UPF EDITORA UNIVERSIDADE DE PASSO FUNDO

RBCA ISSN 2176-6649

Revista Brasileira de Computação Aplicada, Novembro, 2021

DOI: 10.5335/rbca.v13i3.11595
Vol. 13, Nº 3, pp. 10–21
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

Stars: Um ambiente integrado para avaliação de disponibilidade, custo e consumo de energia de sistemas

Stars: an integrated environment for assessing availability, cost and energy consumption of systems

Wenderson de Souza Leonardo[1] and Gustavo Rau de Almeida Callou [2]

[1]Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco, [2]Departamento de Computação, Universidade Federal Rural de Pernambuco

* wenderson.leonardo@ufrpe.br; † gustavo.callou@ufrpe.br

Recebido: 25/08/2020. **Revisado:** 07/07/2021. **Aceito:** 08/09/2021.

Resumo

Sustentabilidade tem recebido atenção crescente da comunidade científica, sendo o maior foco a redução do consumo energético e também na manutenção de recursos não renováveis para as futuras gerações. Em paralelo, a expansão de paradigmas como a computação em nuvem, redes sociais e comércio eletrônico acabou por aumentar a demanda dos *data centers*. Nesse contexto, ferramentas que dão suporte a modelagem de arquiteturas de *data center* e que sejam capazes de computar métricas como a de disponibilidade, custo e consumo energético são de extrema importância. Este projeto propõe uma ferramenta, denominada Stars, para modelagem de arquiteturas de *data centers* que é capaz de computar métricas como consumo energético, disponibilidade e custo. Nessa ferramenta, usuários não especializados não necessitam conhecer o formalismo adotado (ex. RBD, SPN e EFM) pela ferramenta para computar as métricas de interesse. Além disso, um algoritmo de otimização, chamado Algoritmo Genético, foi integrado à ferramenta para maximizar os resultados alcançados através de uma lista de componentes. Este algoritmo é capaz de encontrar uma combinação de componentes para uma dada arquitetura de *data center* em uma fração reduzida de tempo em comparação ao algoritmo de força bruta. Resultados alcançados demonstraram que foi possível se obter respostas em menos de 3 segundos com o algoritmo genético em comparação com os 255 segundos demandados pelo algoritmo de força bruta.

Palavras-Chave: Algoritmo Genético; Disponibilidade; Modelo de fluxo de energia; Redes de Petri.

Abstract

Sustainability has received increasing attention from the scientific community, with a strong focus on reducing energy consumption and maintaining nonrenewable resources for future generations. In parallel, the expansion of cloud computing, social networking, and e-commerce has increased the demand for data centers. In this context, tools that support the modeling of data center architectures and compute metrics such as availability, cost, and energy consumption are extremely important. This paper proposes a tool named Stars for modeling data center architectures that can compute such metrics. Besides that, non-specialized users do not need to know the formalism adopted by the engine to compute the desired metrics (e.g., RBD, SPN, and EFM). Furthermore, an optimization algorithm, named Genetic Algorithm, was integrated into the tool to maximize the results achieved through a list of components. This algorithm can find a combination of components for a given data center architecture in a very reduced fraction of time compared to the brute force algorithm. Results achieved showed that it was possible to obtain responses in less than 3 seconds with the genetic algorithm compared to the 255 seconds required by the brute force algorithm.

Keywords: Availability; Energy flow model; Genetic Algorithm; Petri nets.

1 Introdução

Atualmente, devido ao aquecimento global, o aumento da poluição, a falta de água potável, entre outros fatores ambientais, o mundo começou a entender a necessidade de se realizar mudanças comportamentais em relação ao meio ambiente. Essas mudanças têm como principal foco a redução do consumo de energia e do impacto ambiental decorrente. Dentro desse contexto, a comunidade científica e várias empresas iniciaram uma busca por métodos de desenvolvimento que atendam às atuais necessidades de energia, sem comprometer os recursos não renováveis. Dessa forma, o desenvolvimento sustentável passou não apenas a ser um foco para melhorias, mas sim uma exigência da sociedade (Bell and Morse, 2000).

Em relação à área de Tecnologia de Informação (TI), o surgimento de novos paradigmas como a computação em nuvem e as redes sociais, entre outros, tem demandado uma maior capacidade das infraestrutura dos *data centers*. A computação em nuvem vem dirigindo essa nova tendência de utilizar aplicações online, tomando como base a Internet ao prover *software* como serviço. Dessa forma, crescentes demandas aos *data centers* surgiram, e além de desempenho satisfatório, passou-se a exigir que tais ambientes devem prover os serviços com alta disponibilidade. A consequência direta de tal crescimento da demanda é o aumento no custo operacional e no impacto ambiental, principalmente, devido à redundância nas arquiteturas de equipamentos necessários para prover as operações dos servidores.

Grandes *data centers*, que reúnem servidores com alto poder de processamento, são considerados ameaças ao meio ambiente por tenderem a aumentar sua demanda em 66% até 2035 (Than and Thein, 2020). Na verdade, a maior demanda por energia é uma questão que tem impactado a forma como os sistemas dos *data centers* são concebidos, no sentido de que os projetistas precisam verificar vários *trade-offs* e selecionar a solução mais viável considerando a utilização de energia e outras métricas, como a disponibilidade.

É importante destacar que esforços vêm sendo realizados com o objetivo de tornar os *data centers* mais sustentáveis. Uma análise do consumo de energia em conjunto com a utilização de fontes renováveis e limpas é de fundamental importância para a redução do impacto ambiental desses sistemas. Sendo assim, as arquiteturas de *data centers* sustentáveis devem ser compostas pelos equipamentos com a maior eficiência energética a fim de se reduzir o impacto ambiental decorrente (Bash et al., 2008).

A utilização de ferramentas para modelagem de arquiteturas de *data centers* é de extrema importância no apoio aos projetistas desses sistemas na estimação do impacto ambiental, da disponibilidade e do custo associado antes mesmo de serem implementadas. Porém, atualmente, os projetistas de *data centers* possuem poucos mecanismos e ferramentas para suporte a avaliação das infraestruturas desses ambientes. Todavia, essas ferramentas exigem que os projetistas possuam um prévio conhecimento sobre os modelos (RBD - *Reliability Block Diagram*, EFM - *Energy Flow Model* e SPN - *Stochastic Petri Nets*) utilizados, o que acaba por dificultar a sua aplicação. Nesse contexto, este trabalho tem por objetivo a proposição de um ambiente, denominado Stars (*Stochastic Tool for Availability and Reliability System analysis*), para a modelagem de arquiteturas de *data centers* com suporte aos modelos formais RBD, EFM e SPN. Além disso, também é foco desta pesquisa a proposição de uma visão de alto nível que pode ser automaticamente convertida para a quantificação de métricas como: disponibilidade, custo e consumo energético sem a necessidade do projetista conhecer tais modelos. Esta pesquisa também propõe a utilização do algoritmo Genético para otimizar os resultados alcançados.

O restante desse artigo está dividido da seguinte forma. A Seção 2 mostra os trabalhos relacionados presentes na literatura. A Seção 3 apresenta a fundamentação teórica necessária para um adequado entendimento dessa pesquisa. Uma visão da ferramenta proposta é apresentada na Seção 4. A Seção 5 ilustra a aplicabilidade da ferramenta proposta através de um estudo de caso com um cenário real que modela arquiteturas de *data centers*. Esse estudo de caso faz uso do algoritmo genético implementado na ferramenta para otimizar os resultados obtidos através dos modelos. Por fim, a Seção 6 apresenta as conclusões e os futuros direcionamentos dessa pesquisa.

2 Trabalho Relacionados

Essa seção apresenta os trabalhos relacionados encontrados na literatura sobre modelagem de sistemas computacionais e, também, sobre o ferramental para suporte a tais modelos. Por exemplo, os autores propuseram em Nguyen et al. (2019) uma modelagem hierárquica para a avaliação de disponibilidade e confiabilidade em redes de *data centers* através de estruturas de árvore. O foco foi nas infraestruturas físicas dos *data centers*. Para se computar a disponibilidade e confiabilidade, foram usados modelos matemáticos como as redes de Petri estocásticas (SPN), as redes estocásticas de recompensa (SRN) e as cadeias de Markov (CTMC). Através de um sistema em três camadas, foi possível mostrar que a distribuição ativa de nós na rede conseguiu melhorar a disponibilidade e a confiabilidade em sistemas de computação em nuvem. Esse trabalho fez uso de modelagem com SPN para se computar a disponibilidade. No entanto, não foi o foco dos autores a avaliação da energia e nem a proposição de uma ferramenta para auxiliar na modelagem.

Em Ferreira et al. (2020) foi proposto um algoritmo para melhorar a distribuição do fluxo energético das infraestruturas elétricas de *data centers* PLDA-D (*Probabilistic Linear Discriminant Analysis*). Esse algoritmo é capaz de realizar automaticamente a distribuição dos fluxos nos modelos EFMs e, assim, otimizar o consumo energético do sistema. A ideia adotada foi a de priorizar na distribuição do fluxo os caminhos que fazem uso dos equipamentos com maior eficiência energética. Embora esse trabalho modele infraestrutura de *data center* elétricas, não foi o foco dos autores a propor de um ferramental que suporte otimização, manutenção e auxilie projetistas desses sistemas.

Em Liu et al. (2016), os autores propuseram uma nova abordagem para modelagem do serviço de confiabilidade de *data centers* em nuvem. Para isso, foi proposto um método com dois estágios: pedido e execução. A confiabilidade foi calculada através do modelo baseado numa infraestrutura simplificada. No entanto, não foi o foco dos

autores a criação de um ambiente para auxiliar projetistas na otimização da disponibilidade, por exemplo.

Uma modelagem utilizando programação linear inteira mista (MILP) para o planejamento de capacidade e minimização do custo total de propriedade (CTP) de *data centers* verdes de alta disponibilidade foi proposta em Tripathi et al. (2017). Os autores demonstraram que a integração com energia verde auxilia na minimização do CTP. Resultados obtidos demonstraram que foi possível melhorar a distribuição do servidor, cruzando os *sites*, e também se minimizou o CTP. Embora este trabalho tenha proposto uma forma de otimização de *data centers* utilizando um *framework*, não foi o foco desse trabalho o uso do algoritmo genético nem o desenvolvimento de um ferramental para auxiliar projetistas na modelagem de *data centers*.

Melo et al. (2020) propôs modelos em redes de Petri Estocásticas para aferir os efeitos de diferentes políticas de manutenção em *data centers*. As abordagens de manutenção analisadas foram a corretiva e a preventiva e ambas as políticas têm por objetivo evitar ou reduzir possíveis interrupções do sistema por falha dos equipamentos. A manutenção corretiva é realizada na ocorrência de falhas inesperadas no sistema, já a manutenção preventiva tem o objetivo de evitar a ocorrência de falhas e, assim, manutenções periódicas são realizadas. Os resultados evidenciaram que a aplicação de tais políticas foi capaz de elevar a disponibilidade do sistema sob análise. Esse artigo teve como foco a proposição de modelos SPN para analisar infraestruturas elétricas de *data center* com políticas de manutenção, mas não teve o foco na proposição de um ferramental para auxiliar projetistas de tais sistemas.

Austregésilo and Callou (2019) propuseram o uso de algoritmos genéticos para otimizar custo, impacto ambiental e disponibilidade da infraestrutura de energia elétrica de sistemas *data centers*. O objetivo é maximizar a disponibilidade e minimizar o custo total e a exergia operacional. Com o intuito de computar tais métricas, foram realizados dois estudos de caso para mostrar a aplicabilidade e validação da estratégia proposta. Além disso, também foi utilizada a estratégia de otimização. Em relação aos resultados obtidos, observou-se uma melhora significativa, ficando próximos ao ótimo, obtido por um algoritmo de força bruta que analisou todas as possibilidades. Os autores também observaram que com o uso do algoritmo genético o tempo para obtenção das respostas foi significantemente inferior. Esse trabalho fez uso de uma abordagem de otimização para *data centers* baseada em algoritmos genéticos, porém não foi o foco desse trabalho o desenvolvimento de um ferramental para auxiliar na modelagem desses sistemas de *data centers*.

Em Rampazzo et al. (2013), os autores propuseram uma estratégia de otimização que faz uso dos algoritmos Genético e Diferencial Evolutivo para auxiliar no planejamento de sistemas elétricos. Como resultado foi comprovado que as abordagens demonstraram grande potencial na resolução do problema de maximização da produção de energia elétrica, onde foram obtidos não somente um resultado próximo ao ótimo, mas também um conjunto de soluções distintas e de qualidade. Embora esse trabalho tenha avaliado sistemas elétricos e utilizado técnicas de otimização, não foi o foco dessa pesquisa o trabalho com sistemas elétricos de *data centers*.

Calheiros et al. (2011) propõe um *framework* para modelagem e simulações de infraestruturas em nuvem, chamado CloudSim *Toolkit*. Essa ferramenta permite avaliar a performance dessas infraestruturas em um ambiente controlado, além de analisar em tempo-real a reação do sistema diante de gargalos. Além disso, os autores também fizeram avaliações desse *framework* para comprovar sua capacidade de simulação através de métricas como consumo energético, custo e tempo de execução. Esse trabalho, apesar de envolver o desenvolvimento de um ferramental para modelagem, o qual pode ser usado também para *data centers*, não teve como foco a proposição de uma estratégia de modelagem que auxiliasse os projetistas, e também não fez uso de algoritmos de otimização.

Diferente dos trabalhos anteriores, esse trabalho propõe uma ferramenta, que faz uso de uma visão de alto nível, para a modelagem de arquiteturas de *data centers*. A partir dessa visão de alto nível, a ferramenta automaticamente converte para os modelos formais necessários para se computar as métricas de interesse (SPN, RBD, EFM). Além disso, o ferramental proposto também possui um módulo de otimização, onde o algoritmo genético foi implementado, e foi utilizado para otimizar as arquiteturas analisadas nesse trabalho. Os resultados dessa análise foram comparados com o algoritmo de força bruta. Por fim, a Tabela 1 ilustra uma comparação das estratégias de modelagem, das métricas adotadas e das técnicas de otimização utilizadas em cada artigo.

3 Fundamentação Teórica

Nessa seção são explanados conceitos e conhecimentos necessários para um melhor entendimento desse trabalho.

3.1 *Data centers*

Os *data centers* são ambientes computacionais conhecidos por abrigar equipamentos responsáveis pelo processamento e armazenamento de informações imprescindíveis para a continuidade de negócios das mais diversas organizações, sejam elas empresas, instituições de ensino, indústrias, órgãos governamentais, hospitais, hotéis, entre outros (Marin, 2011). *Data centers* são comumente utilizados como host para *sites* com o objetivo de processar transações comerciais, para proteger dados (por exemplo, registros financeiros, histórico médico de pacientes, gerenciar e-mails). Além disso, as organizações estão procurando por departamentos de TI com o objetivo de receber suporte em muitas fontes de negócios, como produtividade e receita. Portanto, os *data center* devem ter altos níveis de disponibilidade para suportar novos requisitos de tecnologia. O grande problema é que os *data centers* são ambientes dinâmicos onde equipamentos vão se tornando obsoletos e são substituídos; novos equipamentos, que demandam novas interfaces e novos tipos de conexões, são criados para atender as demandas dos novos clientes e das novas necessidades que surgem dia após dia (W. Zucchi, 2013).

Os *data centers* desempenham uma função vital na atualidade, dada a grande quantidade de informações digitais que eles armazenam. De acordo com os padrões, as melho-

Tabela 1: Características dos Trabalhos Relacionados.

Artigo	Modelagem	Métricas	Otimização
(Nguyen et al., 2019)	SPN,SRN,CTMC	disponibilidade, confiabilidade	-
(Ferreira et al., 2020)	EFM	consumo energético	PLDA-D
(Liu et al., 2016)	novo modelo de confiabilidade	confiabilidade	-
(Tripathi et al., 2017)	MILP	custo, disponibilidade	*framework*
(Melo et al., 2020)	SPN	disponibilidade	-
(Austregésilo and Callou, 2019)	RBD	disponibilidade	Alg. Genético
(Rampazzo et al., 2013)	Modelos	Energia	Alg. Genético, Alg. Diferencial Evolucionário
(Calheiros et al., 2011)	Modelos	energético, custo, tempo de execução	-
Este Trabalho	SPN,RBD, EFM	disponibilidade	Alg. Genético

res práticas e os requisitos dos usuários, a infraestrutura do *data center* deve atender a requisitos técnicos rigorosos para garantir a confiabilidade, a disponibilidade e a segurança, pois eles têm um impacto direto no custo e na eficiência. Uma infraestrutura com alta confiabilidade e disponibilidade deve possuir redundância do sistema, e por esse motivo, será mais caro e provavelmente irá gerar um maior consumo de energia elétrica (Levy and Hallstrom, 2017).

Ao se projetar uma sala de *data center*, muitas infraestruturas devem ser analisadas como por exemplo: energia, refrigeração, conectividade, espaço físico, proteções (por exemplo, contra fogo) e monitoramento de temperatura. Além disso, os projetistas dos *data centers* devem se concentrar no aumento da produtividade e evitar o tempo de inatividade, e as estratégias de projeto são fundamentais para se atingir essas necessidades. De uma forma simplificada, as infraestruturas dos *data centers* podem ser compostas por fontes de energia de reserva, que são responsáveis por suportar a carga elétrica do *data center* quando a energia primária falhar; infraestrutura de rede redundante para manter o sistema funcionando em caso de ocorrer uma falha de um dispositivo de rede; servidores de aplicações e de dados redundantes.

3.2 Redes de Petri

O conceito de redes de Petri surgiu em 1962 na tese de doutorado de Carl Adam Petri, na faculdade de Matemática e Física da Universidade Darmstadt, na Alemanha (Murata, 1989). As redes de Petri são uma técnica de especificação de sistemas que permite uma representação matemática utilizada para modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (Maciel et al., 1996).

A rede de Petri (PN) é composta por 4 elementos: lugares, transições, arcos e tokens. Os tokens presentes nos lugares representam o estado do sistema. As transições são ações realizadas para alterar o estado do sistema. Para uma transição está habilitada, é preciso que as precondições sejam satisfeitas. Os arcos representam o fluxo de tokens pela rede, e a distribuição dos tokens ao longo da rede representa o estado em que o sistema se encontra. Graficamente, os lugares são representados por círculos, as transições por barras, os arcos por setas, e os tokens por pontos. A Fig. 1 ilustra os elementos da rede de Petri.

Figura 1: Elementos da rede de Petri

A Fig. 2 mostra um exemplo de uma PN que modela o funcionamento de um sistema. Nessa rede, os lugares representam o estado do sistema (*on* ou *off*) e as transições representam as ações que alteram o estado do sistema (*ligar* e *desligar*). A Fig. 2 (a) representa o sistema ligado, pois possui um token no lugar *on*. Nesse momento, a única transição habilitada para ser disparada é a transição *desligar*. Após o disparo dessa transição, o modelo passará para o estado desligado, e irá possuir um token no lugar *off* (Fig. 2 (b)). Após isso, a transição habilitada para disparar é a *ligar*, e o disparo dessa fará o sistema voltar ao estado de ligado como na Fig. 2 (a). Existem muitas extensões das redes de Petri, e esse trabalho faz uso de uma extensão denominada redes de Petri estocásticas (SPN).

3.3 Redes de Petri Estocásticas

As redes de Petri estocásticas representam uma extensão das PNs apresentadas anteriormente, pois adicionam a possibilidade de se modelar o tempo em que o sistema leva para mudar de um estado para outra. Sendo assim, um elemento denominado por transição temporizada é adicionado nas SPNs. Essa transição temporizada utiliza a noção de tempo, de modo que o disparo da transição corresponde ao período de execução da atividade.

3.4 Diagramas de Bloco de Confiabilidade

Os Diagramas de Bloco de Confiabilidade, do inglês *Reliability Block Diagram* (RBD), foram inicialmente propostos como uma técnica para calcular a confiabilidade de sistemas. Atualmente, também são utilizados para calcular a disponibilidade. Em um RBD, os componentes são representados por blocos, interligados uns com os outros. Esses blocos podem representar composições: em série,

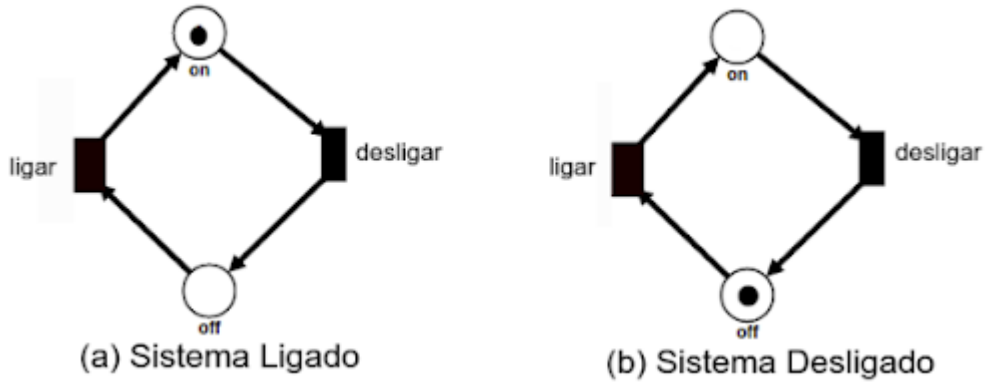

Figura 2: Exemplo de uma rede de Petri.

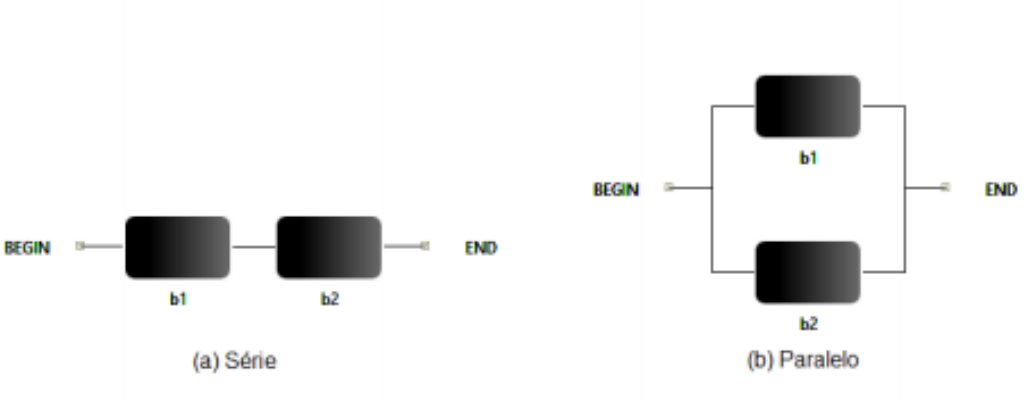

Figura 3: Diagramas de Bloco de Confiabilidade

em paralelo ou combinações dessas (Trivedi et al., 1996).

A Fig. 3 ilustra dois exemplos de modelos em RBD. Na Fig. 3 (a), os blocos são organizados em série; e na Fig. 3 (b), os blocos são organizados em paralelo. No sistema em série, quando um componente falhar, o sistema todo irá parar. Já no sistema em paralelo, o sistema só irá falhar se ambos os blocos falharem.

3.5 Modelo de Fluxo de Energia

O Modelo de Fluxo de Energia (EFM) tem por objetivo representar o fluxo de energia elétrica entre os componentes que representam o sistema sob análise. Esse modelo considera também a eficiência e a capacidade máxima que cada componente pode fornecer ou extrair (Callou et al., 2014). O EFM é representado por um grafo acíclico dirigido em que os componentes são modelados como vértices e as respectivas ligações correspondem às arestas.

A Fig. 4 ilustra um exemplo de um EFM que representa o fluxo entre componentes de um sistema de potência de *data centers*. O peso existente nas arestas é utilizado para direcionar o fluxo. A representação gráfica do EFM tem um (1) por peso padrão nas arestas. Nesse modelo, os retângulos representam o tipo do dispositivo, e as etiquetas correspondem ao nome de cada dispositivo. O EFM pode ser utilizado para calcular a energia necessária para fornecer a potência demandada pelo ambiente de TI (representada no *TargetPoint1* da Fig. 4). Além disso, o EFM também computa tanto o custo de aquisição como o custo operacional do ambiente sob análise.

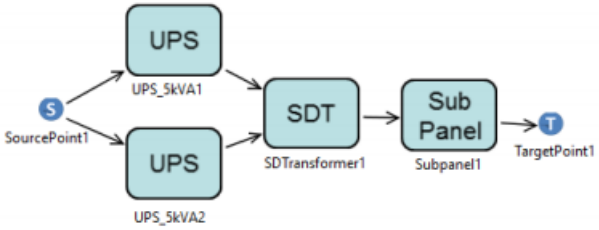

Figura 4: Exemplo de um EFM.

3.6 Otimização

Otimização é uma técnica utilizada para se buscar encontrar a melhor solução possível respeitando as restrições dadas (Rao, 2009). As técnicas de otimização possuem um amplo leque de aplicações, levando em consideração que a maioria das empresas está engajada

na resolução de problemas de otimização. Muitos dos problemas práticos e teóricos em engenharia, economia e planejamento podem ser modelados como problemas de otimização. Por exemplo, empresas de telecomunicações interessadas em otimizar o custo e a qualidade do serviço em seu projeto das redes de comunicação; empresas do mercado financeiro têm o objetivo de obter a melhor estratégia de alocação de ativos para otimizar o lucro e o risco; empresas de distribuição que visam otimizarem a alocação das entregas (rotas) de forma a minimizar a distância percorrida pelos veículos (Nogueira, 2015). Dentre as várias técnicas de otimização existentes, esse trabalho irá fazer uso do algoritmo genético.

3.7 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são algoritmos de otimização baseados na lógica da seleção e genética natural. Embora randomizados, os algoritmos genéticos não são uma caminhada aleatória simples. De maneira eficiente, eles exploram informações históricas para especular sobre novos pontos de busca com desempenho melhorado (Goldberg, 1989). Algoritmos genéticos foram desenvolvidos por Holland (1975), seus colegas e seus alunos na Universidade de Michigan. Os objetivos de sua pesquisa foram: abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais e projetar *software* de sistemas artificiais que possua mecanismos de sistemas naturais significativos (Goldberg, 1989).

Os algoritmos genéticos foram propostos com o intuito de aplicar a teoria da evolução das espécies de Darwin na computação. Esses algoritmos usam conceitos da evolução biológica como genes, cromossomos, cruzamento, mutação e seleção, procurando aprofundar o conhecimento em processos de adaptação em sistemas naturais e, baseado neles, desenvolver sistemas artificiais (simulações computacionais) que mantenham a mesma lógica dos mecanismos originais (Oliveira, 2005). A lógica funciona de forma semelhante ao processo de cruzamento biológico de cromossomos que compartilham informação genética para criar um novo indivíduo. No fim, chegam a obter um indivíduo de alta adaptação, que não significa uma solução ótima, mas sim a "melhor solução" encontrada (Goldberg, 1989).

Um algoritmo genético passa pelas seguintes etapas (Lucas, 2002): (i) codificação do indivíduo, fase onde se cria um indivíduo para representar uma solução do problema (Pappa, 2002); (ii) inicialização, etapa que produz uma população de soluções aleatórias (indivíduos) para o problema a ser otimizado; (iii) avaliação, nessa etapa se verifica a aptidão dos indivíduos através de uma função *fitness*, ou função objetivo, para se estabelecer a qualidade de cada solução gerada resultando numa pontuação para cada indivíduo; (iv) seleção, nessa etapa indivíduos são selecionados para a reprodução a partir de sua aptidão; (v) cruzamento, essa fase é responsável por recombinar características das soluções, gerando novos indivíduos.

4 Stars: uma visão da ferramenta

A ferramenta Stars (*Stochastic Tool for Availability and Reliability System analysis*) surgiu com o intuito de interligar uma visão de alto nível, proposta para representar arquiteturas de *data centers*, com os modelos formais responsáveis por calcular as métricas como disponibilidade, custo e consumo energético. A ferramenta foi desenvolvida em Java, utilizando as bibliotecas JAVA(Swing) e JGraphX. A Fig. 5 mostra como é feita a integração da ferramenta proposta com outras ferramentas computacionais já existentes. Vale ressaltar que as avaliações dos modelos são realizadas a partir de outras ferramentas já existentes como, por exemplo, o CPN Tools (Ratzer et al., 2003) e o Mercury (Oliveira et al., 2017). É possível observar que a ferramenta foi implementada em camadas, existindo as camadas da visão de alto nível, a do parser, e os módulos de otimização e de manutenção. Esse artigo apresenta uma visão geral do funcionamento e, posteriormente, foca na apresentação detalhada do funcionamento da camada do parser.

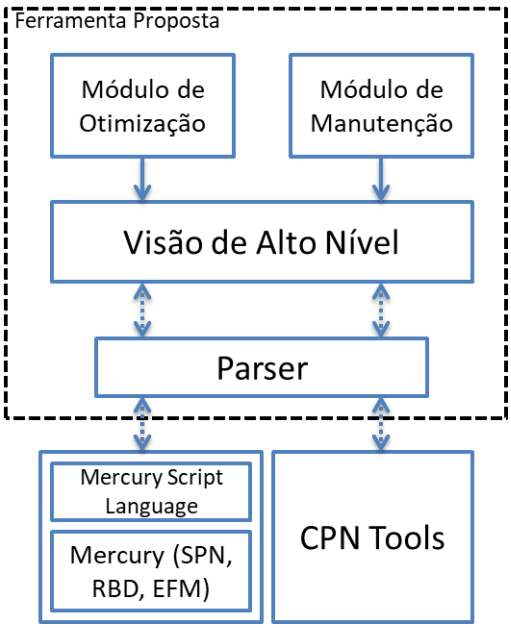

Figura 5: Metodologia adotada.

A camada do parser é responsável pela conversão do modelo gerado pela ferramenta proposta para os modelos formais. Essa conversão é feita de acordo com as métricas que se deseja avaliar. Por exemplo, se o projetista desejar computar a disponibilidade, o parser terá que realizar a conversão do modelo proposto na ferramenta para o formalismo RBD ou SPN. Essa conversão é feita através da representação na linguagem de *script* do Mercury (Oliveira et al., 2017). 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,

poder realizar ajustes e converter pelo parser para modelos em SPN de manutenção preditiva e corretiva, por exemplo.

É importante mencionar a aplicabilidade do ambiente proposto, tendo em vista que a partir dessa ferramenta será possível desenvolver métodos de otimização que fazem uso de formalismos distintos. Dessa forma, técnicas multiobjetivo de otimização que façam uso de forma integrada dos formalismos de redes de Petri coloridas (CPN), redes de Petri estocásticas (SPN), modelo de fluxo de energia (EFM), por exemplo, poderão vir a serem implementadas de forma automatizada com a ferramenta proposta. A seguir, mais detalhes serão fornecidos sobre a visão de alto nível e sobre o parser.

4.1 Visão de Alto Nível

A ferramenta desenvolvida conta com uma visão única da representação dos sistemas computacionais de interesse. O foco dessa pesquisa consiste em utilizar diferentes formalismos para se computar diferentes métricas (custo, disponibilidade, consumo energético, etc.). A Fig. 6 mostra uma visualização de uma arquitetura de *data centers* na ferramenta proposta. Nessa figura, cada um dos retângulos representa um equipamento (componente) de um *data center* e as arestas representam as ligações físicas entre eles. O ferramental proposto fornece suporte a conversões para modelos em SPN, RBD e EFM.

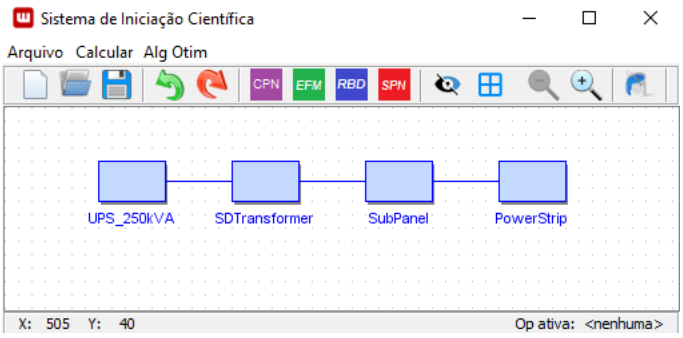

Figura 6: Ferramenta Proposta.

Cada um dos componentes representados possui um conjunto de informações internas e não mostradas na Fig. 6. Essas informações são utilizadas para se calcular as métricas desejadas. Dentre os formalismos para os quais é possível se traduzir o modelo de alto nível, o EFM exige algumas informações para o cálculo de suas métricas como a especificação da energia demandada pelo ambiente de TI, a eficiência energética, o custo de aquisição, e o tempo médio de falha (MTTF) de cada dispositivo.

Essa ferramenta proposta traduz, automaticamente, o modelo de alto nível para os formalismos RBD, SPN e EFM no padrão adotado pela linguagem de *script* do Mercury. O formalismo a ser utilizado vai depender das métricas que se deseja computar e também do arranjamento dos equipamentos. A Fig. 7 mostra um modelo representado no formato da linguagem de *script* do Mercury. A avaliação desse *script* é utilizada para se poder computar as métricas de interesse. Uma vez feita a avaliação, os resultados são obtidos e mostrados ao usuário.

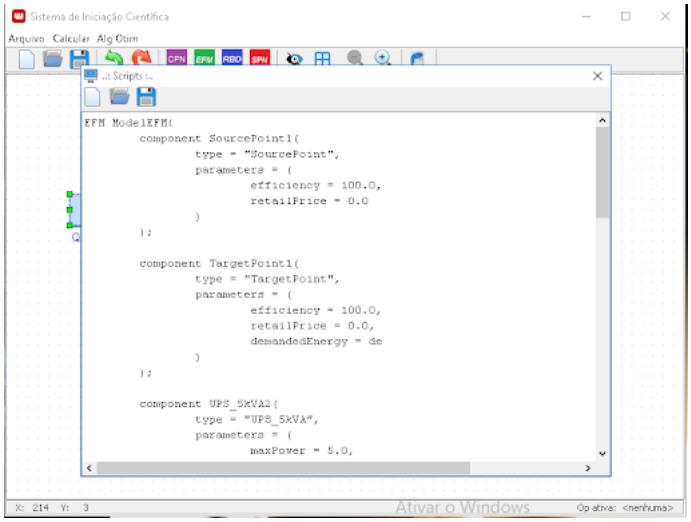

Figura 7: Exemplo de *script*.

Além disso, o Módulo de Otimização é capaz de utilizar os resultados de cada avaliação, realizar ajustes no modelo (troca de componentes com diferentes parâmetros para o custo, eficiência energética, etc.), e avaliar novamente. Esse processo pode ser repetido até se chegar ao critério de parada do algoritmo de otimização utilizado. No presente momento, o Módulo de Otimização conta com a implementação do Algoritmo Genético. A seguir, mais detalhes são mostrados sobre o processo de conversão do modelo de alto nível para os diferentes formalismos utilizados.

4.2 Parser

O parser é responsável por converter os modelos da visão de alto nível da ferramenta para os padrões dos ambientes Mercury *Script Language* e CPN Tools. Esse trabalho foca na conversão para o ambiente de modelagem com o Mercury, que dispõe da linguagem de *script* para representar os formalismos RBD, SPN e EFM. Para realizar tal conversão, o parser faz uso de um algoritmo baseado em busca em profundidade. Basicamente, podemos dividir o parser em três módulos. Cada um desses módulos corresponde ao respectivo tradutor para o *script* dos formalismos RBD, SPN e EFM. Cada módulo constitui-se basicamente das mesmas partes para conversão no *script*, utilizando como entrada de dados o modelo de alto nível, gerando como saída o *script* no respectivo formalismo.

A fim de facilitar a explicação do processo de conversão, o padrão da linguagem de *script* do Mercury foi dividido em três partes: início, meio e fim. Essa divisão é válida para todos os módulos (RBD, SPN ou EFM), porém existem particularidades para cada formalismo. As partes de início e fim serão sempre constantes e independentes da arquitetura que está sendo modelada. Ressaltamos, todavia, que

a parte de início e fim dependem para qual formalismo será convertido. No entanto, a parte intermediaria (meio) sempre irá variar de acordo com a arquitetura sob análise. Assim, o parser fará a conversão da visão de alto nível para o padrão existente na parte intermediária e, em seguida, irá concatenar com as demais que compõe o *script* resultante. A seguir, detalhamos o algoritmo utilizado para fazer a geração dos *scripts* realizado pelo parser.

4.2.1 Geração do Script

A geração do *script* da linguagem do Mercury inicia com a chamada do Algoritmo 1. Esse algoritmo recebe como parâmetro uma lista de vértices que representa os equipamentos que compõe a arquitetura em análise. O algoritmo inicia com a criação das variáveis *início* e *fim* (linha 1 e 2), responsáveis por armazenar as partes constantes da linguagem de *script* (cabeçalho e rodapé) no padrão adotado pela ferramenta Mercury.

O algoritmo procede com a inicialização da variável *caminhos* (linha 3) com uma *string* vazia. Esta variável é utilizada para armazenar os caminhos do vértice inicial até o final. Posteriormente, é criada a variável *equipamentos* que armazenará as informações necessárias, de cada equipamento, para o cálculo das métricas. Essas duas variáveis compõem a parte intermediária do *script* resultante.

O algoritmo então prossegue fazendo uma varredura a partir de todos os vértices passados como parâmetro (Linha 5). Em seguida, a Linha 6 chama o método *montarEquipamento*, cujo retorno será armazenado dentro da variável *equipamentos*. Esse método reescreve as informações do equipamento para o padrão de *script* do respectivo formalismo (ex., SPN, RBD, ou EFM). O passo seguinte, Linha 7, faz uma chamada ao método *identificaInicial* para poder identificar se o vértice *i* é um vértice inicial. Em caso afirmativo, gera-se os caminhos possíveis considerando esse vértice como ponto de partida. Esse método *gerarCaminhos* é baseado na busca em profundidade para poder percorrer e listar todos os caminhos existentes no grafo a partir do vértice passado como parâmetro. Os percursos gerados são retornados e armazenados na variável *caminhos*.

Por fim, o algoritmo retorna a concatenação das variáveis *inicio*, *equipamentos*, *caminhos* e *fim* (Linha 12). Esse retorno corresponde ao *script* já no padrão demandado pelo Mercury. A seguir, os métodos chamados pelo Algoritmo 1 serão detalhados.

4.2.2 Montagem do Equipamento

O Algoritmo 2 *montarEquipamento* é responsável pela montagem do equipamento para cada formalismo desejado. Esse algoritmo recebe como parâmetro um vértice (equipamento) do grafo analisado. A Linha 1 inicia o algoritmo com a criação da variável *bloco* que recebe as informações do equipamento (ex., identificação). Em seguida, a variável *bloco* receberá os parâmetros de acordo com o módulo do formalismo (ex., SPN, RBD, EFM) ao qual será feita a conversão. Sendo assim, esse método é dependente do formalismo de destino. Por último, a variável bloco é retornada (linha 3).

Algoritmo 1 *gerarScript(vertices)*

```
1:  inicio := textoCabealho;
2:  fim := textoRodadpe;
3:  caminhos := "";
4:  equipamentos := "";
5:  for i in vertices do
6:      equipamentos := equipamentos +
        montarEquipamento(i);
7:      inicial := identificaInicial(i);
8:      if (inicial == true) then
9:          caminhos := gerarCaminhos("", i);
10:     end if
11: end for
12: return (inicio + equipamentos + caminhos + fim);
```

Algoritmo 2 *montarEquipamento(vertice)*

```
1: bloco := "eqp" + vertice.getId();
2: bloco := bloco+\n + vertice.getParametros();
3: return bloco;
```

4.2.3 Identificação de Vértice Inicial

O Algoritmo 3 *identificarInicial* identifica se o vértice recebido como parâmetro corresponde a um vértice inicial do modelo sob análise. Vale destacar que mais de um vértice inicial pode está presente no modelo. Esse algoritmo testa o seguinte princípio, onde um vértice é considerado inicial se não temos nenhuma aresta chegando nele. Esse algoritmo começa (Linha 1) inicializando a variável *maximo* com a quantidade de arestas do equipamento (vértice) passado como parâmetro. Em seguida, a Linha 2 inicializa a variável booleana *ehInicial* com *true*. Essa variável será retornada ao final do algoritmo, indicando se corresponde a um vértice inicial ou não.

O algoritmo continua através da atribuição de zero à variável *contador*. Depois, segue-se para um laço onde todas as arestas desse vértice são percorridas (Linha 4). A Linha 5 recebe uma aresta ainda não percorrida do vértice sendo analisado. A partir dessa aresta, é possível saber o vértice de destino dela. Então, a Linha 6 recebe dessa aresta o vértice do grafo alvo ou de destino. Em seguida, é verificado se o alvo da aresta corresponde ao vértice passado como parâmetro ao modelo. O leitor deve recordar que isso é feito para se verificar que um vértice inicial não pode ter nenhuma aresta chegando nele. Caso o alvo seja correspondente ao vértice, esse vértice não é um vértice inicial. Sendo assim, a variável *ehInicial* tem seu valor configurado para *false* (Linha 8), encerrando a análise (Linha 9). Por fim, a variável *ehInicial* é retornada (Linha 13). Caso a Linha 13 retorne *true*, isso indica que o vértice passado por parâmetro corresponde a um vértice inicial. Caso contrário, não é um vértice inicial.

4.2.4 Geração de Caminhos

O Algoritmo 4 tem como parâmetros de entrada o *percurso* feito a partir do vértice inicial até o vértice atual, O vértice recebido como parâmetro do algoritmo corresponde ao vértice atual. Esse algoritmo inicia atualizando a variável *percurso* com a adição da representação do equipamento

Algoritmo 3 *identificarInicial(vertice)*

```
1:  maximo := vertice.getEdgeCount();
2:  ehInicial := true;
3:  contador := 0;
4:  while (contador < maximo) do
5:     aresta := vertice.getEdgeAt(contador);
6:     alvo := aresta.getTarget();
7:     if (alvo == vertice.getId()) then
8:        ehInicial := false;
9:        break;
10:    end if
11:    contador := +contador + 1;
12: end while
13: return ehInicial;
```

Algoritmo 4 *gerarCaminhos(percurs, vertice)*

```
1:  idVertice := vertice.getId();
2:  percurso := percurs + "eqp" + idVertice;
3:  contador := 0;
4:  maximo := vertice.getEdgeCount();
5:  ehFinal := True;
6:  caminhos := "";
7:  while (contador < maximo) do
8:     edge := vertice.getEdgeAt(contador);
9:     target := edge.getTarget();
10:    if not(target.getId() == idVertice) then
11:       ehFinal := False;
12:       caminhos := caminhos + gerarCaminhos(percurso +
          " - - > ", target) + "\n";
13:    end if
14:    contador := contador + 1;
15: end while
16: if (ehFinal == True) then
17:    percurso := percurso + "\n";
18:    return percurso;
19: end if
20: return caminhos;
```

do *vertice* (linha 2). Após isso, as variáveis *maximo* e *ehFinal* são inicializadas (linhas 4 e 5). A variável *maximo* armazena a quantidade de ligações do *vertice*. Já a variável *ehFinal* corresponde a um booleano utilizado para informar se o equipamento é ou não final. Em seguida, a variável *caminhos* (linha 6), a qual irá armazenar todos os percursos de um equipamento inicial até os equipamentos finais, é inicializada com uma *string* vazia.

O algoritmo prossegue com um laço que faz a verificação de todas as ligações desse equipamento(vértice). A Linha 8 recebe uma aresta do *vertice*. A Linha 9 recebe o equipamento alvo dessa aresta. Em seguida, é verificado se o alvo é o equipamento contido na variável *vertice* (linha 10). Em caso negativo, a variável *ehFinal* tem o valor setado para *false* (linha 11). O passo seguinte (linha 12) é setar a variável *caminhos* que recebe o retorno da chamada recursiva ao método *gerarCaminhos*. Nessa chamada, são passados como parâmetros as variáveis *percurso* concatenado com a *string* " - - > " (representação da ligação na linguagem de *script*) e *target* (alvo).

Caso ao terminar a análise de todas as ligações a variável *ehFinal* seja verdadeira (linha 16), significa que durante a recursão chegou-se a um vértice final. Logo, a variável *percurso* receberá uma quebra de linha (linha 17) e será retornada (linha 18). A volta da recursão irá setar a variável *caminhos* que receberá o acréscimo do último caminho percorrido. Ao final, todos os caminhos a partir desse de *vertice* serão retornados.

5 Estudo de Caso

Essa seção apresenta um estudo de caso que tem como objetivo ilustrar a aplicabilidade da ferramenta proposta. Nesse estudo, arquiteturas de *data centers* são avaliadas a fim de se otimizar a métrica de disponibilidade. O algoritmo de otimização utilizado nesse estudo foi o algoritmo genético. Sendo assim, esse estudo também realiza a comparação entre o algoritmo de força bruta e o algoritmo genético a fim de mostrar que soluções muito próximas da ótima são obtidas em um tempo de avaliação bem reduzido. A escolha da comparação com o algoritmo de força bruta se deve pelo fato de que esse algoritmo analisa todas as possibilidades possíveis, levando em consideração toda a base de equipamentos disponíveis para se montar as arquiteturas de *data center* que se deseja otimizar. Assim, esse estudo compara o tempo de execução e a eficácia do algoritmo genético implementado na ferramenta proposta.

Cinco arquiteturas típicas de *data centers* (Avelar, 2003), denominadas nesse trabalho por A1, A2, A3, A4 e A5, com graus de complexidade diferentes, foram adotadas. A Fig. 8 mostra essas cinco arquiteturas elétricas de *data centers* modeladas na ferramenta proposta. É importante destacar a similaridade dessa representação de alto nível adotada pela ferramenta e a utilizada na literatura por projetistas de *data center* (Avelar, 2003). A arquitetura A1 é composta pelos seguintes equipamentos: UPS, transformador (SDTransformer), painel de energia (Subpanel), régua de tomadas (PowerStrip). A arquitetura A2 corresponde a arquitetura anterior acrescida de um UPS redundante e um chaveador estático (Static Transfer switch - STS). Na arquitetura A3 é adicionado um transformador(SDTransformer2). A arquitetura A4 considera a arquitetura anterior acrescida de um painel de energia redundante (Subpanel2). Por fim, A5 tem todos os caminhos redundantes.

A Fig. 8 mostra os modelos representados na visão de alto nível da ferramenta. A partir desses modelos, pode-se conduzir a avaliação em SPN ou RBD, por exemplo, para se obter a métrica de disponibilidade. Além disso, é importante destacar que foi criada uma base de dados, com 5 equipamentos de cada tipo, para simular a variedade de equipamentos do mundo real. Os parâmetros de entrada para cada componente dessa base foram gerados a partir das amplitudes mostrados na Tabela 2.

Durante a execução do algoritmo genético, cada cromossomo gerado se ajusta ao modelo RBD, gerado automaticamente pela ferramenta, para se obter a disponibilidade. A avaliação do modelo RBD é transparente. De posse desse resultado da disponibilidade, a pontuação do cromossomo é realizada.

A Tabela 3 mostra a comparação dos resultados obtidos através do algoritmo de força bruta e dos resultados alcançados a partir do algoritmo genético. Esses resulta-

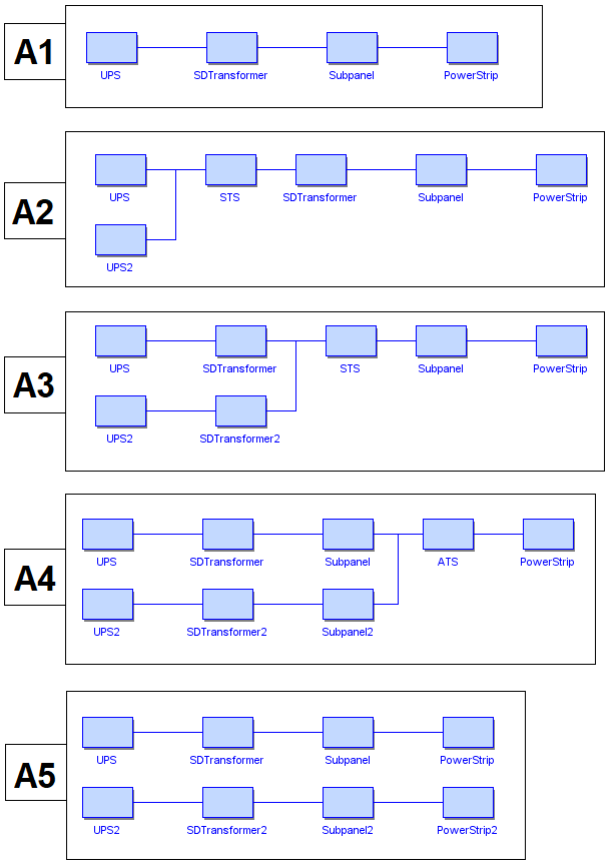

Figura 8: Modelos das arquiteturas na ferramenta proposta.

dos foram obtidos a partir de uma execução de ambos os algoritmos para cada arquitetura avaliada. Vale destacar que o algoritmo de força bruta retorna sempre o mesmo resultado para a disponibilidade, visto que ele testa todas as possibilidades possíveis e retorna o melhor resultado. Enquanto o algoritmo genético possui uma variação. Todavia, é importante destacar que uma execução do algoritmo genético representa a execução do processo que testa a combinação de uma gama de soluções e retorna o melhor conjunto de soluções obtidos. Nessa tabela, Disp. (9s) representa a disponibilidade em número de noves da arquitetura (-log(1 - disponibilidade)) e T. Exec. corresponde ao tempo demandado para realizar a execução de cada algoritmo e se obter os resultados.

É possível observar que os resultados obtidos através da otimização com o algoritmo genético são muito próximos dos resultados obtidos por força bruta. No entanto, o tempo gasto para se executar cada uma das duas técnicas é bem distinto. A medida que a complexidade das arquiteturas aumenta, o tempo gasto para otimizar com força bruta cresce exponencialmente. Já no caso da otimização através do algoritmo genético, o tempo gasto permanece aproximadamente o mesmo, e significativamente inferior ao tempo gasto pela força bruta. Observando a Tabela 3,

Tabela 2: Parâmetro dos equipamentos adotados

Equipamentos	MTTF (h)	MTTR (h)
UPS	[25.000; 75.000]	8,0
STS	[24.038; 72.114]	6,0
ATS	[300.000;750.000]	0,33
Subpanel	[152.000; 456.000]	2,4
S. Down Transformer	[141.290,5; 423.871,5]	156,01
Power Strip	[115.111,8; 345.335,3]	3,8

Tabela 3: Comparativo entre algoritmo genético e força bruta.

	Força Bruta		Algoritmo Genético	
Arquiteturas	Disp. (9s)	T. Exec.	Disp. (9s)	T. Exec.
A1	3,177	2,5s	3,171	2,2s
A2	3,203	11,1s	3,199	1,23s
A3	3,922	55,1s	3,921	1,27s
A4	4,894	260,3s	4,893	1,29s
A5	6,354	257,2s	6,327	1,31s

pode-se perceber que o tempo gasto para se otimizar as arquiteturas utilizando algoritmo genético foi sempre inferior a 3 segundos. Já para o algoritmo de força bruta, chegou-se a obter tempos superiores a 255 segundos de avaliação.

É importante ressaltar que todos esses resultados foram obtidos através da utilização da ferramenta Stars que foi a responsável por converter os modelos propostos nos modelos formais (RBD, EFM) utilizados para se obter os resultados. Assumindo ambientes de complexidades superiores, por exemplo, ambientes em que a falha de um equipamento venha a acionar um outro equipamento de *backup* que estava desligado, nesse caso, existe a necessidade de se modelar usando SPN. Novamente, a ferramenta Stars irá prover suporte, mas o importante a destacar é que o tempo demandado para simular ou avaliar modelos SPN é bem superior aos modelos em RBD. A avaliação desse sistema descrito utilizando força bruta irá demandar um tempo demasiadamente elevado, inviabilizando tal abordagem. Dessa forma, a utilização de técnicas de otimização são de extrema importância para se reduzir o número de cenários a serem analisados e, mesmo assim, se obter resultados satisfatórios em um tempo reduzido.

6 Conclusão

Devido a crescente demanda de toda sociedade global por serviços de Internet, a disponibilidade dos *data centers* que dão suporte a tais serviços vem se tornando cada vez mais indispensável. Não apenas usuários em seu dia a dia, mas também empresas de todo porte dependem, de maneira ininterrupta, que esses serviços estejam sempre em funcionamento para realizar suas operações. Isso significa que qualquer interrupção nos serviços de *data centers* pode causar grandes prejuízos a muitas empresas, simultanea-

mente, deixando a economia passível de sofrer impactos. Nesse contexto, esse trabalho fez a proposição de uma ferramenta para auxiliar projetistas a modelar arquiteturas de *data centers*, a partir de uma visão de alto nível, sem a necessidade de se conhecer o formalismo a ser usado para a obtenção das métricas de interesse, por exemplo, disponibilidade.

A ferramenta Stars proposta permite a tradução do modelo de alto nível para os formalismos RBD, SPN e EFM, utilizando a linguagem de *script* do Mercury. A partir do desenvolvimento do ferramental, foi possível progredir para outra etapa da pesquisa que correspondeu à integração da ferramenta com algoritmos de otimização. Esses algoritmos de otimização fazem uso dos modelos na visão de alto nível proposta, que são automaticamente convertidos para os mesmos formalismos (RBD, SPN e EFM). Um estudo de caso foi apresentado mostrando a aplicabilidade da ferramenta Stars utilizando a técnica de otimização do algoritmo genético, onde foi possível comparar o tempo de execução dessa estratégia com o algoritmo de força bruta. As soluções obtidas foram satisfatórias e em um tempo acima de 200 vezes inferior.

Em relação a trabalhos futuros, deseja-se implementar outros métodos de otimização como, por exemplo, o GRASP e evolução diferencial. Além disso, também iremos realizar a modelagem de outras arquiteturas para mostrar a aplicabilidade das estratégias de otimização em cenários mais complexos e fazendo uso de mais métricas (otimização multiobjetivo).

Agradecimentos

Os autores gostariam de agradecer ao CNPq e a FACEPE pelo apoio para a realização desta pesquisa.

Referências

Austregésilo, M. S. S. and Callou, G. (2019). Stochastic models for optimizing availability, cost and sustainability of data center power architectures through genetic algorithm, *Revista de Informática Teórica e Aplicada* **26**(2): 27–44. https://doi.org/10.22456/2175-2745.83498.

Avelar, V. (2003). Comparing availability of various rack power redundancy configurations, *APC White Paper* **48**: 1–22.

Bash, C. E., Patel, C. D., Shah, A. J. and Sharma, R. K. (2008). The sustainable information technology ecosystem, *2008 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pp. 1126–1131. https://doi.org/10.1109/ITHERM.2008.4544387.

Bell, S. and Morse, S. (2000). Sustainability indicators: Measuring the immeasurable, *Journal of Rural Studies* **16**. https://doi.org/10.1016/S0743-0167(99)00036-4.

Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C. and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software Practice and Experience* **41**: 23–50. https://doi.org/10.1002/spe.995.

Callou, G., Ferreira, J., Maciel, P., Tutsch, D. and Souza, R. (2014). An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost, *Energies* **7**(1): 238–277. https://doi.org/10.3390/en7010238.

Ferreira, J., Callou, G., Maciel, P. and Tutsch, D. (2020). An algorithm to optimise the energy distribution of data centre electrical infrastructures, *International Journal of Grid and Utility Computing* **11**(3): 419–433. https://doi.org/10.1504/IJGUC.2020.107625.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. https://doi.org/10.1023/A:1022602019183.

Holland, J. H. (1975). *Adaptation in Natural and Artificial System*, University of Michigan. Disponpivel em https://ieeexplore.ieee.org/servlet/opac?bknumber=6267401.

Levy, M. and Hallstrom, J. O. (2017). A new approach to data center infrastructure monitoring and management (dcimm), *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–6. https://doi.org/10.1109/CCWC.2017.7868412.

Liu, Y., Li, X., Kang, R. and Xiao, L. (2016). Service reliability modeling of the it infrastructure of active-active cloud data center, *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, pp. 1–7. https://doi.org/10.1109/PHM.2016.7819903.

Lucas, D. (2002). Algoritmos genéticos: uma introdução. Universidade Federal do Rio Grande do Sul, . Apostila elaborada sob a orientação de Luis Otavio Alvares, para a disciplina de Ferramentas de Inteligência Artificial.

Maciel, P., Lins, R. D. and Cunha, P. R. F. (1996). *Introdução às redes de petri e aplicações*, X Escola de Computação, Campinas - SP:.

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

Melo, F. F., Junior, J. S. and de Almeida Callou, G. (2020). Evaluating the impact of maintenance policies associated to sla contracts on the dependability of data centers electrical infrastructures, *Revista de Informática Teórica e Aplicada* **27**(1): 13–25. https://doi.org/10.22456/2175-2745.88822.

Murata, T. (1989). Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* **77**(4): 541–580. https://doi.org/10.1109/5.24143.

Nguyen, T. A., Min, D., Choi, E. and Tran, T. D. (2019). Reliability and availability evaluation for cloud data center networks using hierarchical models, *IEEE Access* **7**: 9273–9313. https://doi.org/10.1109/ACCESS.2019.2891282.

Nogueira, B. (2015). *Exploração multiobjetivo do espaço de projeto de sistemas embarcados de tempo-real não críticos*, PhD thesis, Universidade Federal de Pernambuco. Centro de Informática. Disponível em https://repositorio.ufpe.br/handle/123456789/28832.

Oliveira, D., Matos, R., Dantas, J., Ferreira, J. a., Silva, B., Callou, G., Maciel, P. and Brinkmann, A. (2017). Advanced stochastic petri net modeling with the mercury scripting language, *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2017, ACM, New York, NY, USA, pp. 192–197. https://doi.org/10.1145/3150928.3150959.
URL: *http://doi.acm.org/10.1145/3150928.3150959*

Oliveira, H. (2005). Algoritmo evolutivo no tratamento do problema de roteamento de veículos com janela de tempo. Monografia (Graduação em Bacharelado em Ciências da Computação), Departamento de Ciências da Computação, Universidade Federal de Lavras. Disponível em http://repositorio.ufla.br/jspui/handle/1/9293.

Pappa, G. (2002). *Seleção de atributos utilizando algoritmos genéticos multiobjetivos*, Master's thesis, Programa de Pós Graduação em Informática Aplicada da Pontifícia. Departamento de Estatística e Informática.

Rampazzo, P. C. B., Yamakami, A. and de França, F. O. (2013). Algoritmo genético e evolução diferencial para a resolução do problema de planejamento hidrelétrico, *in* H. S. Lopes, L. C. de Abreu Rodrigues and M. T. A. Steiner (eds), *Meta-Heurísticas em Pesquisa Operacional*, 1 edn, Omnipax, Curitiba, PR, chapter 19, pp. 307–324. http://dx.doi.org/10.7436/2013.mhpo.19.

Rao, S. (2009). *Engineering Optimization: Theory and Practice: Fourth Edition*, John Wiley and Sons. https://doi.org/10.1002/9780470549124.

Ratzer, A. V., Wells, L., Lassen, H. M., Laursen, M., Qvortrup, J. F., Stissing, M. S., Westergaard, M., Christensen, S. and Jensen, K. (2003). Cpn tools for editing, simulating, and analysing coloured petri nets, *in* W. M. P. van der Aalst and E. Best (eds), *Applications and Theory of Petri Nets 2003*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 450–462. https://doi.org/10.1007/3-540-44919-1_28.

Than, M. M. and Thein, T. (2020). Energy-saving resource allocation in cloud data centers, *2020 IEEE Conference on Computer Applications(ICCA)*, pp. 1–6. https://doi.org/10.1109/ICCA49400.2020.9022819.

Tripathi, R., Vignesh, S. and Tamarapalli, V. (2017). Optimizing green energy, cost, and availability in distributed data centers, *IEEE Communications Letters* **21**(3): 500–503. https://doi.org/10.1109/LCOMM.2016.2631466.

Trivedi, K., Hunter, S., Garg, S. and Fricks, R. (1996). Reliability analysis techniques explored through a communication network example. Disponível em http://www.lib.ncsu.edu/resolver/1840.4/971.

W. Zucchi, A. A. (2013). Construindo um data center, *Revista USP* **97**: 43–58. https://doi.org/10.11606/issn.2316-9036.v0i97p43-58.