma solução de nuvem corporativa. **Revista Ibérica de Sistemas e Tecnologias de Informação**, Associação Ibérica de Sistemas e Tecnologias de Informacao, n. E28, p. 244–257, 2020.

REISIG, W. **Petri Nets: An Introduction**. Berlin, Heidelberg: Springer-Verlag, 1985. ISBN 0387137238.

ROSEN, M. A. Improving the efficiency of electrical systems via exergy methods. In: IEEE. **2007 IEEE Canada Electrical Power Conference**. [S.l.], 2007. p. 467–472.

ROSENDO, D.; GOMES, D.; SANTOS, G. L.; GONCALVES, G.; MOREIRA, A.; FERREIRA, L.; ENDO, P. T.; KELNER, J.; SADOK, D.; MEHTA, A. et al. A methodology to assess the availability of next-generation data centers. **The Journal of Supercomputing**, Springer, v. 75, n. 10, p. 6361–6385, 2019.

SAKALKAR, V.; KONTORINIS, V.; LANDHUIS, D.; LI, S.; RONDE, D. D.; BLOOMING, T.; RAMESH, A.; KENNEDY, J.; MALONE, C.; CLIDARAS, J.; RANGANATHAN, P. Data center power oversubscription with a medium voltage power plane and priority-aware capping. In: _____. **Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems**. New York, NY, USA: Association for Computing Machinery, 2020. p. 497–511. ISBN 9781450371025. Disponível em: <https://doi.org/10.1145/3373376.3378533>.

SANCHES, D. S.; MARQUES, L. T.; BORGES, H. F.; DELBEM, A. C. B.; JR, J. B. A. L.; PROCÓPIO, C. **Análise comparativa entre algoritmos evolutivos multi-objetivos aplciados ao problema de reduçao de perdas em sistemas de distribuiçao de grande porte**. [S.l.]: CBA, 2012.

SCHAFFER, J. D. Some experiments in machine learning using vector evaluated genetic algorithms. 1 1985. Disponível em: <https://www.osti.gov/biblio/5673304>.

SRIVASTAVA, G.; SINGH, A.; MALLIPEDDI, R. Nsga-ii with objective-specific variation operators for multiobjective vehicle routing problem with time windows. **Expert Systems with Applications**, Elsevier, v. 176, p. 114779, 2021.

STAMATESCU, I.; PLOIX, S.; FăGăRăşAN, I.; GRIGORE. Data center server energy consumption optimization algorithm. In: **2018 26th Mediterranean Conference on Control and Automation (MED)**. [S.l.: s.n.], 2019. p. 813–818.

TEGHEM, J. Multi-objective combinatorial optimizationmulti-objective combinatorial optimization. In: _____. **Encyclopedia of Optimization**. Boston, MA: Springer US, 2009. p. 2437–2442. ISBN 978-0-387-74759-0. Disponível em: <https://doi.org/10.1007/978-0-387-74759-0_418>.

TIA-942. **Telecommunications Infrastructure Standard for Data Centers ANSI/TIA-942**. Telecommunication Industry Association, 2005. Disponível em: <https://books.google.com.br/books?id=6RZIjwEACAAJ>.

UDDIN, M.; ALSAQOUR, R. A.; SHAH, A.; SABA, T. Power usage effectiveness metrics to measure efficiency and performance of data centers. **Applied Mathematics & Information Sciences**, v. 8, p. 2207–2216, 2014.

UPTIME. **Tier certification**. 2021. Acessado em: 20-01-2022. Disponível em: <https://pt.uptimeinstitute.com/tier-certification>.

VALENTIM, T.; CALLOU, G. A model-based strategy for quantifying the impact of availability on the energy flow of data centers. v. 77, 2021. Disponível em: <https://doi.org/10.1007/s11227-020-03353-4>.

VELDHUIZEN, D. On measuring multiobjective evolutionary algorithm performance. In: . [S.l.: s.n.], 2000. v. 1, p. 204 – 211 vol.1. ISBN 0-7803-6375-2.

VELDHUIZEN, D. A. V.; LAMONT, G. B. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. **Evol. Comput.**, MIT Press, Cambridge, MA, USA, v. 8, n. 2, p. 125–147, jun 2000. ISSN 1063-6560. Disponível em: <https://doi.org/10.1162/106365600568158>.

WANG, D.; XIE, C.; WU, R.; LAI, C. S.; LI, X.; ZHAO, Z.; WU, X.; XU, Y.; LAI, L. L.; WEI, J. Optimal energy scheduling for data center with energy nets including cchp and demand response. **IEEE Access**, IEEE, v. 9, p. 6137–6151, 2021.

WANG, D.-J.; LIU, F.; JIN, Y. A multi-objective evolutionary algorithm guided by directed search for dynamic scheduling. **Computers Operations Research**, v. 79, p. 279–290, 2017. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S030505481630096X>.

WANG, Q.; SONG, M.; FANG, Q.; WANG, J. Thermal-aware flow field optimization for energy saving of data centers. In: **2018 Annual American Control Conference (ACC)**. [S.l.: s.n.], 2018. p. 3744–3749.

WANG, W.; LOMAN, J.; ARNO, R.; VASSILIOU, P.; FURLONG, E.; OGDEN, D. Reliability block diagram simulation techniques applied to the ieee std. 493 standard network. **IEEE Transactions on Industry Applications**, v. 40, n. 3, p. 887–895, 2004.

WANG, X.; LI, H.; WANG, Y.; ZHAO, J.; ZHU, J.; ZHONG, S.; LI, Y. Energy, exergy, and economic analysis of a data center energy system driven by the co2 ground source heat pump: Prosumer perspective. **Energy Conversion and Management**, Elsevier, v. 232, p. 113877, 2021.

YAO, Z.; DESMOUCEAUX, Y.; TOWNSLEY, M.; CLAUSEN, T. H. Towards intelligent loadbalancing in data centers. **5th Workshop on Machine Learning for Systems at 35th Conference onNeural Information Processing Systems (NeurIPS 2021)**, 2021.

ZHANG, Q.; JIANG, S.; YANG, S.; SONG, H. Solving dynamic multi-objective problems with a new prediction-based optimization algorithm. **PLOS ONE**, Public Library of Science, v. 16, n. 8, p. 1–39, 08 2021. Disponível em: <https://doi.org/10.1371/journal.pone.0254839>.

ZHU, M.; HAN, F. Multi-objective particle swarm optimization based on space decomposition for feature selection. In: **2021 17th International Conference on Computational Intelligence and Security (CIS)**. [S.l.: s.n.], 2021. p. 387–391.