Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Matheus Correia Bentes

Implementação de Plataforma Evolucionária Extrínseca de Circuitos Eletrônicos

Rio de Janeiro

2020

Matheus Correia Bentes

Implementação de Plataforma Evolucionária Extrínseca de Circuitos Eletrônicos

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientador: Prof. Dr. José Franco Machado do Amaral

Rio de Janeiro

2020

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

B475 Bentes, Matheus Correia.
Implantação de plataforma evolucionária extrínseca de circuitos eletrônicos / Matheus Correia Bentes. – 2020.
140f.

Orientador: José Franco Machado do Amaral.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Circuitos eletrônicos - Teses. 3. Algorítmos genéticos - Teses. 4. Sistemas difusos - Teses. 5. Otimização matemática - Teses. I. Amaral. José Franco Machado do. II. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. III. Título.

CDU 007.52

Bibliotecária: Júlia Vieira – CRB7/6022

Assinatura _____ Data _____

Matheus Correia Bentes

Implementação de Plataforma Evolucionária Extrínseca de Circuitos Eletrônicos

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovada em 18 de fevereiro de 2020.

Banca Examinadora:

_____

Prof.º Dr. José Franco Machado do Amaral

Faculdade de Engenharia – UERJ

_____

Prof.º Dr. Pedro Henrique Gouvêa Coelho

Faculdade de Engenharia – UERJ

_____

Prof.º Dr. Ricardo Tanscheit

Departamento de Engenharia Elétrica – PUC-Rio

Rio de Janeiro

2020

DEDICATÓRIA

Dedico este trabalho à minha família, pelo suporte e motivação.

AGRADECIMENTOS

Agradeço à minha família, pelo estímulo ao longo de minha formação.

Ao meu orientador, Prof.º José Franco Machado do Amaral, por ter aceitado me orientar e por ter me auxiliado no decorrer do trabalho realizado.

Aos professores do Programa de Pós-Graduação em Engenharia Eletrônica, pelo ensinamento repassado ao longo do curso.

Ao Programa de Pós-Graduação em Engenharia Eletrônica, pela oportunidade de realizar o curso de mestrado e aprimorar minha formação.

À UERJ, por me acolher desde minha graduação e por proporcionar essa e outras oportunidades de aprimoramento profissional e pessoal.

Aos professores da banca pela disponibilidade de avaliar o trabalho realizado.

A todos aqueles que contribuíram direta e indiretamente para que a realização desse trabalho fosse possível.

RESUMO

BENTES, Matheus Correia. *Implementação de plataforma evolucionária extrínseca de circuitos eletrônicos*. 2020. 140f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

Este trabalho apresenta uma plataforma de evolução de circuitos eletrônicos baseada em algoritmos genéticos com diferentes modos de funcionamento. A plataforma possui uma estrutura extrínseca de avaliação de indivíduos, realizando chamadas a um simulador de circuitos para cada possível solução avaliada. A plataforma pode executar evoluções em busca de valores para componentes, topologias adicionais a um circuito fixo e uma busca com variação total dos tipos de componentes, valores e conexões. A aptidão avaliada pode ser com base em um único objetivo, avaliando apenas a saída do circuito, como também baseada em vários objetivos. O método escolhido para essa quantificação de múltiplos objetivos é baseado num Sistema Fuzzy com o objetivo de facilitar a especificação do projetista. As evoluções podem ser realizadas no domínio do tempo bem como no domínio da frequência, sendo possível ao usuário alterar o modo de funcionamento sem alterações no código já criado. A troca entre os modos de operação, entradas utilizadas e a utilização de funções presentes na plataforma é realizada diretamente através de variáveis de configuração, sem necessidade de alteração no código fonte da plataforma. Com o objetivo de verificar o desempenho da plataforma, cada modo foi avaliado utilizando diferentes circuitos com complexidades variadas. No decorrer do trabalho é apresentada uma conclusão do impacto de cada variável e do modo utilizados no processo de evolução.

Palavras-chave: Eletrônica Evolucionária; Circuitos Eletrônicos; Algoritmos Genéticos; Sistemas Fuzzy; Otimização multiobjetivo.

ABSTRACT

BENTES, Matheus Correia. Implementation of an extrinsic evolutionary platform for electronic circuits. 2020. 140f. Dissertation (Master in Electronic Engineering) - Faculty of Engineering, State University of Rio de Janeiro, Rio de Janeiro, 2020.

This work presents an electronic circuit evolution platform based on genetic algorithms with different modes of operation. The platform has an extrinsic structure for evaluating individuals, making calls to a circuit simulator for each possible evaluated solution. The platform can perform evolutions in search of values for components, additional topologies to a fixed circuit and a search with total variation in the types of components, values and connections. The suitability assessment can be based on a single objective, evaluating only the output of the circuit, but also on several objectives. The method chosen for this quantification of multiple objectives is based on a Fuzzy System in order to facilitate the designer's specification. The evolutions can be carried out in the time domain as well as in the frequency domain, being possible for the user to change the operating mode without changes in the code already created. The switching between operating modes, inputs used and the use of functions present on the platform is carried out directly through configuration variables, without the need to change the platform source code. In order to verify the performance of the platform, each mode was evaluated using different circuits with varying complexities. In the course of the work a conclusion is presented of the impact of each variable and the way used in the evolution process.

Keywords: Evolutionary Electronics; Electronic circuits; Genetic Algorithms; Fuzzy systems; Multiobjective optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Motivação

Com o avanço da tecnologia na área da engenharia eletrônica, os dispositivos tiveram sua capacidade de processamento ampliada e o espaço necessário para implementação foi reduzido. Isto viabilizou que os equipamentos se tornassem cada vez menores e com maior desempenho.

Devido a essa capacidade de miniaturização e alto desempenho, dispositivos eletrônicos têm sido empregados de forma dedicada nas mais diversas aplicações. Tais dispositivos, conhecidos como Sistemas Embarcados, permitem agregar em um único equipamento todo o processamento necessário à execução de uma determinada ação.

As ferramentas adicionais que um dispositivo pode conter variam desde sistemas básicos de diagnósticos a ferramentas adicionais às funções desempenhadas pelo dispositivo. Seja qual for o acréscimo ao sistema básico, todos têm como objetivo melhorar a experiência do usuário ao demandar o mínimo de interferência do mesmo e integrar-se a outros sistemas utilizados.

A separação de sistemas em partes dedicadas a uma tarefa específica (módulos) permite identificar mais facilmente divisões defeituosas e realizar a substituição mais rapidamente. Tal separação também possibilita a atualização do sistema, bastando substituir um módulo por outro com desempenho superior.

Entretanto, apesar de facilitar o reparo e atualização de sistemas, esta segmentação não busca solucionar eventuais falhas internas. Tal abordagem considera a utilização de dispositivos até o aparecimento de falhas e, consequentemente, descarte do módulo defeituoso e substituição por outro com comportamento adequado. Dessa forma, esta metodologia afeta diretamente o impacto tecnológico negativo no meio ambiente, contribuindo para o aumento de lixo eletrônico.

Além da preocupação com o impacto ambiental do descarte acelerado de dispositivos, há aplicações que não permitem uma fácil substituição dos módulos. Aplicações nas áreas de exploração espacial e exploração petrolífera demandam soluções perante falhas funcionais de modo rápido e automático, devido ao custo e complexidade associados a qualquer reparo exigido. Para tais aplicações, busca-se uma ferramenta que, associada ao funcionamento dos dispositivos, seja capaz de contornar ou mitigar os problemas causados por qualquer parte defeituosa. Essa busca compreende uma linha especifica de estudo.

Com o objetivo de possibilitar adaptabilidade aos dispositivos, pode-se utilizar uma linha de pesquisa denominada Eletrônica Evolucionária (HADDOW; TYRRELL; 2018). Esta linha de pesquisa é baseada na técnica de Computação Evolucionária, que se inspira na Evolução Darwiniana para a resolução de problemas (COELLO; 1999) (COELLO; 2013) (LABATI et. Al.; 2016). A aplicação desta pesquisa, empregada no autorreparo, busca evoluir o funcionamento de circuitos eletrônicos de forma autônoma (SINOHARA; 2001) (SANTOS; SILVA; SUETAKE; 2012). Essa evolução de circuitos de forma automática corresponde ao comportamento objetivo do trabalho realizado.

Objetivo do Trabalho

A proposta desta pesquisa consiste em desenvolver um sistema que, utilizando eletrônica evolucionária, seja capaz de evoluir topologias de circuitos de acordo com a especificação do usuário.

O sistema criado compõe uma plataforma de evolução que tem como objetivo fornecer uma avaliação rápida para quaisquer tipos de circuito avaliados. Tal avaliação será feita diretamente a partir de informações fornecidas pelo usuário. O conceito de plataforma é dado ao trabalho proposto, pois fornece funções que auxiliam a análise de projetistas durante a concepção de circuitos analógicos.

A plataforma de evolução proposta deverá ser capaz de utilizar uma representação de circuito (Informações de conexões, valores e tipos de componentes) e, a partir dela, realizar a criação de arquivos compatíveis com o simulador utilizado independentemente do tipo de circuito evoluído.

O código base da plataforma já engloba componentes básicos como resistores, capacitores, fontes DC, transistores e modelos mais complexos como amplificadores operacionais. Entretanto, a plataforma proposta deverá ser capaz de absorver outros modelos e tipos de componentes apenas adicionando tais atualizações à lista de modelos utilizados e determinando uma representação que as identifique. Após esta incorporação tais componentes ficam disponíveis para utilização de acordo com o desejo do usuário.

Os tipos de componentes disponíveis em uma evolução, quantidade de componentes utilizadas, quantidade de nós disponíveis e outras variáveis de evolução deverão ser completamente customizáveis pelo usuário. Do mesmo modo que tais variáveis, o modo de

evolução é facilmente alterado apenas por seleção do usuário, sem necessidade de alterações no código de avaliação.

A plataforma possuirá três modos de funcionamento, que serão explicados mais detalhadamente no decorrer deste trabalho. Tais modos permitirão a evolução desde valores de componentes para uma topologia fixa à evolução completa de uma topologia. Cada modo de funcionamento utilizará informações específicas para realizar evoluções customizáveis pelo usuário, informações essas também apresentadas no decorrer deste trabalho.

O primeiro modo de evolução deverá ser capaz de, a partir de uma topologia informada pelo usuário, evoluir valores para tais componentes de modo a aproximar ao máximo a saída observada da saída desejada.

O segundo modo de evolução deve ser capaz de, a partir de uma topologia disponível, evoluir uma topologia adicional ao circuito que o aproxime da saída desejada. Tal modo de evolução pode ser compreendido como adaptação de circuitos reconfiguráveis, através da adição de componentes, de modo a corrigir uma falha da resposta observada. O processo de detecção de tais falhas não faz parte do escopo deste projeto, mas podem ser citados os trabalhos realizados em (AMARAL; 2006) e (AMARAL; AMARAL; TANSCHEIT; 2006). O escopo do trabalho terá como foco a adaptabilidade para correção de falhas.

O terceiro e último modo corresponde ao caráter exploratório da plataforma. Neste modo, o algoritmo deverá ser capaz de alterar todas as variáveis de formação de um circuito em busca de topologias não diretas para um problema. Através desse modo de evolução, o algoritmo deve ser capaz de alterar desde o valor e tipo de um componente até suas conexões com os demais componentes utilizados.

Além da avaliação focada na compatibilidade da saída observada com a saída desejada, o sistema deverá ser capaz de realizar uma avaliação multiobjetivo durante a evolução. Nesse modo o algoritmo, de acordo com critérios pré-estabelecidos, buscará evoluir uma solução que melhor atenda ao problema.

Tanto na avaliação de apenas um objetivo quanto na avaliação multiobjectivo, a plataforma busca apenas fornecer uma ferramenta de apoio que facilite o trabalho do projetista. Essa ferramenta deverá funcionar corretamente para cada um dos três modos de evolução, fornecendo sugestões durante o processo de concepção de circuitos eletrônicos analógicos.

O trabalho demonstrará as funções que compõe a plataforma, apresentando estudos de casos com valores de parâmetros fixos. Uma análise simplificada do impacto de cada parâmetro é apresentada, entretanto, tal avaliação não é o objetivo principal do projeto. O propósito é fornecer uma plataforma que possibilite tal pesquisa de modo simplificado.

Organização da Dissertação

Esta dissertação está dividida em cinco capítulos adicionais à Introdução.

O capítulo 2 trata de forma resumida os conceitos importantes do embasamento teórico da plataforma desenvolvida. Este capítulo também aborda a base de trabalhos utilizada na criação do algoritmo final.

O capítulo 3 foca na estrutura do algoritmo final desenvolvido. Neste capítulo são apresentados as funcionalidades e os métodos utilizados no decorrer do trabalho realizado. Tal detalhamento vai desde o formato dos dados de entrada aos formatos de arquivos gerados para o usuário ao final da evolução.

O capítulo 4 apresenta o embasamento necessário para a implementação da lógica multiobjetivo na plataforma desenvolvida. Além da abordagem teórica necessária, é apresentada a metodologia adotada para agregar tal lógica ao algoritmo desenvolvido de modo a facilitar a customização pelo usuário.

O capítulo 5 trata dos estudos de caso para os diferentes modos de funcionamento da plataforma. Este capítulo visa apresentar os resultados e desempenhos observados durante cada evolução para uma posterior conclusão no capítulo seguinte.

No capítulo 6 tem-se uma conclusão sobre a eficiência da plataforma ao evoluir as diferentes topologias de circuitos. Neste capítulo também é apresentada uma discussão sobre o impacto das variáveis no processo evolutivo e, consequentemente, em seu desempenho, além de discutir situações pertinentes observadas durante a elaboração deste trabalho.

1 ALGORITMOS GENÉTICOS E ELETRÔNICA EVOLUCIONÁRIA

A Teoria da Evolução das espécies, proposta pelo cientista britânico Charles Darwin, revolucionou a área da biologia e, posteriormente, a ciência como um todo. Através da observação de determinadas espécies de animais, Darwin constatou que indivíduos de uma determinada população repassam suas características biológicas aos seus descendentes. Tais características tornam um indivíduo mais ou menos apto ao ambiente em que está inserido. A partir da aptidão de um indivíduo às características do ambiente, um indivíduo tende a repassar mais ou menos seus atributos e, consequentemente, tais aspectos tendem a ser mais comuns na população ou tendem a se tornar mais atípicos.

Inspirado no comportamento observado por Darwin, uma linha de pesquisa surgiu com o objetivo de otimizar a busca de soluções no âmbito computacional.

1.1 Algoritmos Genéticos e Suas Aplicações

Uma linha de pesquisa inspirada na Teoria da Evolução de Darwin recebeu o nome de Computação Evolucionária (HOLLAND; 1975). Tal linha de pesquisa é um ramo da inteligência computacional e faz uso de algoritmos de busca na resolução de problemas cujo cálculo direto demandaria elevado tempo e processamento matemático (EREMIA; LIU; EDRIS; 2016). Um exemplo deste tipo de algoritmo são os chamados Algoritmos Genéticos (LACERDA; CARVALHO; 1999).

Os Algoritmos Genéticos foram desenvolvidos com inspiração na eficiência da Evolução Natural. Através de seus processos modificadores, a Evolução Natural é capaz de gerar novas soluções para problemas existentes, como, por exemplo, doenças. Algoritmos Genéticos buscam utilizar tal metodologia de processos modificadores para evoluir soluções mais rapidamente para problemas multidimensionais ou que demandassem elevado tempo de processamento (GOLDBERG; 1989).

A metodologia de funcionamento dos Algoritmos Genéticos utiliza processos modificadores baseados na lógica de Descendência com Modificação presente na Evolução Natural. Tais processos consistem em conceitos biológicos que regem todas as formas de vida.

Esses conceitos e seus equivalentes utilizados nos Algoritmos Genéticos podem ser observados na Tabela 1.

Tabela 1 - Relação Entre Conceitos da Evolução Natural e dos Algoritmos Genéticos

Conceito da Natureza	Componente dos Algoritmos Genéticos
Cromossomo	Representação Binária
Indivíduo	Possível Solução
População	Conjunto de Soluções Disponíveis
Geração	Ciclo de Busca
Meio Ambiente	Algoritmo Genético
Reprodução Sexual	Operador Combinação (*Crossover*)
Mutação Genética	Operador Mutação (*Mutation*)
Seleção Natural	Operador Seleção (*Selection*)
Doença	Problema a Ser Solucionado
Imunidade	Aptidão

Fonte: O autor, 2019.

A compreensão da equivalência entre esses conceitos é fundamental para o entendimento dos Algoritmos genéticos e de sua utilização na evolução de soluções. Tal percepção também auxilia na correta configuração dos parâmetros utilizados durante a busca por soluções.

O primeiro conceito corresponde à representação de um indivíduo. Essa representação permite a identificação do mesmo e, através destas características, permite a futura avaliação do mesmo para a circunstância observada. Na natureza essa representação é cromossômica e na resolução matemática de problemas pode corresponder a uma representação binária.

O segundo conceito é entender que cada indivíduo corresponde a uma solução potencial para qualquer aspecto analisado. Uma população de indivíduos corresponde a um conjunto de possíveis soluções distintas disponíveis em um determinado período de tempo, tempo este medido em gerações na Evolução Natural e em ciclos de busca na Evolução Computacional. Na evolução computacional, a população inicial é criada de forma aleatória, ou seja, cada indivíduo é criado com valores de representação aleatórios para as variáveis observadas.

O meio ambiente, assim como o Algoritmo Genético, tem o papel de manter o processo evolucionário de acordo com os parâmetros que o determinam. O papel tanto do meio ambiente quando do Algoritmo Genético nessa abordagem seria o de determinar os parâmetros aos quais

os indivíduos reagirão e conservar tais indivíduos durante a execução do processo. Há, entretanto, uma abordagem em que o meio ambiente executa o papel de avaliar a aptidão de cada indivíduo. Nela a relação do meio ambiente e do Algoritmo genético seria com o processo de determinação da aptidão de cada indivíduo.

Cada um os operadores presentes nos Algoritmos Genéticos correspondem a um processo modificador presente na Evolução Natural. O primeiro processo modificador corresponde ao operador responsável pela seleção dos indivíduos. Tal fator representa a taxa com que os indivíduos são selecionados e está relacionada com sua aptidão. Esse fator e o método correspondente são os responsáveis por selecionar indivíduos em forma de sorteio, associando a probabilidade à aptidão do indivíduo. Deste modo, os indivíduos mais aptos têm mais chances de sobreviver e deixar descendentes.

O operador que realiza a combinação de indivíduos (*crossover*) corresponde ao processo de reprodução sexual entre os indivíduos selecionados anteriormente. Tal fator determina o percentual de indivíduos que combinam suas representações gerando indivíduos com representações mistas. Tais indivíduos correspondem ao nascimento de descendentes na Evolução Natural ou a obtenção de soluções com valores de variáveis combinadas em um problema computacional.

Associado ao fator de combinação de indivíduos, existe o processo de modificação durante o processo de combinação. Esse processo de modificação é realizado pelo Operador Mutação (*Mutation*). Esse processo modificador realiza uma variação aleatória no valor das combinações das representações durante a concepção de um novo indivíduo. Tal ação corresponde a uma modificação genética aleatória na Evolução Natural e a atribuição de um valor aleatório para uma variável na Evolução Computacional.

Através desses processos modificadores, indivíduos são selecionados pela sua aptidão (Evolução Computacional) ou sua resposta a doença (Evolução Natural) até que um indivíduo alcance uma solução ideal (Conceito Computacional) ou alcance imunidade (Conceito Natural) ao problema avaliado.

Após a constatação da eficiência desse método de evolução, tal processo evolucionário começou a ser empregado na evolução não somente de soluções matemáticas. A partir daí surge uma outra linha de pesquisa voltada à Engenharia Eletrônica.

1.2 Eletrônica Evolucionária e suas aplicações

Com a aplicação da Evolução Computacional na área de Engenharia Eletrônica uma nova linha de pesquisa surgiu. Essa linha de pesquisa é denominada Eletrônica Evolucionária. Nessa linha de pesquisa a evolução é realizada através da avaliação de circuitos eletrônicos e o objetivo é evoluir tais circuitos até a obtenção de especificação desejada. (GREENWOOD; TYRREL; 2007) (REORDA; STERPONE; ULLAH; 2017)

Para realizar a evolução desses circuitos, são utilizados os mesmos operadores e a lógica de funcionamento utilizada na Computação Evolucionária. Entretanto, tais características devem ser adaptadas para este tipo de evolução.

Ao se utilizar Algoritmos Genéticos na evolução de circuitos, os indivíduos avaliados são os circuitos ao invés de soluções numéricas como na Computação Evolucionária. A representação de cada indivíduo é adaptada para representar uma característica especifica de circuitos, característica essa que será alvo do processo de evolução.

Os conceitos de população e geração são transportados de uma linha de pesquisa para a outra sem alteração de significado. Para ambas as linhas de pesquisa, a população de indivíduos representa um conjunto de possibilidades (Representações) para a característica alvo da evolução. Na evolução de circuitos evoluem-se topologias de circuitos. Essa característica pode variar de acordo com o projetista, podendo, por exemplo, representar o valor de componentes ou sua tolerância.

Uma vez determinadas as características de um conjunto de soluções, tais indivíduos podem ser avaliados e selecionados pelo processo responsável por avaliar a aptidão de cada possível solução (*Selection*). Essa seleção e modificação da geração seguinte são então realizadas de forma iterativa até o critério de parada estabelecido anteriormente para o circuito evoluído. Esse critério pode ser avaliar a resposta observada pelo circuito ou o tempo necessário para o processo, ou seja, uma quantidade máxima de ciclos (Gerações) disponíveis para o processo de busca.

Os operadores responsáveis pela modificação da população (conjunto de soluções) apresentarão o mesmo comportamento apresentado anteriormente, comportamento esse responsável pela eficiência do método. O operador de combinação (*Crossover*) será responsável por mesclar as representações em busca de uma solução com melhor desempenho. A taxa da população que realizará combinação afeta diretamente a velocidade da evolução, pois determina a velocidade de variação observada a cada ciclo de busca.

O processo de mutação (*Mutation*) realizará sobre os indivíduos combinados atribuindo um caráter de aleatoriedade sobre a característica evoluída. Isso permite à evolução a descoberta de características promissoras sem a restrição as pertencentes aos indivíduos geradores.

Dessa forma, a compreensão da evolução de uma topologia através de Algoritmos Genéticos é intercambiável com o modo que uma população se torna imune a uma doença através da Evolução Natural. Tal comportamento funcional desses casos pode ser observado e comparado na Figura 1.

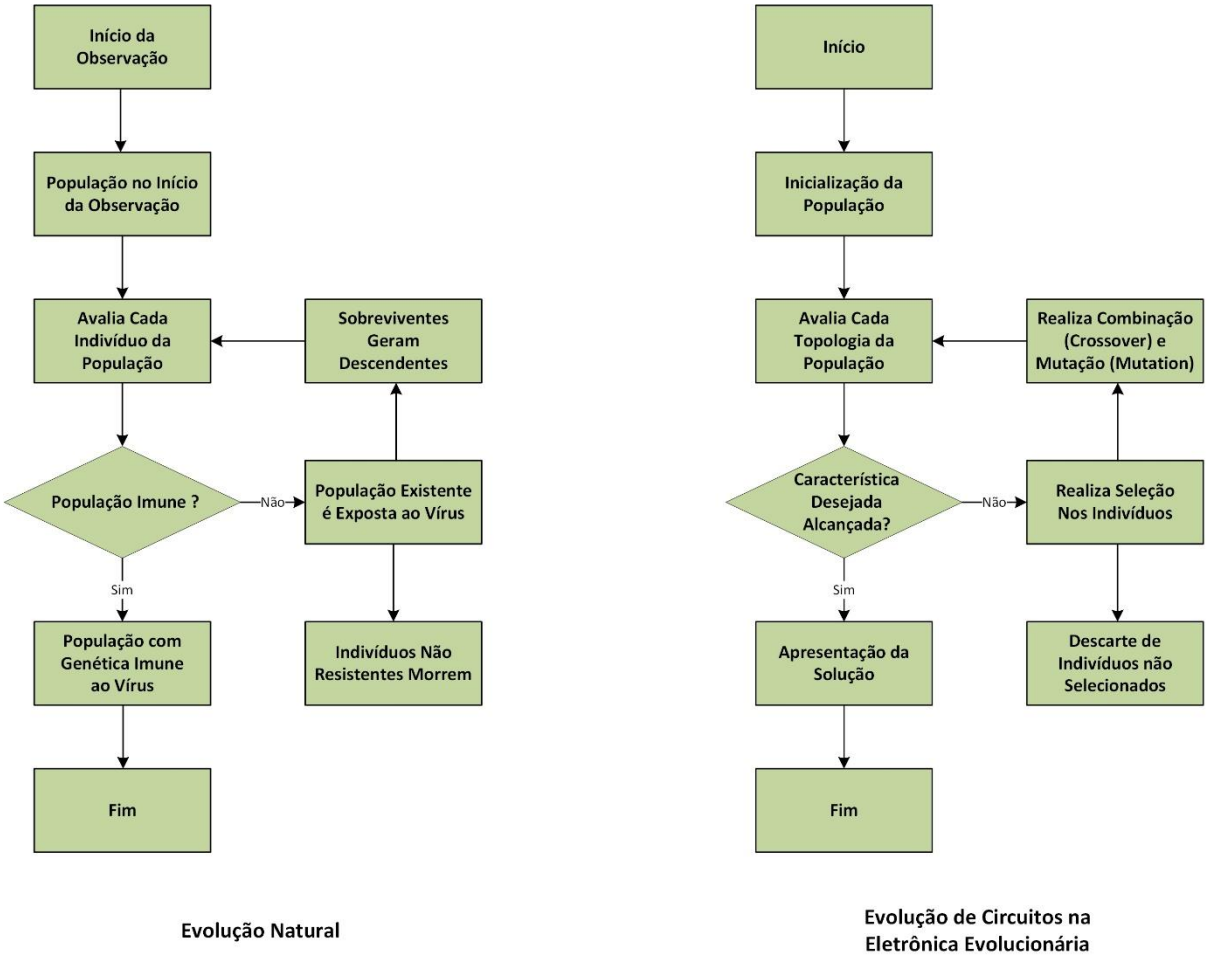

Figura 1 - Comparação Entre Evolução Natural e Eletrônica Evolucionária
Fonte: O autor, 2019.

Uma vez definido o modo de operação para a evolução de circuitos, o Algoritmo Genético pode ser inserido em plataformas dedicadas à evolução de circuitos. Tais plataformas são denominadas de Plataformas Evolucionárias e são classificadas de acordo com o seu modo de funcionamento.

1.3 Tipos de plataformas Com Eletrônica Evolucionária

As Plataformas Evolucionárias podem ser utilizadas para atuar em circuitos e adaptá-los no caso de presença de falhas (AMARAL; 2003). Nesse tipo de aplicação, plataformas reconfiguráveis são capazes de reestruturar suas conexões e dispositivos utilizados para alcançar uma resposta desejada, resposta essa que pode ser analógica ou digital.

Essas plataformas são classificadas de acordo com o tipo de projeto realizado, a natureza do projeto evoluído e a estrutura de funcionamento da plataforma. O tipo de projeto realizado é classificado de acordo com o objetivo da evolução realizada. Este objetivo pode ser a otimização de um circuito já existente ou a síntese de uma topologia de circuito que satisfaça uma determinada saída desejada.

A classificação da natureza do projeto refere-se à natureza da topologia evoluída. Essa natureza corresponde ao tipo de grandeza observada na saída da topologia e avalia se o circuito possui uma saída analógica ou digital.

Em relação à estrutura de funcionamento, a classificação observa o modo de avaliação realizada pela plataforma. A avaliação de circuitos pode ser realizada de forma interna ou externa à plataforma de aplicação. Quando o processo de avaliação é realizado de forma interna, os indivíduos são carregados diretamente na plataforma e a aptidão de cada representação é dada observando-se a resposta obtida na saída da plataforma. Esse tipo de plataforma recebe a classificação de Plataforma Intrínseca. Tal técnica garante um valor de aptidão perfeitamente compatível com o desempenho final do circuito, uma vez que tal aptidão é avaliada diretamente na plataforma de hardware em que a topologia será aplicada.

Por outro lado, a avaliação de indivíduos pode ser realizada de forma externa à plataforma em que a topologia será aplicada. Esse modo de avaliação utiliza simuladores de circuitos para a avaliação da resposta de cada topologia e atribui o valor de aptidão de acordo com o desempenho simulado de cada uma das representações utilizadas. Esse tipo de avaliação não possui uma avaliação de aptidão tão fidedigna como a avaliação de Plataformas Intrínsecas. Entretanto, a utilização de simuladores na avaliação garante uma avaliação mais flexível de cada topologia evoluída, sendo mais apropriada à experimentação. Às plataformas com avaliação por meio de simuladores é dado o nome de Plataformas Extrínsecas, ou seja, de modo externo à plataforma onde será utilizada.

Dessa forma, o modo de operação de plataformas Intrínsecas e Extrínsecas difere apenas no modo de avaliação dos indivíduos da evolução. Trabalhos cujo foco é o conhecimento exato

da resposta após implementadas, ao invés de uma aproximação simulada, utilizam normalmente Plataformas Intrínsecas (SANTINI; 2001). Nesse tipo de trabalho, o sistema deve possuir uma interface permanente com um hardware microcontrolado para a troca de informações (FONSECA; 2017).

Por outro lado, projetos nos quais o foco é otimizar a experimentação, ou quando não é possível interromper o funcionamento da plataforma, usualmente utilizam a simulação dos circuitos. Essa simulação pode ser realizada diretamente pelo ambiente onde a programação é realizada (BACELAR; 2017) ou ser realizada por um simulador específico para posterior implementação do circuito final evoluído (ZEBULUM; 1999).

2 CONCEPÇÃO DO AMBIENTE EVOLUTIVO

A escolha do ambiente utilizado neste trabalho teve como foco o desenvolvimento de uma plataforma que fosse capaz de evoluir topologias que utilizassem valores comerciais de componentes (CARVALHO; 2018). Dessa forma, o processo de avaliação pode ser realizado de maneira ágil e a resposta observada pelas topologias são muito próximas da resposta real observada.

A plataforma desenvolvida neste trabalho consiste em uma plataforma extrínseca cuja simulação de circuitos será realizada externamente por um simulador específico. O simulador escolhido foi o LTSpice XVII, desenvolvido pela fabricante de circuitos integrados Linear Technology Corporation, que agora faz parte da multinacional Analog Devices. Essa escolha é justificada por ele ser um simulador globalmente utilizado e distribuído, disponível tanto para o sistema operacional Windows como para o MAC, e por ser um software de distribuição gratuita, não necessitando de licença para seu funcionamento.

A plataforma realizará a evolução de circuitos analógicos, ou seja, será uma plataforma de natureza analógica. Além da classificação quanto à natureza da resposta observada, há a classificação quanto à natureza do processo realizado.

Como será apresentado, a plataforma possuirá três modos de operação. No modo em que a plataforma evolui os valores de componentes, ela será classificada como uma plataforma de síntese de topologia. A classificação de síntese de topologia também é dada ao modo em que é realizada uma alteração completa de componentes e conexões com o objetivo de se obter uma topologia satisfatória.

No modo adição de componentes há uma topologia inicialmente fixada que deve ter seu comportamento aprimorado através da adição de componentes. A esse modo que busca corrigir o funcionamento de uma topologia é atribuída a classificação plataforma de natureza de otimização.

A plataforma utilizada para a programação e, consequentemente, responsável por realizar chamadas ao simulador e operar o algoritmo será o MATLAB (*MATrix LABoratory*). O MATLAB corresponde a um software desenvolvido pela MathWorks e possui um conjunto de ferramentas (*toolbox*) que auxiliam na programação e nas associações necessárias ao utilizar um Algoritmo Genético.

O Algoritmo Genético utilizado como base para o trabalho é o GAOT (*Genetic Algorithm Optimization Toolbox*). Esse algoritmo foi desenvolvido por pesquisadores da *North*

Caroline State University para ser utilizado diretamente no ambiente de desenvolvimento do MATLAB na maximização de uma função. Tal ferramenta possui código aberto, ou seja, seus arquivos podem ser utilizados e modificados, possibilitando que seja utilizado em diversas aplicações. Neste trabalho o algoritmo foi adaptado para a evolução de topologias de circuitos eletrônicos como realizado em (RIBEIRO; 2012).

A comunicação entre o GAOT, contido no MATLAB, e o simulador de circuitos é realizada através de arquivos *netlist*, ou seja, arquivos de descrição de conexões de circuitos. Dessa forma, o MATLAB criará os arquivos com as topologias a serem testadas e realizará a leitura dos arquivos com a amostragem resultante do processo de simulação. Essa conexão é mais detalhada adiante.

O MATLAB será utilizado para ajustar os parâmetros de evolução (utilizados) e realizar a troca de informações entre o Algoritmo Genético utilizado (GAOT) e o simulador de circuitos utilizados (LTSpice). O detalhamento do valor desses parâmetros será realizado em conjunto com os estudos de caso para facilitar o entendimento. Uma estrutura resumida da plataforma desenvolvida pode ser observada a seguir na Figura 2.

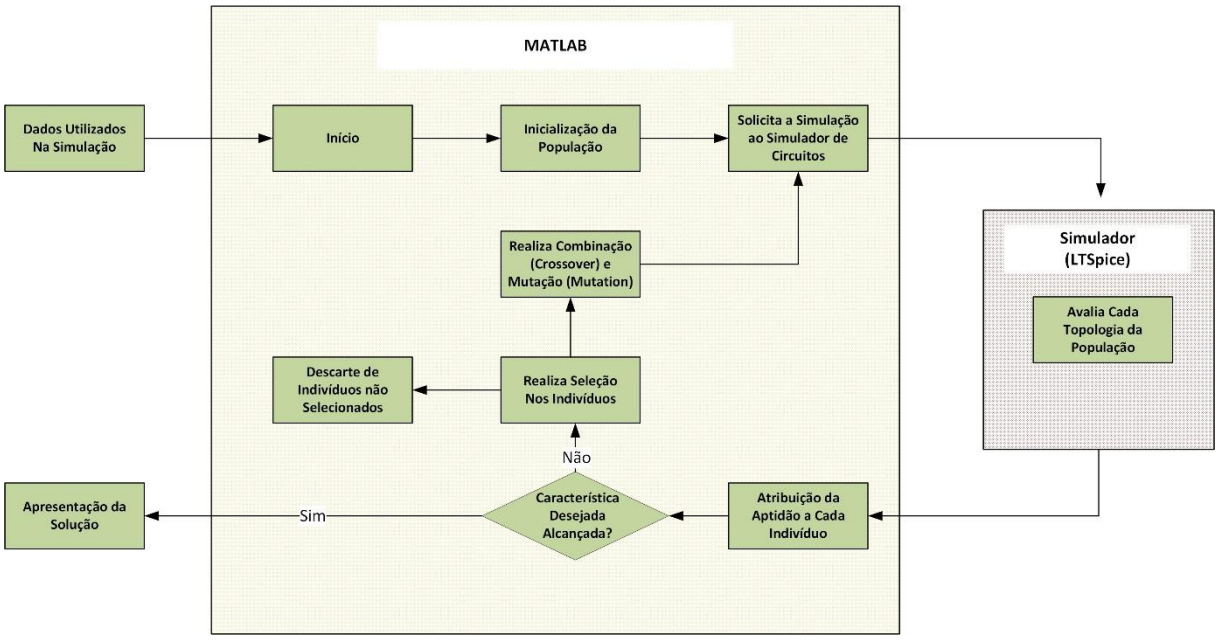

Figura 2 - Fluxograma de Funcionamento da Plataforma Desenvolvida

Fonte: O autor, 2019.

O algoritmo de evolução e o simulador de circuitos foram definidos após a análise dos trabalhos existentes. Dessa forma, o passo seguinte no desenvolvimento do trabalho foi determinar cada característica de funcionamento da plataforma.

Cada característica foi pensada com o objetivo de, obedecendo às restrições do ambiente evolutivo utilizado, adaptar o projeto de modo a alcançar todos os objetivos idealizados. A primeira destas características é o modo que a aptidão de cada elemento é calculada.

2.1 Cálculo da Avaliação de Aptidão

Inicialmente, o cálculo de aptidão foi considerado como a relação direta ao erro. Neste modo de avaliação, a aptidão de cada indivíduo seria determinada pela diferença entre a resposta observada e a resposta desejada. O melhor indivíduo seria o que obtivesse a menor diferença entre a resposta alcançada e a resposta almejada pelo circuito.

Entretanto, como visto na apresentação do algoritmo escolhido, o GAOT possui a capacidade de maximizar a resposta para uma determinada função. Dessa forma, a avaliação inicialmente cogitada para a aptidão não pode ser utilizada pois o erro deveria ser minimizado e não maximizado, como é realizado pelo algoritmo escolhido.

Sabendo o padrão de funcionamento do algoritmo escolhido, foi necessário conceber uma nova função de aptidão para o sistema. Nessa nova fórmula de avaliação, a aptidão deverá aumentar conforme a resposta dada por um indivíduo se aproxime da resposta desejada. Dessa forma, a função escolhida para avaliar os indivíduos da evolução é dada por:

$$Aptidão = \frac{1}{1 + Erro}$$
$$Erro = \sum_{i=1}^{N} \left| \frac{V_{desejada}(i) - V_{observada}(i)}{N} \right| \quad (1)$$

Na fórmula escolhida, o erro entre a saída desejada ($V_{desejada}$) e a saída obtida pelo circuito evoluído ($V_{observada}$) é considerado em módulo e em média. Tal avaliação foi utilizada para garantir que o valor individual de cada amostra seja considerado no valor final, impedindo que valores de erros positivos sejam compensados por valores de erros negativos.

Uma vez que o erro é considerado em módulo, o valor da aptidão será obrigatoriamente um valor entre zero e um. Esse intervalo restrito da aptidão permite a utilização tanto do método de seleção roleta quanto do método de seleção normalizado disponíveis pelo algoritmo de evolução utilizado como base. A escolha do modo de seleção utilizado e a possibilidade de

entendimento desses valores entre zero e um como um valor percentual de aptidão serão o foco de capítulos futuros deste trabalho.

2.2 Conexão entre algoritmo e simulador escolhidos

Definida a equação para a avaliação de aptidão, o próximo passo para o desenvolvimento do sistema é definir como será a comunicação com o simulador de circuitos. O simulador a ser utilizado e o seu modo de operação foram definidos anteriormente.

O objetivo é determinar como será realizada a comunicação entre o MATLAB e simulador de modo a obedecer ao modo de operação de ambos. Para realizar esta conexão do simulador escolhido (LTSpice) como o MATLAB, são realizadas chamadas ao simulador em segundo plano.

As chamadas ao simulador são realizadas em segundo plano para cada circuito simulado. A realização de chamadas em segundo plano desobriga o usuário de intervenções sucessivas no decorrer da evolução. Esse tipo de chamada ao simulador também permite ao usuário focar e acompanhar o andamento da evolução, observação parcial esta que será foco de discussões futuras do trabalho.

O simulador utilizado, apesar de possuir uma interface gráfica, utilizará arquivos de texto denominados *netlist*. Arquivos *netlist* são arquivos texto simplificados que contêm a representação de circuitos elétricos. Nestes arquivos, são especificados componentes, seus respectivos valores, os nós em que estão conectados e regras básicas de sintaxe que definem características de simulação.

O MATLAB, linguagem principal utilizada no desenvolvimento do sistema, utilizará essa característica padronizada dos arquivos *netlist* para criá-los e transmiti-los ao simulador de circuitos. As diferentes topologias serão avaliadas de acordo com suas diferentes características e sintaxes utilizadas, características e sintaxes estas que serão foco na Seção 3.4.2. Vale ressaltar, entretanto, como o padrão de escrita destes arquivos possibilita a realização de chamadas entre o MATLAB e o simulador de circuitos.

Para realizar as chamadas ao simulador será utilizada a função “system” que é capaz de executar um determinado comando como se o mesmo fosse executado no *prompt* de comando do Windows. Para realizar a simulação de um arquivo com a *netlist* do circuito a ser avaliado,

é necessário informar o local de instalação do simulador (*Spicepath*) e o nome do arquivo a ser simulado, como observado na Equação 2.

$$[Erro, Mensagem] = system([Spicepath, ' - b - ascii ', arquivo]) ; \quad (2)$$

O nome do netlist (arquivo) pode ser apenas o nome com a extensão, "nome.cir" por exemplo, ou o caminho completo da localização do arquivo, "C:\Pasta\nome.cir". O nome simplificado pode ser utilizado para casos em que o arquivo se encontra na mesma pasta das funções utilizadas, enquanto o caminho completo é indicado para arquivos localizados em locais diferentes ao da execução das funções do MATLAB.

Em conjunto com a localização do simulador de circuitos e com a localização do arquivo são utilizadas as opções "-b" e "-ascii". A opção "-b" é responsável por executar o simulador de circuitos em segundo plano no Windows, não demandando ações do usuário para cada topologia simulada.

Ao mesmo tempo, a opção "-ascii" é utilizada para que o simulador de circuitos entenda que deverá gerar os arquivos de resposta utilizando a codificação ASCII. Esta codificação, acrônimo de *American Standard Code for Information Interchange* – ASCII, traduzido como Código Padrão Americano para o Intercâmbio de Informação, faz com que a codificação utilizada pelo simulador na criação dos arquivos seja compreendida pelo Matlab; possibilitando a posterior leitura dos resultados.

Após a execução do comando, a função retorna um *flag* de erro (Erro) e uma eventual mensagem de erro na execução. Caso a execução seja realizada de modo satisfatório, a função retornará valor zero para o *flag* de erro (Erro) e a mensagem de erro (Mensagem) vazia. Ao realizar uma simulação de modo satisfatório o simulador de circuitos gera um arquivo da extensão "arquivo.raw". Esse será o arquivo que contém a amostragem do circuito avaliado e que deverá ser lido pelo MATLAB.

Caso a execução não seja realizada de modo satisfatório, a função retornará valor verdadeiro, igual a um, para o *flag* de erro (Erro) e uma breve descrição do erro na mensagem de erro (Mensagem). Maiores detalhes sobre a possibilidade de erros e o modo que estes erros foram corrigidos são o foco na Seção 3.8.

Uma vez que a localização do arquivo é crucial para a comunicação entre o simulador de circuitos e o MATLAB, foi concebido um modo de criação de arquivos que facilite a utilização pelos usuários do algoritmo.

2.3 Definição da localização dos Arquivos Utilizados

Os arquivos de texto, com as topologias de circuitos a ser testados, poderiam ser salvos na mesma pasta em que as funções do MATLAB são executadas sem perda de eficiência. Entretanto, a quantidade de arquivos já existentes na pasta raiz de execução somada com os arquivos gerados por cada simulação realizada torna pouco eficiente a verificação posterior desses arquivos. Dessa forma, para facilitar a utilização posterior dos arquivos, é gerado um diretório separado para cada simulação realizada.

Em cada simulação realizada, a localização da pasta com as funções do MATLAB é uma das variáveis definida pelo usuário do mesmo modo que o local de instalação do simulador de circuitos. Utilizando o nome do circuito, também informado pelo usuário, a plataforma desenvolvida cria uma pasta separada, no interior da pasta raiz de execução, para os arquivos produto da simulação realizada. Utilizando novamente a função "*system*" do MATLAB o algoritmo testa se a pasta com o nome especificado já existe e, caso não exista, cria a mesma. Isso permite a separação de todos os arquivos de uma nova simulação ou a atualização dos arquivos de uma simulação realizada anteriormente.

A pasta destino gerada armazenará o arquivo com a topologia a ser testada com a extensão ".cir", o arquivo com o log de simulação da topologia pelo simulador, com a extensão ".log", e o arquivo com a resposta amostrada do circuito simulado, com a extensão ".raw". Todos esses arquivos recebem o nome do circuito especificado pelo usuário. Em conjunto com esses arquivos são gerados os resultados da simulação como histórico de simulação, resposta final da melhor topologia evoluída e um relatório de evolução, arquivos estes discutidos com mais detalhes na Seção 3.9 sobre exportação de resultados.

Outra pasta é criada no interior do diretório específico de simulação com o nome "Erros", na qual os erros no decorrer da evolução são salvos. Tais erros foram utilizados para identificar e mitigar as fontes de erros no decorrer da realização deste trabalho. A identificação dos tipos de erros possíveis e suas causas são o assunto da Seção 3.8.3.

Todo esse processo da criação dos arquivos a serem simulados e avaliados é realizado em uma função central de avaliação. Essa função recebe o nome de "Função de Avaliação".

2.4 Função de Avaliação de Circuitos

A função de avaliação foi concebida com o objetivo de facilitar a avaliação independentemente de qual seja o circuito objetivo da evolução. Normalmente, a função de avaliação utilizada na evolução de circuitos deve ser atualizada a cada nova evolução que se deseja realizar. Nessa metodologia, o programador deve alterar a especificação de cada componente utilizado, a quantidade de componentes utilizados, as conexões entre os componentes e utilizar o algoritmo genético para evoluir apenas os valores dos mesmos.

Este trabalho visou desobrigar o usuário de alterar a função de avaliação a cada alteração de circuito alvo da evolução. O algoritmo possui três modos de evolução, modos esses explicados mais detalhadamente na Seção 3.7. No modo de evolução completa, o usuário deve apenas informar os tipos de componentes que serão utilizados e uma quantidade limite de componentes que devem ser utilizados. A partir dessas variáveis o algoritmo escreverá a *netlist* das topologias de circuitos a serem evoluídos sem necessidade de intervenção por parte do usuário.

O procedimento de criação automática das *netlists*, de acordo com os tipos especificados pelo usuário, é mais detalhado nas seções a seguir. A criação das *netlists* é realizada utilizando um cromossomo, que representa cada uma das possíveis soluções a serem evoluídas, e uma variável de opções, contendo todas as variáveis de importância na evolução de circuitos.

2.4.1. Transmissão de Informações Entre as Funções

A função de avaliação, na maioria das aplicações, recebe apenas o cromossomo, a ser avaliado, e retorna uma aptidão que é utilizada para classificar as possíveis soluções ao problema proposto.

Na plataforma desenvolvida, entretanto, foi necessário repassar à função de avaliação algumas variáveis que definem seu modo de comportamento, como observado abaixo na Equação 3. Tais opções são repassadas através de uma estrutura de variáveis com o nome “Opções” e permitem à função de avaliação retornar o valor de aptidão obtido (Retorno), a solução que resultou esse resultado (Solução), e uma variável de erro (Erro), responsável por informa se houve algum erro durante a avaliação.

$$[Solução, Retorno, Erro] = FunçãoAvaliação(Solução, Opções) \quad (3)$$

A Solução retorna como resultado, apesar de ser entrada da função, porque pode ser alterada ou corrigida pela função de avaliação. Dependendo das opções repassadas à função de avaliação, ela irá realizar a correção dos valores do cromossomo e evitar situações de inutilização da solução devido à má formação da topologia. As situações de inutilização de uma solução e as maneiras utilizadas para corrigi-las serão foco da Seção 3.8.3 e o impacto dessas correções no tempo de evolução será foco na conclusão do trabalho.

O retorno, no modo de operação normal da função de avaliação, representa um valor único de aptidão e será usado na classificação nos indivíduos de cada geração. Entretanto, dependendo das opções repassadas a função, a variável de retorno pode conter a resposta do circuito em todos os pontos amostrados e os respectivos pontos. Tal comportamento é utilizado nos casos de exportação de resultado, explicado mais detalhadamente na Seção 3.9, que permite mostrar ao usuário como a resposta daquele circuito específico é em comparação com a resposta desejada, objetivo da evolução.

A variável de erro é utilizada para identificar e classificar os erros que podem ocorrer no decorrer da evolução. Através desta variável foi possível verificar a incidência de erros no decorrer de cada geração da evolução e avaliar a eficiência das correções realizadas nas topologias simuladas. Essa verificação é realizada através da emissão, pela própria plataforma, de relatórios de execução para cada evolução realizada. A emissão de relatórios de evolução, contendo os erros ocorridos e outras grandezas de interesse, pode ser vista mais detalhadamente na Seção 3.8.2.

Utilizando o cromossomo informado e as opções informadas à função, o arquivo contendo a *netlist* de circuito é completamente gerado pela função de avaliação. Esse modo de funcionamento permite a avaliação de uma topologia específica apenas informando o cromossomo que representa a mesma, sem necessidade da alteração de funções e arquivos por parte do usuário.

2.4.2. Criação do Arquivo a Ser Simulado

A criação da *netlist* por parte da função de avaliação ocorre através de uma estrutura específica que é alterada de acordo com cada modo de evolução e o tipo de entrada utilizada para o circuito, modos de evolução e tipos de entradas apresentados nas seções 3.6 e 3.7, respectivamente. A estrutura base de escrita de uma *netlist* pode ser observada na Figura 3.

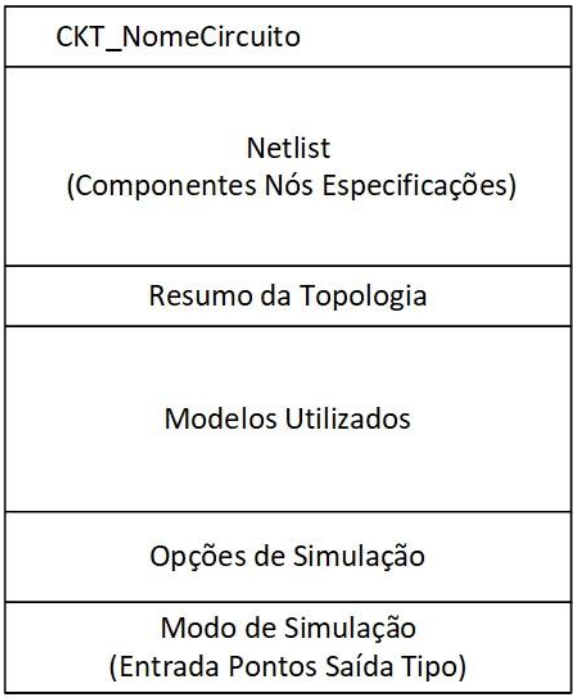

Figura 3 - Estrutura dos Arquivos Simulados
Fonte: O autor, 2019.

A partir das opções repassadas à função de avaliação, a primeira variável verificada é o nome do circuito a ser evoluído. Esse nome será usado para criar os diretórios citados anteriormente e que conterão os arquivos a serem simulados, a resposta à simulação e, posteriormente, os arquivos exportados ao final da execução. O nome também será utilizado para formar o título do arquivo, que deverá ser colocado na primeira linha do arquivo de texto da *netlist*.

Após dar nome ao arquivo, a função escreverá no arquivo todos os componentes que compõe a topologia a ser testada. Essa escrita poderá ser realizada utilizando exclusivamente os valores do cromossomo, para os casos de variação completa da topologia, ou usando o valor das conexões por uma variável dentro de opções e utilizar o cromossomo apenas para variar o

valor dos componentes, para os casos de variação apenas de valores de componentes. Os tipos de variações disponíveis e os consequentes modos de escrita da *netlist* no arquivo serão apresentados mais detalhadamente na 3.7.

Após a determinação de todos os componentes, a função apresenta um pequeno resumo da topologia apresentada acima. Este resumo contém a quantidade de componentes, seus tipos e o cromossomo utilizado para a escrita do mesmo. Estas informações são apresentadas em forma de comentários e podem ser usadas, por exemplo, para avaliação direta do cromossomo descrito na função de avaliação. Essa utilização permite a avaliação especifica de uma determinada topologia sem a necessidade de realizar o processo de evolução novamente, extraindo seus valores de saídas para demonstrações ou montagem de circuitos.

Os Modelos Utilizados são os componentes disponíveis para evolução. Componentes básicos, como resistores e capacitores, não necessitam de modelos descritivos de funcionamento. Entretanto, componentes básicos que se deseja especificar alguma característica e componentes mais complexos, como amplificadores operacionais, devem ser especificados com uma definição de modelo pré-determinada.

Após a escrita dos modelos de circuitos utilizados, a função escreve as opções que definem como a simulação deve ser realizada pelo simulador. Será apresentado na seção sobre erros de execução, Seção 3.8.3, como a determinação de alguns parâmetros de simulação *a priori* evita o aparecimento de determinadas falhas. Tais falhas interferem diretamente no tempo de evolução das simulações e, em determinados casos, trava a evolução devido a um único elemento.

Por fim, a função escreve as características do modo de simulação a ser realizado. Este modo é determinado pelo tipo de entrada utilizada (fonte DC ou fonte AC), os pontos de amostragem (Através do intervalo e passo de amostragem), o nó de saída avaliado e o tipo de avaliação que deve ser realizada (Avaliação DC ou no domínio da frequência). Após essa determinação, a escrita no arquivo é encerrada e o mesmo está pronto para ser simulado utilizando os componentes especificados.

2.4.3. Componentes Disponíveis na Plataforma

Com o objetivo de aproximar a resposta evoluída pelo sistema da resposta observada após implementação, optou-se pela utilização apenas de valores comerciais de componentes.

Essa escolha garante que as soluções de circuitos evoluídos pela plataforma não possuam componentes com “valores quebrados”, possibilitando a posterior implementação dos circuitos evoluídos em qualquer plataforma que o usuário deseje.

Para possibilitar a variação dos componentes apenas entre valores comerciais, a variação de cada componente do cromossomo foi realizada apenas entre números inteiros. As componentes referentes a nós de conexão do circuito são utilizadas diretamente do cromossomo e as componentes referentes a valores são utilizadas como índices, indexando os valores de componentes dentre os disponíveis.

A especificação do valor de resistores e de capacitores é dada por uma função de indexação especifica para cada um dos tipos. Utilizando o índice informado pelo cromossomo, a função de indexação retorna o valor comercial referente ao inteiro e esse valor é usado para a escrita da topologia do circuito. Neste procedimento, a determinação da faixa de valores disponíveis para evolução pode ser facilmente alterada atualizando os valores na função de indexação e nos limites de variação do cromossomo. A lógica de seleção para valores de resistores pode ser observada na Figura 4.

```
function x = indexacaoR(i)

valores = [1.0 1.1 1.2 1.3 1.5 1.6 1.8 2.0 2.2 2.4 2.7 3.0 3.3 3.6 3.9 4.3 4.7 5.1 5.6 6.2 6.8 7.5 8.2 9.1];
valores = [1E+03;1E+04;1E+05;1E+06;1E+07] .* valores;

x = valores(i);
```

Figura 4 - Indexação de Valores Comerciais de R

Fonte: O autor, 2019.

Os valores comerciais de R escolhidos correspondem à série de resistores com 5% de tolerância. Tais valores correspondem aos múltiplos dos valores presentes na Tabela 2. Os valores escolhidos da série foram valores da tabela múltiplos da faixa de 10^3 a 10^7. Ou seja, a faixa de valores que vai de 1 kΩ a 91 MΩ.

Tabela 2 - Valores Comerciais de Resistores

1,0 Ω	1,1 Ω	1,2 Ω	1,3 Ω	1,5 Ω	1,6 Ω
1,8 Ω	2,0 Ω	2,2 Ω	2,4 Ω	2,7 Ω	3,0 Ω
3,3 Ω	3,6 Ω	3,9 Ω	4,3 Ω	4,7 Ω	5,1 Ω
5,6 Ω	6,2 Ω	6,8 Ω	7,5 Ω	8,2 Ω	9,1 Ω

Fonte: O autor, 2019.

Os valores comerciais de capacitores correspondem aos múltiplos da mesma tabela de valores que os resistores. Entretanto, os valores utilizados são os múltiplos da faixa de 10^{-8} a 10^{-12}. Dessa forma, a faixa de valores dos capacitores que compõe a plataforma vai de 10 nF a 9.1 pF.

Outros componentes, como transistores BJTs e *MOSFETs*, são definidos na parte de modelos utilizados para determinação de alguma característica interna. Por outro lado, componentes como o amplificador operacional necessitam de uma descrição mais detalhada para ser usado na evolução de topologias.

O amplificador operacional utilizado foi retirado de uma base de circuitos online voltada para o LTSpice (ENGELHARDT; 2004). O modelo escolhido considera a faixa limitada de valores de tensão em que o funcionamento do componente é linear, importante na implementação de circuitos reais. O modelo utilizado pode ser observado na Figura 5.

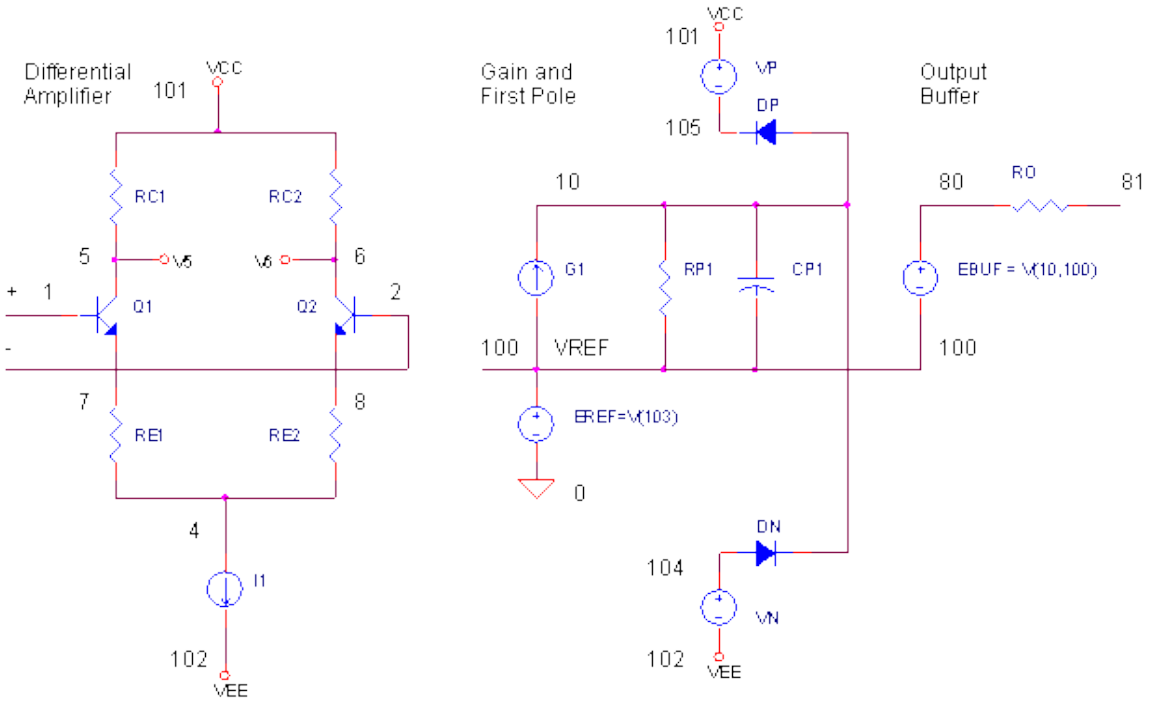

Figura 5 - Modelo de Amplificador Operacional Utilizado nas Evoluções

Fonte: <www.ecircuitcenter.com/OpModels/V_Limit/opmodel3 V Limit.html>. Acesso em: 24 ago. 2019.

A limitação de valores da saída de um amplificador operacional deve-se a limitações de arquitetura e, principalmente, aos valores de alimentação do mesmo. O componente é capaz de amplificar o sinal a ele fornecido até uma tensão limite denominada tensão de saturação (TS), diretamente relacionada com suas tensões de alimentação superior e inferior. Este comportamento pode ser observado na Figura 6.

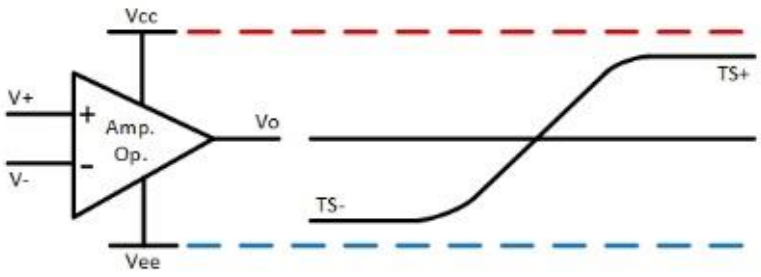

Figura 6 - Funcionamento de um Amplificador Operacional Considerando Saturação
Fonte: O autor, 2019.

Por se tratar de uma variável diretamente relacionada ao desempenho do circuito, as tensões de alimentação do amplificador operacional são evoluídas ao mesmo tempo que os valores dos demais componentes. Diferentemente do valor de resistores e capacitores, o índice evoluído é utilizado diretamente na função de avaliação e garante que o valor das fontes de alimentação sejam um múltiplo de meio volt até um valor determinado nos limites de variação. Dessa forma, a evolução de circuitos considerará o comportamento de saturação na saída dos amplificadores operacionais utilizados e aproximará a resposta evoluída à resposta obtida pela implementação real do circuito.

Outro componente disponível na plataforma desenvolvida também possui limitação em suas especificações de evoluções. As fontes evoluídas possuem um limite na tensão que podem assumir, limite este definido pelo usuário. Dessa forma, o usuário pode definir a quantidade máxima de valores unitários de tensão que uma fonte utilizada pode possuir.

Através destes componentes disponíveis, o algoritmo irá buscar a topologia que mais se aproximará da resposta desejada pelo circuito. Entretanto, dependendo do espaço de busca, essa busca pode se tornar bastante demorada. Para otimizar a busca de uma solução, pode-se utilizar uma solução conhecida (circuito semente) e partir dela na procura de uma solução melhor.

2.4.4. Utilização de Um Circuito Semente

A plataforma desenvolvida utilizará a característica elitista do algoritmo base para evoluir as topologias analisadas. Nessa característica, a melhor solução será mantida inalterada

de uma geração para a seguinte, garantindo que a melhor solução encontrada não seja perdida no decorrer da evolução.

Em conjunto com a característica elitista, uma alteração na função de inicialização da população foi realizada. Nesta alteração, duas variáveis são repassadas dentro de opções às demais funções, “PossuiSolucaoConhecida” e “SolucaoConhecida”. A primeira informa à função de inicialização se existe uma solução que servirá de base para a evolução. Caso exista tal solução, ela estará na segunda variável e será introduzida na população inicial, servindo de base para o restante da evolução.

A utilização de uma possível solução no início da execução pode acelerar consideravelmente o processo de evolução. Tal funcionalidade permite ao usuário, conhecendo um determinado circuito cuja saída obedeça ao mesmo padrão da saída desejada, utilizar o algoritmo de evolução para alterar o valor dos componentes para encontrar uma topologia satisfatória. O impacto da utilização de uma solução inicial na evolução de uma topologia será observado mais detalhadamente no decorrer do trabalho.

A solução semente acelerou bastante o processo de evolução, mas ainda demandava que o usuário aguardasse todo o processo para avaliar se a busca pela nova topologia estava ocorrendo de forma satisfatória. Essa avaliação durante a execução da evolução será apresentada no decorrer do trabalho junto com outras variáveis utilizadas no funcionamento da plataforma.

A utilização de múltiplas variáveis modificadoras na plataforma visa possibilitar uma rápida customização de seu funcionamento de acordo com o desejo do usuário. Tais customizações podem então ser realizadas de forma rápida e sem necessidade de alterações no código fonte desenvolvido para a plataforma.

A plataforma possibilita futuros estudos sobre os impactos dos diferentes modos de realização de uma evolução. Tais estudos podem avaliar o impacto da utilização de soluções semente, locais de criação e utilização dos arquivos e características modificadoras do processo de evolução na resposta observada ao final do processo.

Outras funcionalidades foram adicionadas à plataforma desenvolvida. Tais funcionalidades serão apresentadas no decorrer deste trabalho. Entretanto, um resumo das variáveis repassadas à plataforma pode ser observado na Figura 7. Tais variáveis são comuns a todos os modos de funcionamento, que serão apresentados na seção 3.6. Através da imagem é possível observar a capacidade de customização inicial da plataforma pelo usuário. Tal customização é realizada diretamente pelas variáveis descritas, sem necessidade de alteração do código da plataforma.

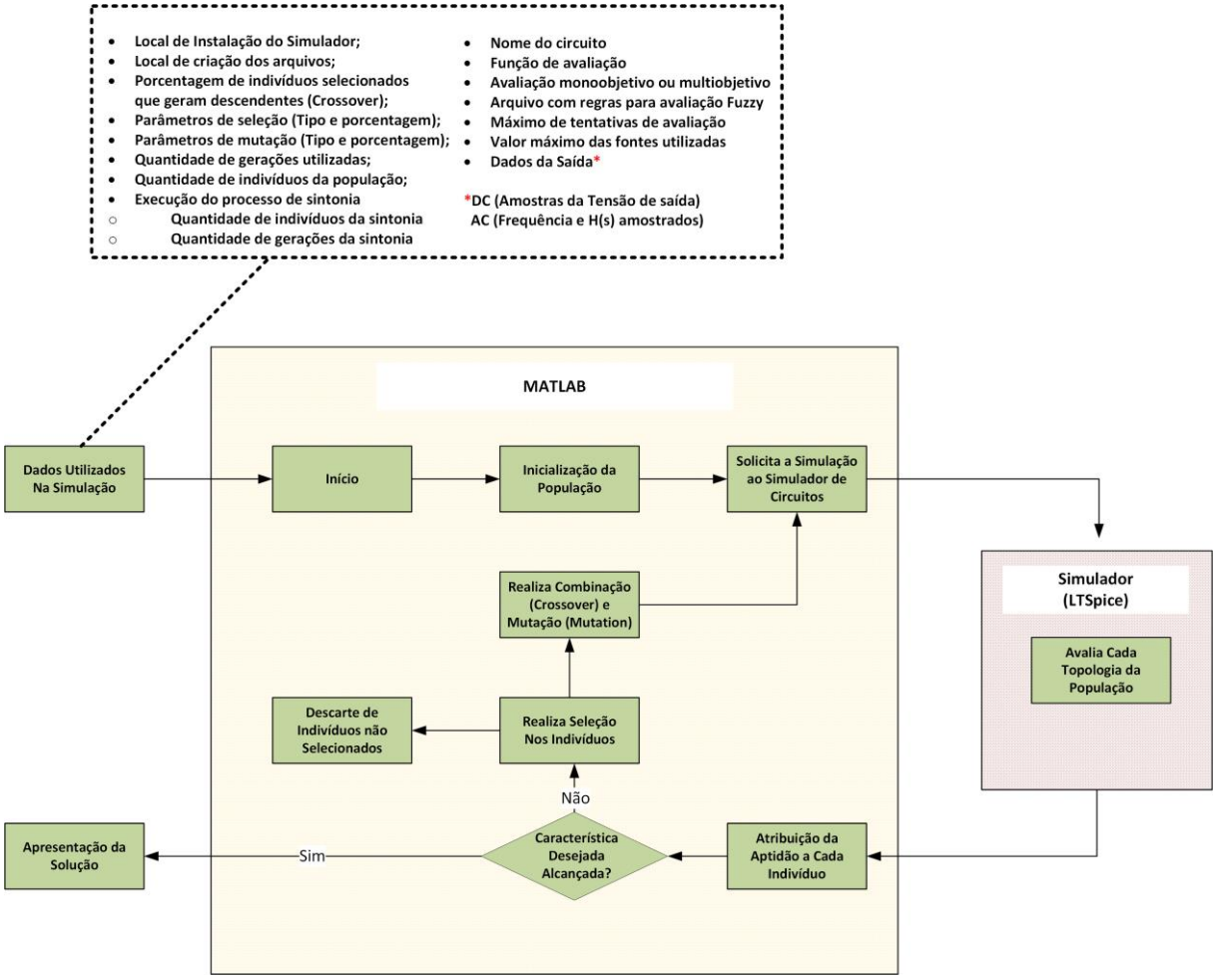

Figura 7 – Fluxograma de Funcionamento da Plataforma com Resumo de Variáveis

Fonte: O autor, 2019.

2.5 Apresentação do Resultado Parcial de Evolução

Com o objetivo de fornecer um retorno mais rápido ao usuário, a função de evolução base do algoritmo escolhido foi alterada. Na alteração, o usuário observa um gráfico com o valor do melhor indivíduo, e a média da população, da geração atual e das anteriores. Este gráfico, observado na Figura 8, permite que usuário avalie se a evolução está ocorrendo normalmente e, caso necessário, encerre a evolução para alterar algum parâmetro ou opção.

Figura 8 - Janela Com Resultado Parcial da Evolução

Fonte: O autor, 2019.

Uma das formas de identificar se a evolução ocorre de modo satisfatório é observar se a média da população melhora no decorrer das gerações. A permanência da melhor solução em um mesmo valor por muitas gerações, e distante do resultado desejado, também é um forte indício de que algum parâmetro poderia ser alterado para melhorar a busca por soluções. Tal avaliação dependerá, também, do modo de evolução escolhido pelo usuário, diretamente relacionado com o espaço de busca da topologia.

2.6 Modos de Evolução de Um Circuito

A plataforma desenvolvida possui três modos de evolução disponíveis ao usuário. Cada modo de funcionamento terá um tipo específico de cromossomo e, consequentemente, um espaço de busca proporcional. A escolha do modo de operação e o fornecimento das variáveis correspondentes ao modo são feitos antes da execução do processo de evolução.

2.6.1 Modo Variação de Componentes

O modo de evolução Variação de Componentes é utilizado nos casos em que a topologia a ser evoluída é fixa e apenas o valor dos componentes é evoluído. Nesse modo de evolução, o usuário informa a topologia fixa à função de avaliação e os tipos de componentes utilizados e a evolução buscará o melhor conjunto de valores para a saída desejada.

A topologia fixa consiste em uma estrutura similar à do cromossomo e contém a quantidade de componentes que a formam, o tipo de cada componente e os nós em que este componente está conectado. Essa estrutura pode ser observada na Figura 9.

Circuito Fixo	Nº Componentes	Tipo	1º Nó	2º Nó	3º Nó	Tipo	1º Nó	2º Nó	3º Nó	...	Tipo	1º Nó	2º Nó	3º Nó	Nó Saída
	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	...	Valor	Valor	Valor	Valor	Valor
		1º Componente				2º Componente				...	Componente n				

Figura 9 - Circuito Fixo Base Para a Evolução de Valores

Fonte: O autor, 2019.

O tipo de cada componente do circuito fixo é repassado em formato numérico dentro da variável de circuito fixo. Em conjunto com a topologia fixa, é informada à plataforma os tipos de componentes que farão parte do circuito. Cada tipo é representado por uma letra, padronizada com a função de avaliação, e que será utilizada pela mesma para escrever cada *netlist* de cada topologia a ser avaliada. A tabela de componentes, e suas respectivas representações, podem ser observadas na Tabela 3.

Tabela 3 - Representação dos Tipos de Componentes Disponíveis

Tipo de Componente	Representação
Resistor	R
Capacitor	C
Fonte DC	V
Amplificador Operacional	A
Transistor BJT	B
Transistor Mosfet	M

Fonte: O autor, 2019.

A partir da topologia fixa fornecida à função de avaliação, a função usará o cromossomo, evoluído no decorrer desse modo de operação, para determinar o valor de cada componente na busca de uma solução. O cromossomo, para esse modo de evolução, será composto pelo valor do componente e as duas fontes de alimentação, para os casos em que o componente é um amplificador operacional. A estrutura do cromossomo utilizado pode ser observada na Figura 10.

Cromossomo	1ª Variável	2ª Variável	3ª Variável	4ª Variável	5ª Variável	6ª Variável	...	Variável 3n-2	Variável 3n-1	Variável 3n
	Valor	Vcc	Vee	Valor	Vcc	Vee	...	Valor	Vcc	Vee
	1º Componente			2º Componente			...	Componente n		

Figura 10 - Cromossomo do Modo Variação de Componentes

Fonte: O autor, 2019.

Neste modo de evolução, o espaço de busca é minimizado uma vez que é composto apenas pelos valores dos componentes utilizados. A equação que calcula o espaço de busca na utilização deste modo é apresentada na Equação 4 e será utilizada nos estudos de caso do Capítulo 5. Ela considerará os limites de cada variável, limites esses gerados automaticamente pelo algoritmo a partir da quantidade máxima de componentes utilizada.

$$\text{Espaço Busca} = \text{Nº Valores}^{\text{Nº Componentes}} \ \therefore$$
$$\text{Espaço Busca} = \text{Vr}^{\text{Nr}} + Vc^{Nc} + Vf^{Nf} + (Vcca \,.\, Veea)^{Na} \quad (4)$$

Onde:

Nr = Número de Resistores

Vr = Quantidade de Valores Para Resistores

Nc = Número de Capacitores

Vc = Quantidade de Valores Para Capacitores

Nf = Número de Fontes

Vf = Quantidade de Valores Para Fontes

Na = Número de Amplificadores Operacionais

Vcca = Quantidade de Valores Para Vcc de Amp. Ops.

Veea = Quantidade de Valores Para Vcc de Amp. Ops.

O fluxograma de funcionamento da plataforma pode ser atualizado para o Modo Variação de Componentes. Nesse modo a plataforma utilizará um circuito fixo para descrever os tipos e conexões dos componentes e evoluirá apenas o valor dos mesmos. Esse fluxograma pode ser observado abaixo na Figura 11.

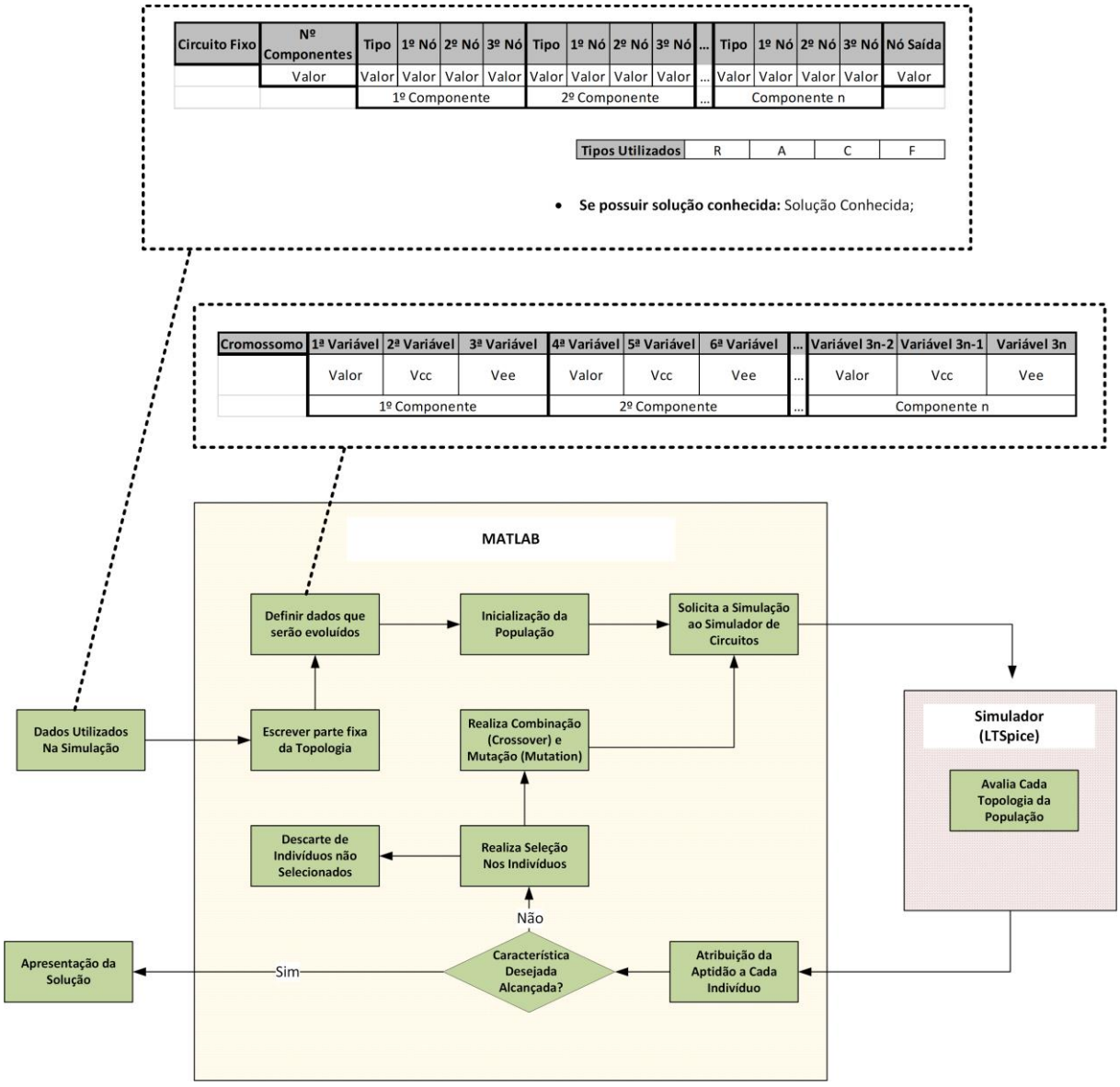

Figura 11 - Fluxograma de Funcionamento Para o Modo Variação de Componentes

Fonte: O autor, 2019.

2.6.2 Modo Adição de Componentes

O modo de evolução Adição de Componentes é utilizado nos casos em que existe uma topologia fixa inicial e o algoritmo de evolução adicionará componentes a essa topologia, com

o objetivo de alcançar a saída desejada. Nesse modo de evolução, o usuário informa a topologia fixa à função de avaliação e os tipos de componentes utilizados. A partir daí a evolução buscará uma topologia complementar que se aproxime da saída desejada.

A topologia fixa consistirá em uma estrutura similar à topologia fixa do modo anterior (Variação de Componentes). Entretanto, neste modo ela conterá os valores de cada componente somados as informações de quantidade de componentes e nós de conexão. A estrutura atualizada, utilizada neste modo de evolução, pode ser observada na Figura 12.

Circuito Fixo	Nº Componentes	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	...	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	Nó Saída
	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	...	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
		1º Componente							2º Componente							...	Componente n							

Figura 12 - Circuito Fixo Base o Modo Adição de Componentes

Fonte: O autor, 2019.

O tipo de cada componente, definido junto com o circuito fixo, também será utilizado na definição dos novos componentes adicionados. Após a escrita do circuito fixo, o algoritmo de evolução adicionará elementos a ele até uma quantidade máxima de elementos, quantidade esta definida pelo usuário em momento anterior ao processo de evolução.

O cromossomo utilizado neste modo de evolução, assim como o modo de evolução anterior, conterá o valor de cada componente e o valor de fontes de alimentação para caso o componente seja um amplificador operacional. Tais valores e fontes utilizadas nos componentes variam de acordo com os tipos utilizados. Somado a estes valores, o cromossomo conterá a quantidade e o posicionamento de cada componente na topologia final. Esse posicionamento considera os nós já existentes da topologia fixa e uma quantidade máxima de nós extras que podem ser adicionadas a topologia final. A estrutura do cromossomo deste novo modo de operação pode ser observada na Figura 13.

Cromossomo																								
Nº Componentes	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	...	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee
Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	...	Valor	Valor	Valor	Valor	Valor	Valor	Valor
	1º Componente							2º Componente							...	Componente n						

Figura 13 - Cromossomo do Modo Adição de Componentes

Fonte: O autor, 2019.

No modo de evolução apresentado, o espaço de busca é consideravelmente maior que o modo anterior. Esse aumento do espaço de busca ocorre pelo fato dos nós de posicionamento fazerem parte da evolução. O cálculo das possibilidades de formação de uma topologia, espaço de busca de soluções, considera que os nós de um componente podem ocupar qualquer um dos nós existentes. Na Seção 3.8.3 é discutido como o curto dos terminais de um componente pode não impactar, como no caso de resistores em que o mesmo é ignorado, ou pode comprometer a avaliação de toda a topologia, como no caso de fontes. Entretanto tais casos devem ser considerados no cálculo de possíveis formações de circuitos. A equação que representa este novo espaço de busca pode ser observada na Equação 5.

$$\begin{aligned} &\text{Espaço Busca} = \text{Nº Componentes} \ \therefore \\ &\text{Espaço Busca} = (\text{Nº Componentes x Nº Valores}) \ \therefore \\ &= \text{MaxComp} \,.\, \text{NTipos} \,.\, \text{NmaxNós} \,.\, \text{NmaxNós} \,.\, \text{NmaxNós} \,.\, \text{Val} \,.\, \text{Vcca} \,.\, \text{Veea} \end{aligned} \tag{5}$$

Onde:

MaxComp. = Número máximo de componentes disponíveis

NTipos = Quantidade de tipos de componentes utilizados

NmaxNós = Número máximo de nós da topologia

Val = Quantidade de Valores para os componentes

Vcca = Quantidade de Valores Para Vcc de Amp. Ops.

Veea = Quantidade de Valores Para Vcc de Amp. Ops.

Da mesma forma que o modo anterior, o fluxograma de funcionamento da plataforma pode ser atualizado para o Modo Adição de Componentes. Nesse modo a plataforma utilizará um circuito fixo, como estrutura inicial, e adicionará componentes a esse circuito com o objetivo de otimizar a resposta observada em relação a uma resposta desejada. Esse modo de evolução pode ser entendido como uma evolução a falhas de funcionamento, que busca corrigir a resposta de um circuito com a adição de componentes a sua topologia inicial. O fluxograma resumo com as variáveis utilizadas pelo Modo Adição de Componentes pode ser observado abaixo na Figura 14.

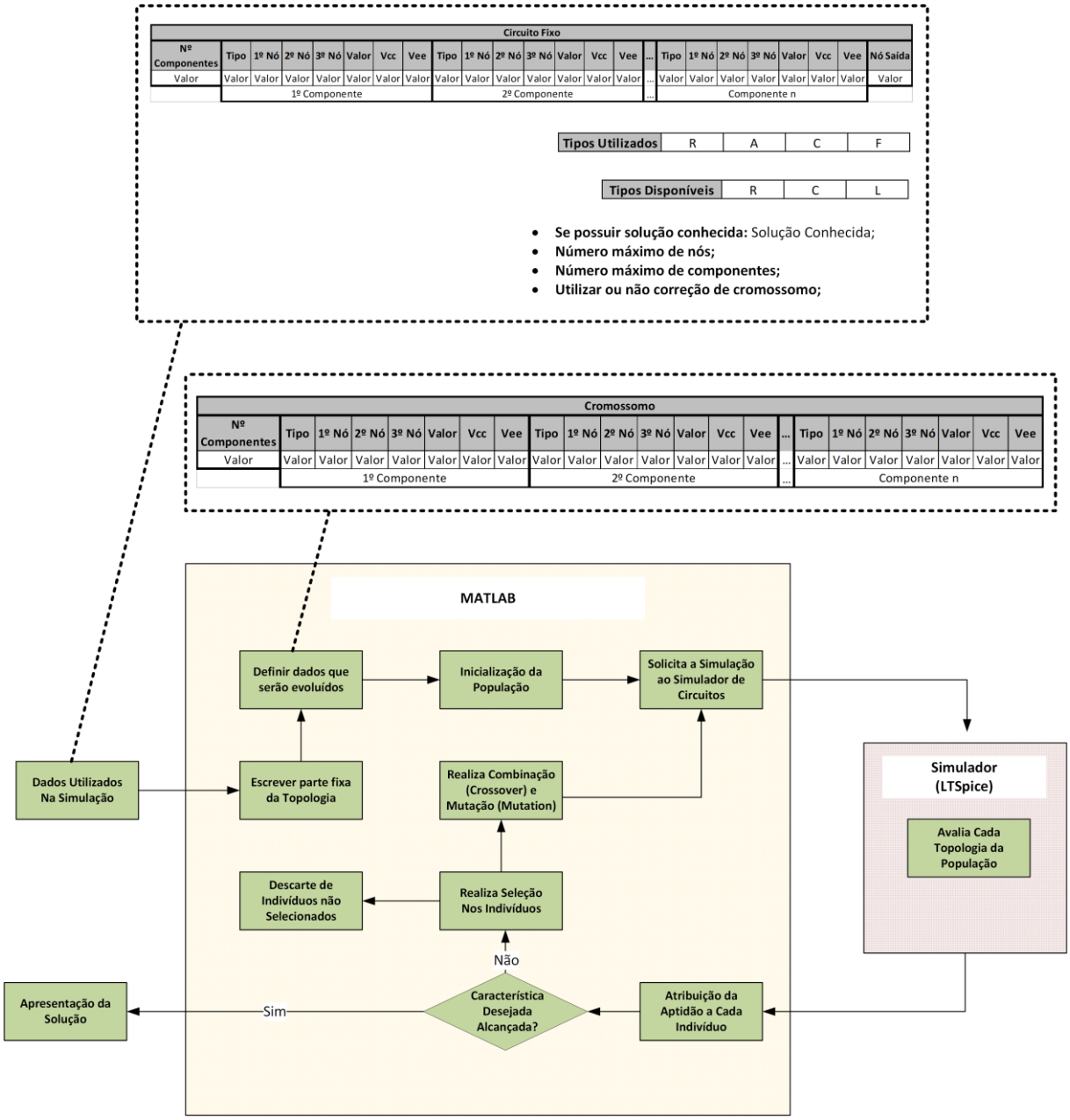

Figura 14 - Fluxograma de Funcionamento Para o Modo Adição de Componentes

Fonte: O autor, 2019.

2.6.3 Modo Variação Completa

O modo de evolução Variação Completa é utilizado nos casos em que o usuário deseja realizar uma evolução completa em busca de uma solução. Neste modo o usuário informa uma quantidade máxima de componentes que deverão ser utilizados e uma quantidade máxima de nós em que estes componentes deverão estar conectados. Estas variáveis, em conjunto com as

variáveis já utilizadas nos modos anteriores, como a quantidade de valores de componentes disponíveis, determinarão a evolução a ser realizada. A partir daí a evolução buscará uma topologia completa, que deverá se aproximar da saída desejada.

O tipo de cada componente, assim como nos modos anteriores, é definido pelo usuário em um momento anterior à execução da evolução. A partir dos tipos de componentes disponibilizados ao algoritmo, este realizará uma evolução completa da topologia, alterando até mesmo o nó definido como saída do circuito.

O comportamento de avaliar a saída em diversos nós do circuito é justificado pela própria natureza de evolução do modo. Uma vez que toda a topologia é modificada durante a evolução, pode ocorrer de uma determinada construção satisfatória ser alcançada, mas com a saída diferente de um determinado ponto especificado. Este caso resultaria na perda desta construção pelo fato da aptidão ser verificada apenas em um nó específico. Dessa forma, otimiza-se a busca de soluções considerando as diferentes formas que um circuito pode ser montado.

O cromossomo deste modo de evolução, assim como o modo de evolução anterior, conterá a quantidade de componentes utilizados, seus valores de especificação e fontes de alimentação e os pontos em que estão conectados. Somada a estes valores, estará a variável responsável por indicar qual a posição da saída que deverá ser avaliada. Dessa forma, a estrutura de cromossomo que representa este modo de evolução será a apresentada na Figura 15.

Cromossomo																									
Nº Componentes	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	...	Tipo	1º Nó	2º Nó	3º Nó	Valor	Vcc	Vee	Nó Saída
Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor	...	Valor	Valor	Valor	Valor	Valor	Valor	Valor	Valor
	1º Componente							2º Componente							...	Componente n							

Figura 15 - Cromossomo do Modo Variação Completa de Componentes

Fonte: O autor, 2019.

Para o modo de evolução apresentado, o espaço de busca é maior do que o modo anterior, pois considera os diferentes nós para a saída. Dessa forma, a equação que representa este novo espaço de busca pode ser observada na Equação 6.

$$\begin{aligned} \text{Espaço Busca} &= \text{Nº Componentes x Saídas} \ \therefore \\ \text{Espaço Busca} &= (\text{Nº Componentes x Nº Valores}) \text{ x Saídas} \ \therefore \\ &= \text{MaxComp} \,.\, \text{NTipos} \,.\, \text{NmaxNós} \,.\, \text{NmaxNós} \,.\, \text{NmaxNós} \,.\, \text{Val} \,.\, \text{Vcca} \,.\, \text{Veea x Saídas} \end{aligned} \tag{6}$$

Onde: MaxComp. = Número máximo de componentes disponíveis

NTipos = Quantidade de tipos de componentes utilizados

NmaxNós = Número máximo de nós da topologia

Val = Quantidade de Valores para os componentes

Vcca = Quantidade de Valores Para Vcc de Amp. Ops.

Veea = Quantidade de Valores Para Vcc de Amp. Ops.

Saídas = Quantidade de nós possíveis para observação da saída.

Da mesma forma que no modo anterior, o fluxograma de funcionamento da plataforma pode ser atualizado para o Modo Variação Completa de Componentes. O fluxograma resumo com as variáveis utilizadas pelo Modo Adição de Componentes pode ser observado abaixo na Figura 16.

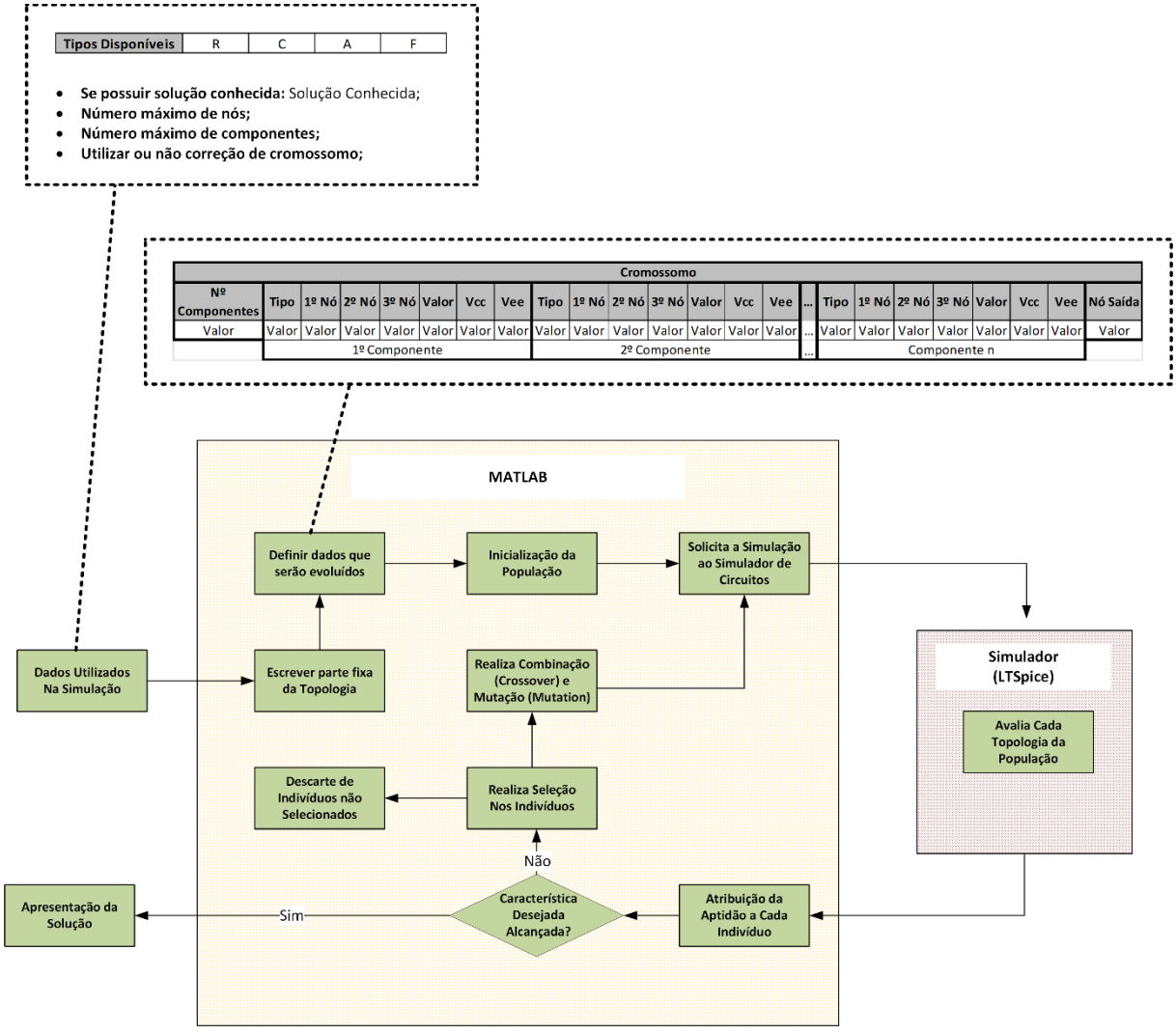

Figura 16 - Fluxograma de Funcionamento Para o Modo Variação Completa de Componentes

Fonte: O autor, 2019.

Após a escrita de todos os componentes utilizados, a função de avaliação realiza a escrita dos modelos utilizados e opções de simulação que serão comuns a todos as *netlists* analisadas. Em seguida, a função realizará a escrita referente ao Modo de Simulação. O Modo de simulação consiste em determinar valor e tipo da entrada utilizada, os pontos de amostragem a ser analisados e o modo e posição de análise da saída.

2.7 Tipos de Entradas Permitidas

A plataforma desenvolvida permite que a análise seja realizada utilizando dois tipos de análises, análise DC (corrente contínua) e análise AC (corrente alternada), sendo a análise de entradas de corrente alternada realizada no domínio da frequência.

Cada um dos tipos terá uma característica de especificação das fontes e dos nós de amostragem utilizados, bem como características específicas no repasse dos resultados simulados para o algoritmo.

2.7.1 Análise DC

O primeiro passo para a realização da Análise DC é a especificação da fonte utilizada. Esta especificação é realizada determinando um nome para fonte utilizada, o nó conectado ao terminal positivo da fonte, o nó conectado ao terminal negativo e o valor máximo que a mesma poderá assumir. Os nós de conexão da fonte de entrada são, por padrão, o nó um para o terminal positivo e o terminal zero para o terminal negativo da fonte.

Os valores máximo e mínimo que a fonte poderá assumir são variáveis determinadas pelo usuário antes do processo de evolução. A partir destas variáveis, o algoritmo analisa a amostragem disponível para evolução, presente na saída desejada, para determinar o passo de amostragem a ser utilizado. Isto dispensa o usuário de calcular o passo necessário para realizar a simulação, avaliando diretamente esse valor da amostragem fornecida para a evolução. Definidos os parâmetros de simulação, o algoritmo escreve o pedido de simulação reconhecido pelo LTSpice.

Este tipo de análise é reconhecido e realizado utilizando-se o comando ".dc" do LTSpice. Neste comando, deve-se informar o nó que deverá ser avaliado pela simulação e o tipo de grandeza que será avaliada. Para a evolução de um único objetivo, a grandeza avaliada será a tensão no ponto determinado. A determinação do nó de saída será de acordo com o modo de evolução escolhido pelo usuário, sendo fixa para os modos de evolução "Variação de Valores" e "Adição de Componentes" e variável para o modo "Variação Completa".

A estrutura básica de uma simulação realizada utilizando análise de corrente contínua pode ser observada na Figura 17.

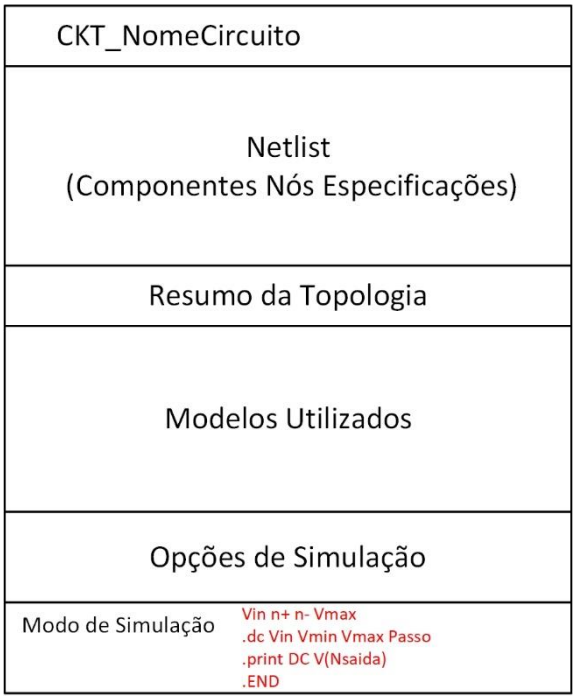

Figura 17 - Modo de Simulação na Análise DC

Fonte: O autor, 2019.

O modo de simulação trabalha com amostragens de simulação igualmente distribuídas, o que facilita a identificação de casos de erro, como explicado mais detalhadamente na Seção 3.8.3. Tal comportamento já não pode ser observado quando a análise é realizada no domínio da frequência.

2.7.2 Análise no Domínio da Frequência

A análise no domínio da frequência utiliza fontes de corrente alternada e a resposta do circuito é dada em relação à variação da frequência. Tal comportamento é objetivo de interesse principalmente na evolução e na correção de filtros de sinais. A plataforma desenvolvida realiza a evolução do ganho em relação à frequência, entretanto, o ângulo dos circuitos evoluídos é mostrado ao final da evolução.

Do mesmo modo que a análise anterior, o primeiro passo para realizar a análise no domínio da frequência é determinar a fonte a ser utilizada. As variáveis referentes à fonte de entrada são a tensão de amplitude da fonte e a frequência de variação da mesma. Outra variável é a existência de uma tensão de *offset* para a mesma, ou seja, um deslocamento ou nível DC para a mesma.

Após a determinação da fonte a ser utilizada, o usuário determina o intervalo de frequências em que o circuito evoluído deve ser analisado. Esse intervalo é determinado pelos valores máximo e mínimo de variação e por um passo que determinará a quantidade de pontos avaliados.

Diferentemente do comportamento na análise DC, o LTSpice não realiza a amostragem estritamente nos pontos especificados para análises AC. O simulador utiliza o passo informado como referência no decorrer da simulação, entretanto, caso ocorram variações mais abruptas no sinal, o simulador aumentará a amostragem independente do passo determinado. Dessa forma, a saída simulada pelo LTSpice é interpolada antes de ser utilizada pelo MATLAB para a avaliação da aptidão de cada topologia. O MATLAB recebe os dados de simulação vindos do LTSpice e adequa tais valores para os pontos de interesse definidos pela variável Saída Desejada. Essa adequação é realizada pela função “interp1” nativa do MATLAB e garante que a avaliação da aptidão, e consequentemente do erro observado, se atente aos pontos de amostragem informados pelo usuário.

Uma vez definidas as variáveis de simulação, a solicitação de uma análise AC é realizada pela função através da escrita na *netlist* criada. Essa solicitação é reconhecida e realizada utilizando o comando “.AC” no LTSpice. Neste comando, do mesmo modo que na análise DC, deve-se informar o nó que deverá ser avaliado pela simulação e o tipo de grandeza que será avaliada.

Para a evolução de um único objetivo, a grandeza avaliada será somente a tensão no ponto determinado e a determinação do nó de saída será de acordo com o modo de evolução

escolhido pelo usuário. Dessa forma, a estrutura básica de uma simulação realizada utilizando entrada de corrente alternada pode ser observada na Figura 18.

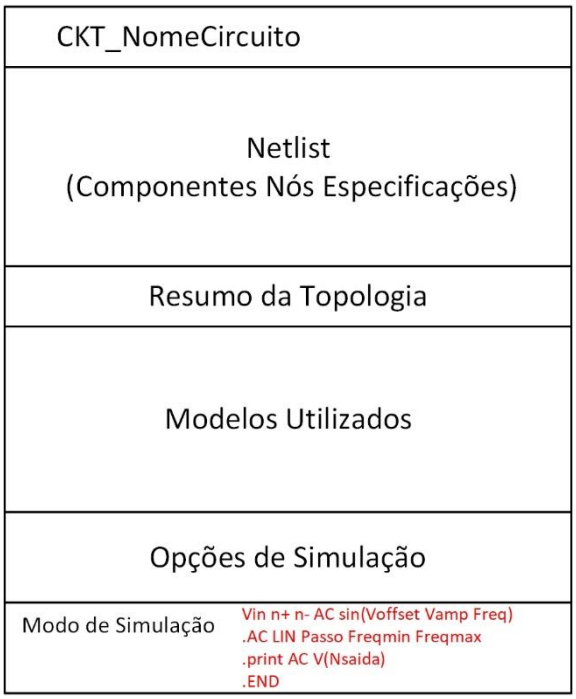

Figura 18 - Modo de Simulação na Análise AC

Fonte: O autor, 2019.

Uma vez determinada a estrutura de escrita da *netlist* enviada ao simulador de circuitos, o algoritmo realizará a escrita da mesma para cada indivíduo no decorrer do processo de evolução. Essa simulação da *netlist* deverá retornar uma aptidão, que será utilizada na classificação de cada indivíduo na busca de uma solução ideal.

Entretanto, apesar de obedecer a estrutura básica reconhecida pelo simulador, as topologias avaliadas podem resultar em um erro de funcionamento. Essa possibilidade de ocorrência de erros demandou uma análise mais detalhada do funcionamento da plataforma e um procedimento de mitigação de falhas.

2.8 Mitigação de Problemas de Desempenho na Evolução

Durante a realização das evoluções, situações de falhas de funcionamento foram identificadas e, em alguns casos, comprometiam toda a evolução realizada anteriormente. Tais

situações compreenderam desde valores inválidos de resultados até travamentos ou interrupção abrupta do processo de evolução.

Visando identificar as causas de tais falhas e buscar técnicas preventivas, uma análise minuciosa do funcionamento de plataforma foi realizada e classificada na tentativa de uma futura busca de soluções. Essa análise encontrou fontes de erros em funções base do algoritmo evolutivo utilizado, funções essas que necessitaram de modificações para a utilização na plataforma.

2.8.1 Alterações nas Funções Originais do GAOT

Durante os testes de evolução realizados pela plataforma, foi verificado que o algoritmo retornava valores incompatíveis com os determinados pelo código. Tais valores consistiam em valores reais no cromossomo apesar de, como apresentado anteriormente, o cromossomo utilizar apenas valores inteiros de variação.

Através de uma avaliação mais detalhada foi constatado que duas funções apresentavam falhas na execução, as funções “b2f” e “f2b”. Essas funções são responsáveis por converter, durante a evolução, os valores do cromossomo para binário e, após os processos de crossover e mutação, retorná-los para a representação utilizada. Entretanto, apesar de a variável responsável por determinar o intervalo de variação (épsilon) ser definida com um valor unitário, o algoritmo permitiu a variação das grandezas em valores fracionados.

Esse erro de funcionamento fez com que a evolução dos componentes ocorresse em valores fracionados, aumentando drasticamente o espaço de busca e, consequentemente, o tempo necessário para a evolução.

Outro erro encontrado nas funções utilizadas como base de evolução foi com a função “calcbits”. Essa função, disponibilizada pelo algoritmo base de evolução, realiza o cálculo da quantidade de bits necessária para representação dos valores na base binária. Como citado na apresentação do algoritmo base, o mesmo utiliza as variáveis em representação binária para realizar as operações de cruzamento e mutação. Entretanto, para intervalos de variáveis igual ou menor a dois a função apresentava erro na realização do cálculo. Dessa forma, para evitar erro do algoritmo, por exemplo, em evoluções que utilizem apenas dois tipos de componentes, tal função base também teve que ser alterada.

A função “ga” utilizada como base também foi alterada devido ao processo de mutação. Conforme mencionado anteriormente, o algoritmo define uma probabilidade de ocorrer mutação durante o processo de cruzamento (*crossover*) de indivíduos. Com base nessa probabilidade, o algoritmo seleciona descendentes e aplica o processo de mutação binária a eles. Entretanto, como a plataforma pode ser customizada a trabalhar com quantidades baixas de indivíduos, principalmente durante o processo de sintonia, pode ocorrer de o algoritmo trabalhar com gerações sem indivíduos selecionados para mutação. Tal ocorrência acarretava em erro da função original ao trabalhar com conjuntos vazios de indivíduos, demandando atualização no código.

Além das falhas apresentadas por funções do MATLAB, a análise das fontes de erro verificou falhas sucessivas na evolução de topologias pelo simulador de circuitos. Em conjunto com a identificação de cada um dos focos de erros nas topologias enviadas ao simulador, foi constatada a necessidade de quantificar a ocorrência destes erros no decorrer do processo de evolução.

Com o objetivo de armazenar e categorizar os erros no decorrer da evolução, uma nova variável foi adicionada ao algoritmo de evolução. Essa variável, denominada “HistoricoErros”, contém um histórico de cada erro ocorrido, e a geração em que ocorreu, durante todo o processo de evolução. Tal variável foi fundamental na avaliação de eficiência das medidas preventivas de erros e avaliação do impacto desses erros no processo de busca de soluções. A partir da mesma é exportado um relatório de execução, utilizado para analisar a incidência de erros na evolução.

2.8.2 Criação do Relatório de Evolução

O relatório gerado ao final de cada evolução busca resumir de forma especifica as variáveis de importância da execução e servir de base para análises mais específicas. O mesmo também foi utilizado como meio de verificação se a identificação de um erro, e o respectivo modo de resolução, foram corretamente identificados durante a criação da plataforma.

A estrutura básica do relatório emitido ao final de cada evolução pode ser observada na Tabela 4. Nela podem ser observadas as informações sobre o melhor elemento descoberto, a cada geração decorrida, e a média dos elementos da respectiva geração. Também podem ser

observadas as ocorrências de cada um dos tipos de erros identificados na produção da plataforma, tipos estes detalhados no decorrer da Seção 3.8.3.

Tabela 4 - Estrutura Base do Relatório de Execução

Geração	G1	G2	G3	G4	G5
Melhor Resultado	0.6496	0.6496	0.6496	0.6496	0.6496
Média dos Elementos	0.1445	0.1311	0.1245	0.1174	0.1228

Erros por geração	G1	G2	G3	G4	G5
Erro 1	6	10	5	3	4
Erro 2	0	0	0	0	0
Erro 3	0	0	0	0	0
Erro 4	0	0	1	0	0
Erro 5	0	0	0	0	0
Erro 6	0	0	0	0	0
Erro 7	16	14	19	22	17

População:	65	indivíduos

	G1	G2	G3	G4	G5
Erros de simulação críticos	22	24	25	25	21
Porcentagem de críticos	33.85%	36.92%	38.46%	38.46%	32.31%

Tempo de execução	10121.88223	Segundos

Fonte: O autor, 2019.

O relatório também apresenta um resumo de quantos desses erros foram erros críticos, ou seja, erros em que o indivíduo deixa de ser avaliado devido a determinada característica. Tal cálculo é baseado na quantidade de elementos, ou indivíduos, contidos a cada geração; informação também contida no relatório. O relatório também apresenta o tempo necessário para a realização com os respectivos parâmetros. Tal informação é utilizada na discussão da eficiência das evoluções nos estudos de casos contidos no Capítulo 5. A estrutura básica do relatório sofreu alterações para conter grandezas que serão apresentadas no decorrer deste trabalho, entretanto, sem alterar consideravelmente sua estrutura base apresentada acima.

2.8.3 Circuitos Com Problemas na Simulação e Identificação das Causas

Os erros identificados no decorrer do desenvolvimento da plataforma foram classificados em sete tipos distintos e identificados pela plataforma na ordem em que são apresentados. A identificação de cada um dos tipos apresentados a seguir foi feita a partir dos arquivos simulados (*netlists*) que acarretavam algum tipo de erro durante a simulação.

Para otimizar a busca por erros e, consequentemente, suas fontes, foi adicionada uma nova função, já citada anteriormente na Seção 3.3 sobre a localização de arquivos. Nesta função, cada circuito defeituoso, além de receber aptidão zero na evolução, é copiado para uma pasta específica com o nome de “Erros”. Essa pasta é criada junto com o diretório que conterá todos os arquivos resultado da simulação. Isso permite, junto com o histórico de erros da evolução, uma análise detalhada das fontes de erro para cada situação e o impacto na evolução. O diretório criado para armazenar os resultados da simulação e a pasta com os erros observados podem ser observados na Figura 19.

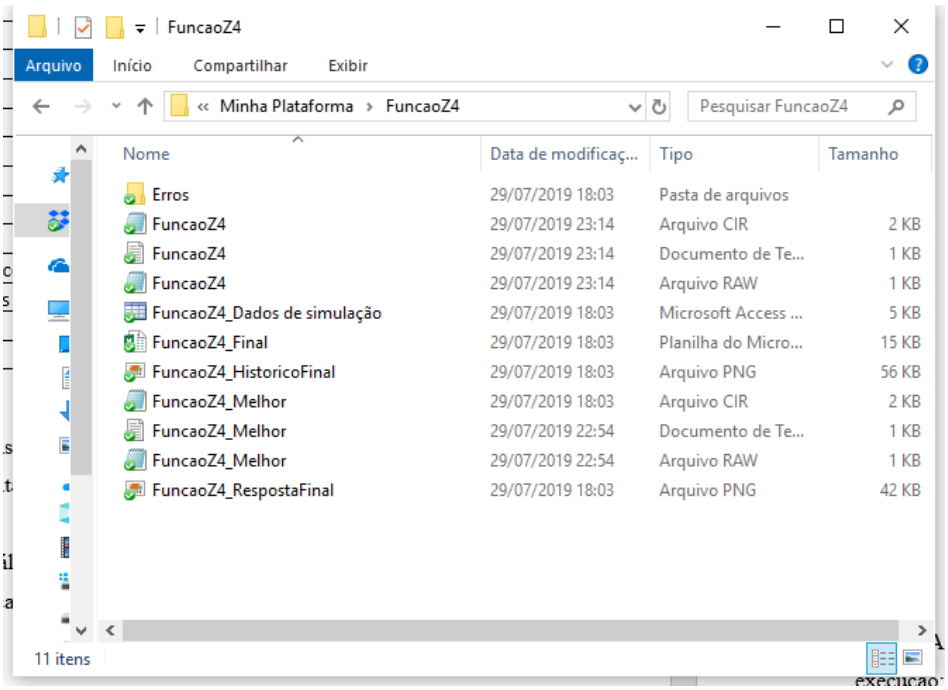

Figura 19 - Diretório Criado Para Cada Evolução Realizada

Fonte: O autor, 2019.

A partir da análise realizada, os possíveis erros durante uma evolução foram classificados em sete categorias. Essas categorias estão resumidas na Tabela 5 em conjunto com um resumo das respectivas causas.

Tabela 5 - Erros Observados Durante a Evolução

Erros Observados	Causa
1	Topologia definida de forma defeituosa
2	Falha de leitura no arquivo
3	Simulador gera um arquivo em branco
4	Saída indeterminada em algum ponto (NAN)
5	Topologia contém componentes flutuantes
6	Nó de saída definido incorretamente
7	Quantidade de tentativas de execução excedida

Fonte: O autor, 2019.

O primeiro caso de erro (Erro 1) ocorre em situações em que ocorre alguma transgressão nas conexões da topologia. As transgressões podem ser, por exemplo, o cruzamento ou curto de fontes de alimentação. Nesses casos, a chamada ao simulador, utilizando caminho do arquivo e do simulador, retornará o *flag* de erro igual a verdadeiro, ou 1, indicando uma violação na topologia avaliada.

O segundo erro consiste no erro de leitura da *netlist* pelo simulador de circuitos. Neste caso, o comando de avaliação é realizado de forma satisfatória ao indicar ao simulador o local da *netlist* a ser avaliada. Entretanto, devido a alguma rotina interna do sistema operacional ou a algum outro problema, o simulador não é capaz de abrir o arquivo a ser simulado. Nestes casos, a função retorna erro nulo durante a avaliação, entretanto, o simulador informa, através do arquivo log de simulação, a mensagem, em inglês, que o mesmo não pode ser aberto. Caso a avaliação continuasse normalmente sem o arquivo ser aberto, a saída avaliada, para o cálculo de aptidão, seria do último arquivo simulado satisfatoriamente e não atual que apresentou a falha, comprometendo a atribuição da aptidão.

O terceiro erro encontrado ocorre quando o arquivo é aberto pelo simulador de circuitos, entretanto, devido a falhas na avaliação o arquivo de saída é gerado em branco. Esse erro ocorre em casos em que o simulador é encerrado pelo sistema operacional antes que possa terminar a simulação.

O quarto erro observado ocorre quando a saída assume um valor indefinido em algum dos pontos amostrados. Esse erro ocorre em topologias em que a saída se torna oscilatória ou indeterminada em algum ponto. Nesses casos, o valor da saída retorna o acrônimo NAN que em inglês significa "*Not a number*", ou seja, não um número ou indefinido.

O quinto erro encontrado consiste na presença de componentes flutuantes na topologia. A existência de componentes com nós flutuantes acarreta valores flutuantes de tensão em determinados pontos do circuito. Nestes casos, o simulador realiza sucessivos métodos iterativos em busca da tensão em nós com valores flutuantes. Tais métodos consomem um elevado tempo para a realização dessas iterações. Caso o simulador não obtenha um valor determinado, o mesmo retorna um arquivo de resposta (“.RAW”) contendo cabeçalho com dados de simulação, mas nenhum ponto de amostragem para a saída.

O sexto, e penúltimo, erro ocorre quando o nó utilizado para verificar a saída não é utilizado pela topologia. Nestes casos, em que a *netlist* tenta avaliar um nó que não é utilizado, o simulador retorna o valor de todas as variáveis para os pontos de amostragem informado. Dessa forma, o simulador retorna uma quantidade maior de amostras que o esperado devido a observação de um nó não utilizado.

O último tipo de erro observado (Erro 7) consiste em uma redundância adicionada às correções a erros de operação. Este erro consiste em limitar o número de tentativas que o algoritmo tenta para abrir um arquivo que apresentou falha no processo de abertura.

O diretório, denominado “Erros”, pode ser observado na Figura 20. Nele, é possível observar o armazenamento de todos os erros ocorridos durante um processo de evolução. Os erros observados durante a evolução são então renomeados e salvos no interior da pasta, recebendo um nome correspondente ao erro observado e a data e hora em que ele ocorreu.

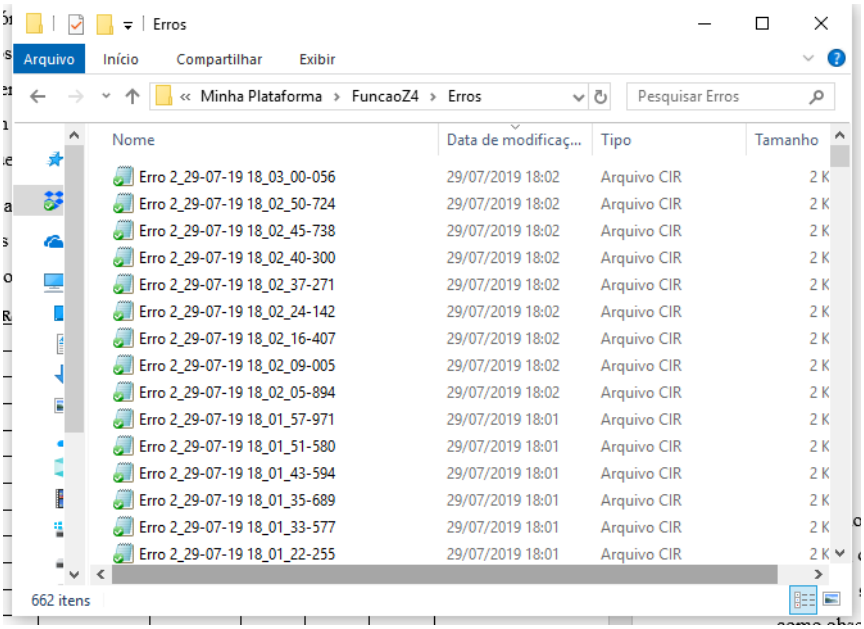

Figura 20 - Diretório Contendo Erros na Avaliação

Fonte: O autor, 2019.

A partir da classificação apresentada para os erros e o armazenamento das topologias responsáveis, uma avaliação mais detalhada das causas de cada erro foi possível. Essa compreensão das fontes causadoras destes erros resultou em uma estrutura de correção de falhas, que foi adicionada na plataforma desenvolvida.

2.8.4 Proposta Para Correção dos Problemas

A estrutura de correção de falhas adicionada na plataforma busca prevenir cada uma das situações problemáticas no decorrer da evolução. Tais prevenções foram adicionadas gradativamente na plataforma e, através do histórico de aparecimento de erros, consideradas efetivas no objetivo para que foram criadas.

A primeira correção adicionada a plataforma foi referente ao erro do Tipo 2. Nessa correção, a função de avaliação realiza novamente a chamada ao simulador para todo circuito em que a *netlist* não pôde ser avaliada. Essa verificação é realizada através do arquivo de log de execução. Caso seja verificado que o simulador teve dificuldade em abrir o arquivo a ser simulado, o algoritmo da plataforma realiza novamente o pedido de simulação. A realização de requisições recorrentes visa evitar a perda de topologias durante a evolução. Entretanto, não deve ser priorizada ao detrimento a execução da evolução como um todo.

Dessa forma, uma trava foi adicionada ao algoritmo com o objetivo de evitar que o algoritmo permaneça preso por muito tempo numa determinada topologia. Essa trava consiste em um número máximo de tentativas que o algoritmo deve tentar avaliar uma topologia que não foi lida pelo simulador. Essa quantidade máxima de tentativas é definida pelo usuário e, quando ultrapassada, caracteriza uma situação de erro do tipo sete. Logo, os arquivos apresentados como erro dois são caracterizados por não serem simulados em um primeiro momento, mas satisfatoriamente simulados após uma nova tentativa. Devido a isso, esse erro não é caracterizado como um erro que impeça sua simulação (erro crítico). Em contrapartida, erros do tipo sete são considerados erros críticos, uma vez que o circuito é ignorado após exceder o número máximo de tentativas.

Após verificada a efetiva simulação da topologia, o algoritmo verifica se o arquivo de saída contém dados a serem lidos. Tais situações, características do erro do tipo três, ocorrem em situações que o simulador é encerrado antes do fim do processo de simulação.

Uma vez comprovada a presença de informações no arquivo de saída, o algoritmo verifica se existe algum ponto de indefinição nas amostras avaliadas. Tal existência, característica do erro de tipo quatro, indicaria que instabilidade da topologia evoluída. Em conjunto com a verificação, métodos corretivos foram adicionados à análise do cromossomo. Essa metodologia de correção, apresentada na próxima seção (3.8.5), será utilizada na correção dos casos de erro do tipo quatro e seis, como apresentado posteriormente.

Após a busca de algum valor indefinido, o algoritmo verifica casos em que o simulador foi encerrado sem conseguir calcular um valor de tensão para os pontos especificados. Este comportamento, referente ao erro do tipo cinco, ocorre devido aos sucessivos métodos iterativos não alcançarem um valor para o cálculo dos pontos amostrados. Essa verificação identifica a ocorrência deste tipo de erro, mas não previne, sendo necessária uma prevenção na evolução de topologias.

A prevenção aos casos de sucessivos cálculos iterativos na determinação de valores é realizada através da definição de opções de simulação através da *netlist*. O conjunto de opções utilizadas, seus respectivos valores e seu significado podem ser observados na Tabela 6.

Tabela 6 - Opções de Simulação Do LTSpice Utilizadas

Opção Utilizada	Valor	Significado
noopiter	1	Inicia a avaliação pelo método de condutância(gmin stepping)
gminsteps	0	Define passo zero para o método de condutância (gmin stepping)
itl4	0	Limite de iterações para análise transitória
itl6	0	Define como zero para impedir que a fonte avance para a solução DC inicial.
ptrantau	0	Passo da análise pseudo-transitória
ptranmax	0.001	Tempo de análise pseudo-transitória

Fonte: O autor, 2019.

A última verificação realizada por código é referente ao sexto erro de simulação. Nele, a quantidade de amostras no retorno do simulador é maior do que a desejada. Esse erro ocorre nos casos em que a saída é identificada de forma errada e, por não identificar a variável solicitada, o simulador retorna todas as variáveis existentes na simulação. Apesar de facilmente

identificada, através da dimensão de amostras solicitada e a retornada, a resolução desse erro demanda uma especificação mais precisa das variáveis desejadas.

O processo completo de verificação de casos de falhas pode ser observado abaixo na Figura 21. Nela são apresentados a sequência de verificações realizadas pela plataforma durante o processo de cálculo da aptidão de cada topologia. Caso algum dos casos de erro seja identificado, o arquivo com o a topologia é armazenado na pasta de erros e recebe valor zero de aptidão na avaliação. Tal procedimento, entretanto, não se aplica aos casos de erro dois em que foi necessária uma nova requisição de simulação. Em tais casos, a contagem de ocorrências serve apenas para posterior avaliação da ocorrência da mesma.

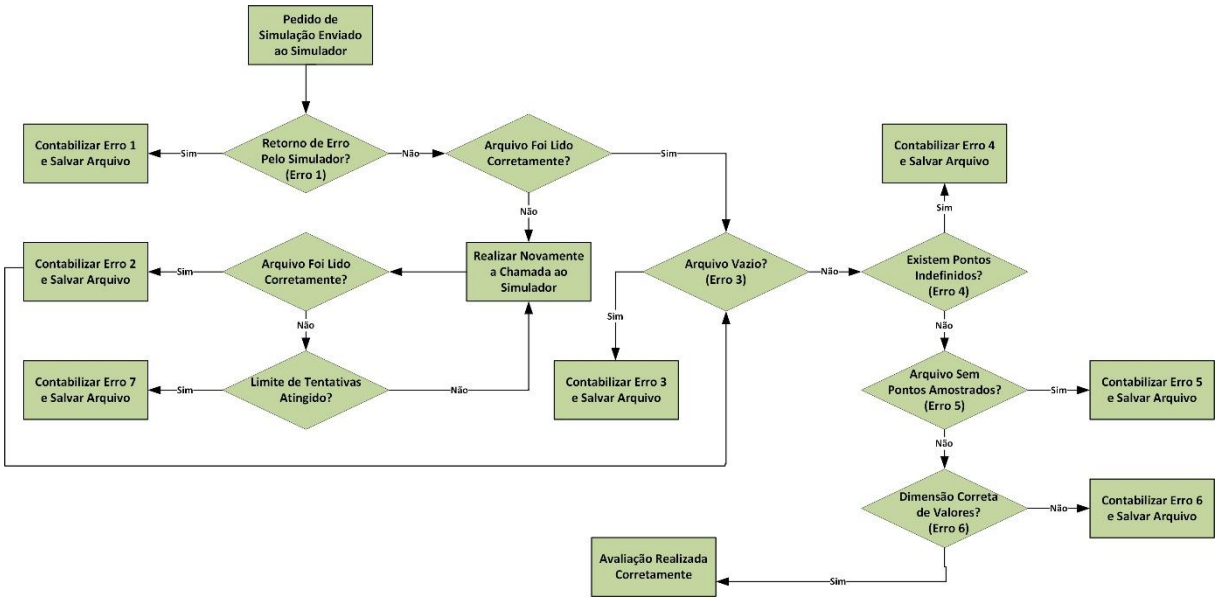

Figura 21 - Verificação de Erros Pelo Algoritmo

Fonte: O autor, 2019.

O processo de verificação da ocorrência de erros previne situações de travamento da plataforma ao trabalhar com repostas de simulação fora do esperado. Entretanto, tais procedimentos não foram suficientes para solucionar os casos de erro dos tipos um, quatro e seis. Para tais casos foi necessária uma utilização mais crítica do cromossomo, evitando topologias impróprias de circuitos. Tal especificação é realizada através de correções realizadas no cromossomo durante a simulação, retornando da função com seus valores atualizados.

2.8.5 Correção do Cromossomo Durante Execução

O processo de correção do cromossomo ocorre enquanto o arquivo da topologia a ser analisada é gerado. Esse processo é ativado e desativado pelo usuário através de uma variável (Correção) com o objetivo de verificar o impacto dessa função no processo de evolução. Dessa forma, durante o processo de escrita da netlist a ser simulada, a função de avaliação realiza correções no cromossomo para evitar conexões impróprias dos componentes.

Essas correções consistem na alteração dos nós, em casos de curto de componentes, e o impedimento que uma ou mais fontes sejam posicionadas de forma cruzada; como observado na Figura 22. Este tipo de posicionamento caracteriza um tipo de topologia imprópria e retorna erro um de simulação por parte do simulador.

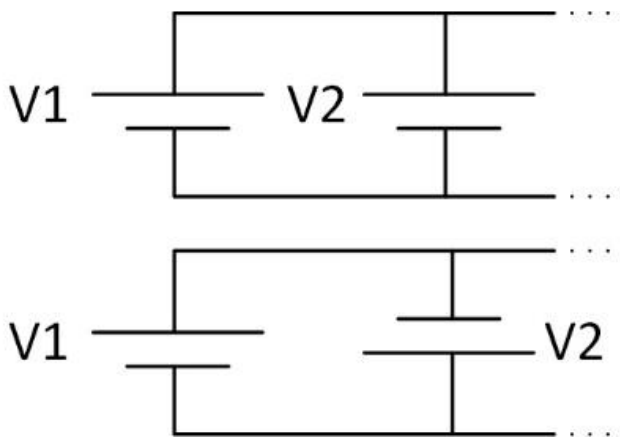

Figura 22 - Exemplo de Topologias Impróprias Corrigidas
Fonte: O autor, 2019.

Além da correção realizada na conexão de componentes, a função de avaliação realiza a correção do nó avaliado como saída do circuito. Durante a escrita da topologia a ser simulada, a função observa quais os nós possuem componentes conectados. Caso a saída seja requisitada em um nó não utilizado, a função de avaliação escolhe outro nó dentre os que estão sendo utilizados. Dessa forma, a correção realizada evita o aparecimento de erros do tipo 6 em que a variável requisitada não existe e a resposta apresenta dimensão diferente da esperada.

Através das simples alterações durante a avaliação do cromossomo, resumidas na Figura 23, o algoritmo foi capaz de eliminar o restante dos erros críticos e, consequentemente, otimizar o processo de evolução

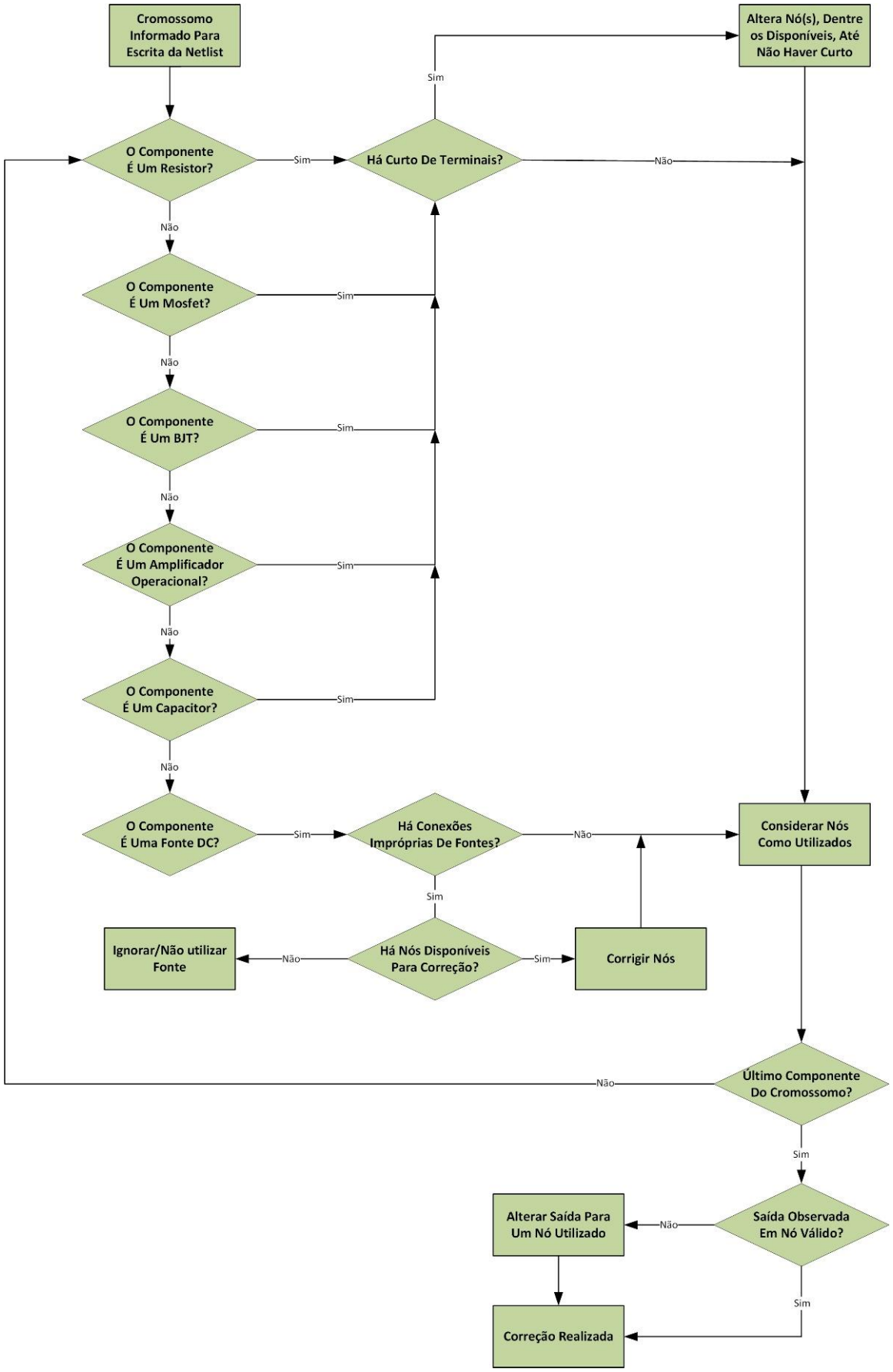

Figura 23 - Correção do Cromossomo Durante Avaliação

Fonte: O autor, 2019.

2.9 Exportação de Resultados

Após a execução da evolução, além de salvar a topologia que obteve melhor aptidão, o algoritmo apresenta o histórico do processo no decorrer das gerações. Este histórico é similar ao resultado parcial, apresentado durante seu funcionamento. Na tela, entretanto, é possível observar de forma resumida os parâmetros de simulação e a aptidão do melhor elemento encontrado; como observado abaixo na Figura 24.

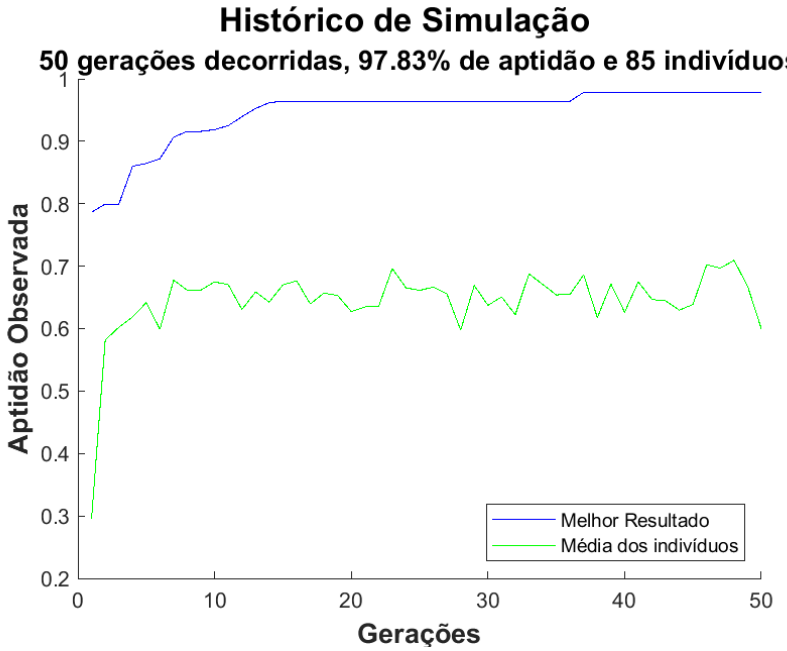

Figura 24 - Histórico de Simulação

Fonte: O autor, 2019.

Além do histórico de evolução, o algoritmo apresenta a saída apresentada pelo melhor elemento. Esta saída é apresentada em conjunto com a saída desejada informada no início da evolução. Dessa forma, o usuário é capaz de facilmente observar se o processo de evolução foi realizado de forma satisfatória.

A observação da saída é apresentada através de valores de tensão para os casos de evoluções que utilizem entradas DC, como observado na Figura 25. Nela também são apresentadas duas grandezas importantes para a avaliação da evolução, que são a quantidade de gerações necessárias e a aptidão observada após essas gerações.

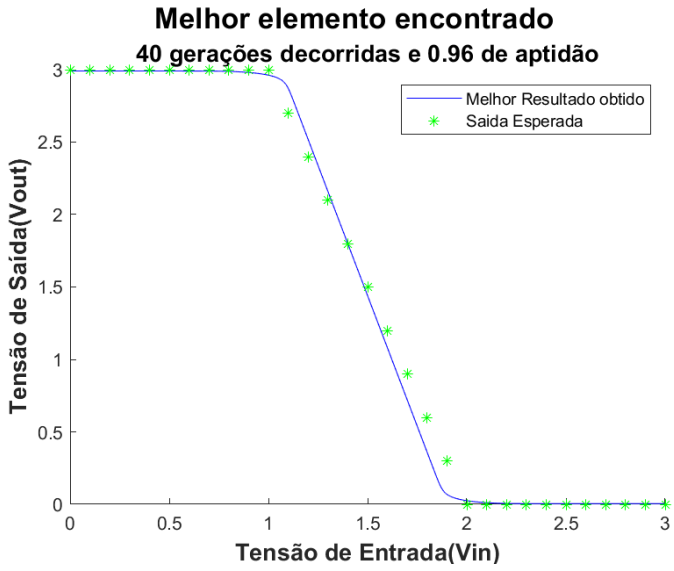

Figura 25 - Apresentação do Melhor Elemento Para Evoluções DC

Fonte: O autor, 2019.

Já em simulações no domínio da frequência, a observação da saída é apresentada pela amplitude no intervalo de frequências utilizado. Para os casos de evolução no domínio da frequência, além de informar a quantidade de gerações e a aptidão do melhor indivíduo, o algoritmo apresenta a variação da fase para o mesmo intervalo de frequência. Dessa forma, o usuário é capaz de avaliar ambas as respostas; apesar de a evolução ser realizada utilizando apenas o módulo da resposta. Tal apresentação pode ser observada na Figura 26.

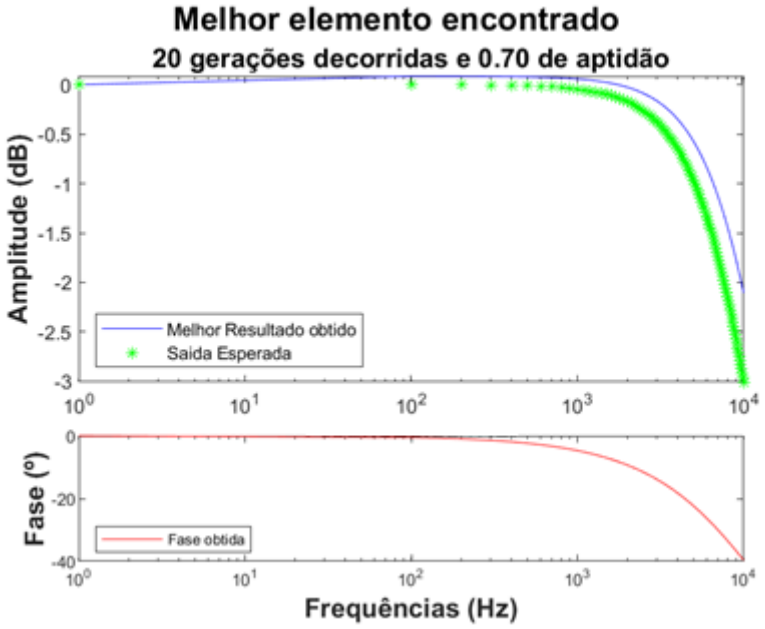

Figura 26 - Apresentação do Melhor Elemento Para Evoluções No Domínio da Frequência

Fonte: O autor, 2019.

Tais representações gráficas da resposta também são armazenadas, pelo algoritmo, automaticamente na pasta criada especificamente para a simulação. Isso permite uma fácil exportação do resultado da evolução realizada.

A exportação apresentada permite ao usuário o compartilhamento apenas do comportamento observado pela evolução, sem a possibilidade de trabalhar com os dados observados. Dessa forma, foi adicionada ao algoritmo a capacidade de salvar todas as variáveis de interesse da simulação. Tal funcionalidade realiza, ao final da evolução, o salvamento das variáveis contidas no MATLAB em um arquivo ".mat". Tal exportação de variáveis permite ao usuário a utilização dos dados de simulação sem a necessidade de realizar todo o processo de evolução novamente.

Outro arquivo exportado ao final do processo de evolução é o relatório de evolução apresentado na Seção 3.8.2. Nele, é possível observar detalhadamente os valores apresentados graficamente ao usuário ao final da execução. Tal arquivo é exportado no formato de uma planilha do Microsoft Excel e salvo na pasta referente a evolução realizada. Na Figura 27, é possível observar o conteúdo de uma dessas pastas geradas durante a evolução.

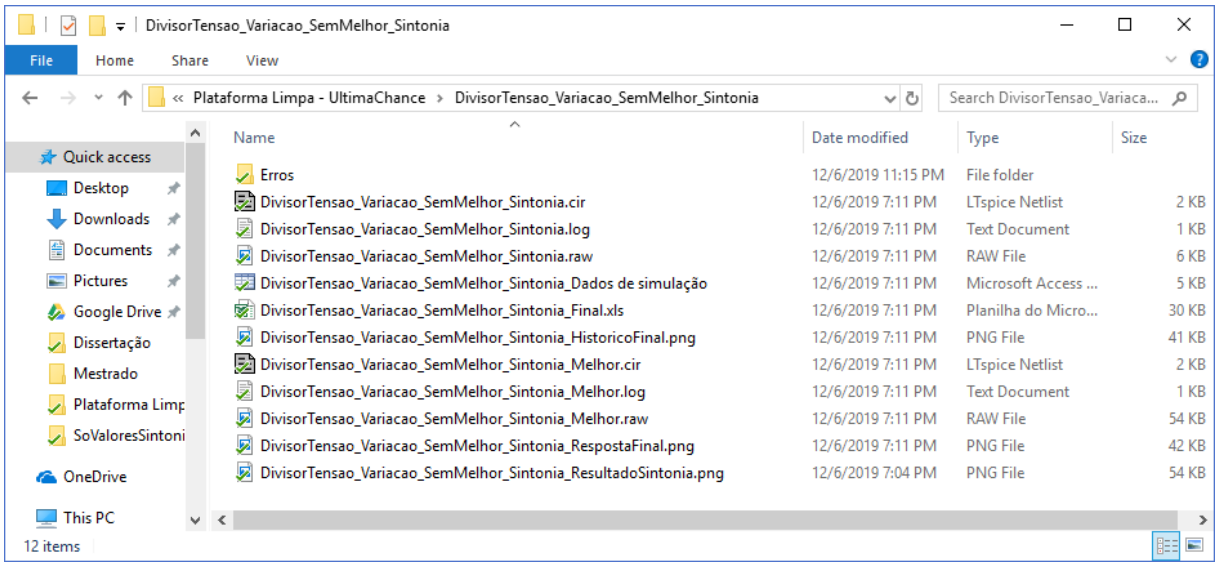

Figura 27 - Arquivos Exportados ao Final da Evolução

Fonte: O autor, 2019.

Nesta pasta é possível observar, além dos arquivos anteriormente apresentados, uma imagem denominada "ResultadoSintonia". Esse arquivo diz respeito a outra função necessária durante os estudos de caso realizados; a sintonia de variáveis para a evolução.

2.10 Sintonia de Evolução

Durante a realização de evoluções, foi observado que pequenas variações nos parâmetros de mutação e seleção resultavam significativamente na eficiência do processo de evolução. Para cada circuito evoluído, existia uma combinação que resultava em evoluções satisfatórias e outras, não muito distantes, que resultavam na estagnação da evolução; como observado na Figura 28.

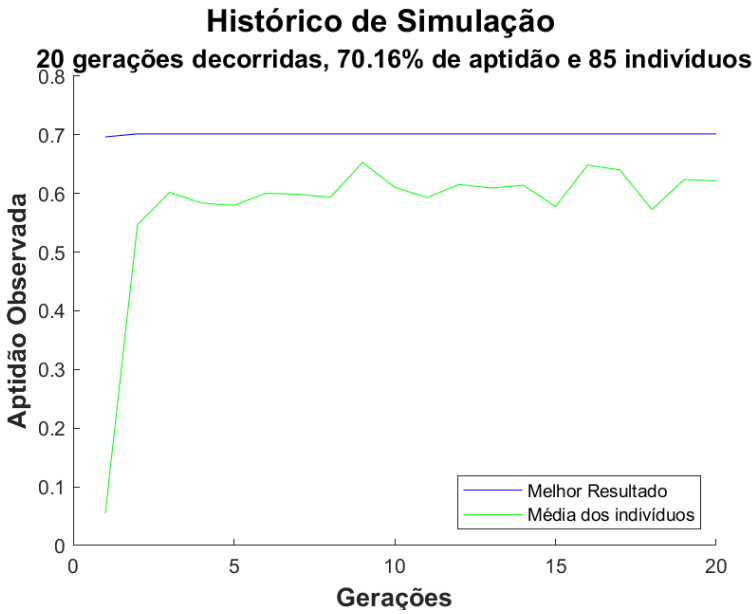

Figura 28 - Exemplo de Curva de Evolução Com Resposta Estagnada

Fonte: O autor, 2019.

Com o objetivo de encontrar a melhor combinação de valores para os parâmetros de mutação e seleção, foi adicionada na plataforma a capacidade de realizar uma sintonia de valores. Nesse processo de sintonia, apresentado abaixo na Figura 29, o usuário solicita o processo através de uma variável denominada “RealizarSintonia”.

O algoritmo, caso solicitado, realiza uma sintonia para determinar qual o valor dos parâmetros mutação e seleção possuem melhor combinação. Após definida a melhor combinação, o algoritmo utiliza tais valores para realizar a evolução do circuito analisado.

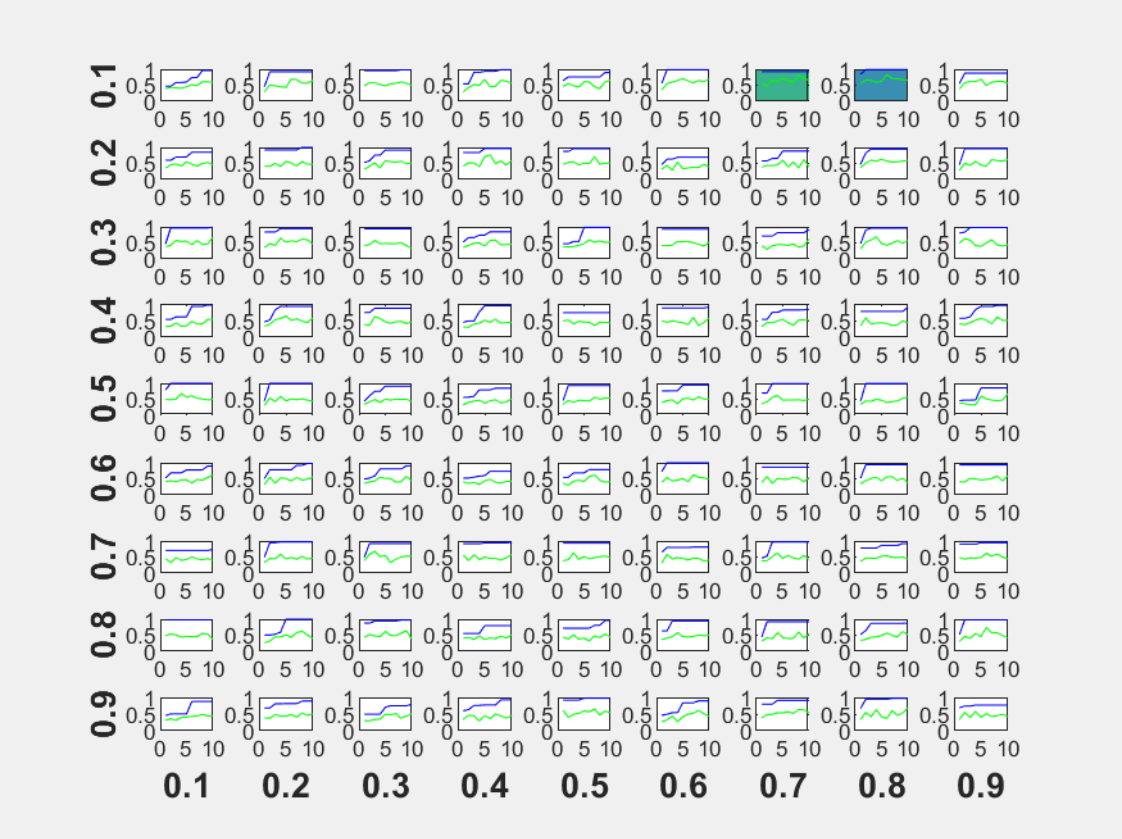

Figura 29 - Apresentação do Resultado da Sintonia de Valores Para Mutação (Horizontal) e Seleção (Vertical)
Fonte: O autor, 2019.

Como observado no diagrama anterior, o algoritmo realiza a simulação variando a mutação e a seleção de 0,1 a 0,9. Após realizar as combinações desses valores, ele verifica qual valor de média melhor progrediu no intervalo de teste especificado. Tal parâmetro foi definido por ter sido considerado que o sucesso da evolução é diretamente proporcional a média crescente da população durante o processo.

Após essa primeira sintonia, o algoritmo seleciona os dois melhores elementos avaliando a média da população durante as gerações de teste. O melhor elemento é destacado na cor azul e o segundo melhor é destacado na cor verde, como observado anteriormente na Figura 29. A partir dos valores identificados com a melhor variação, o algoritmo poderá realizar uma segunda sintonia com granulação dez vezes menor. Tal lógica de avaliação pode ser observada na Figura 30.

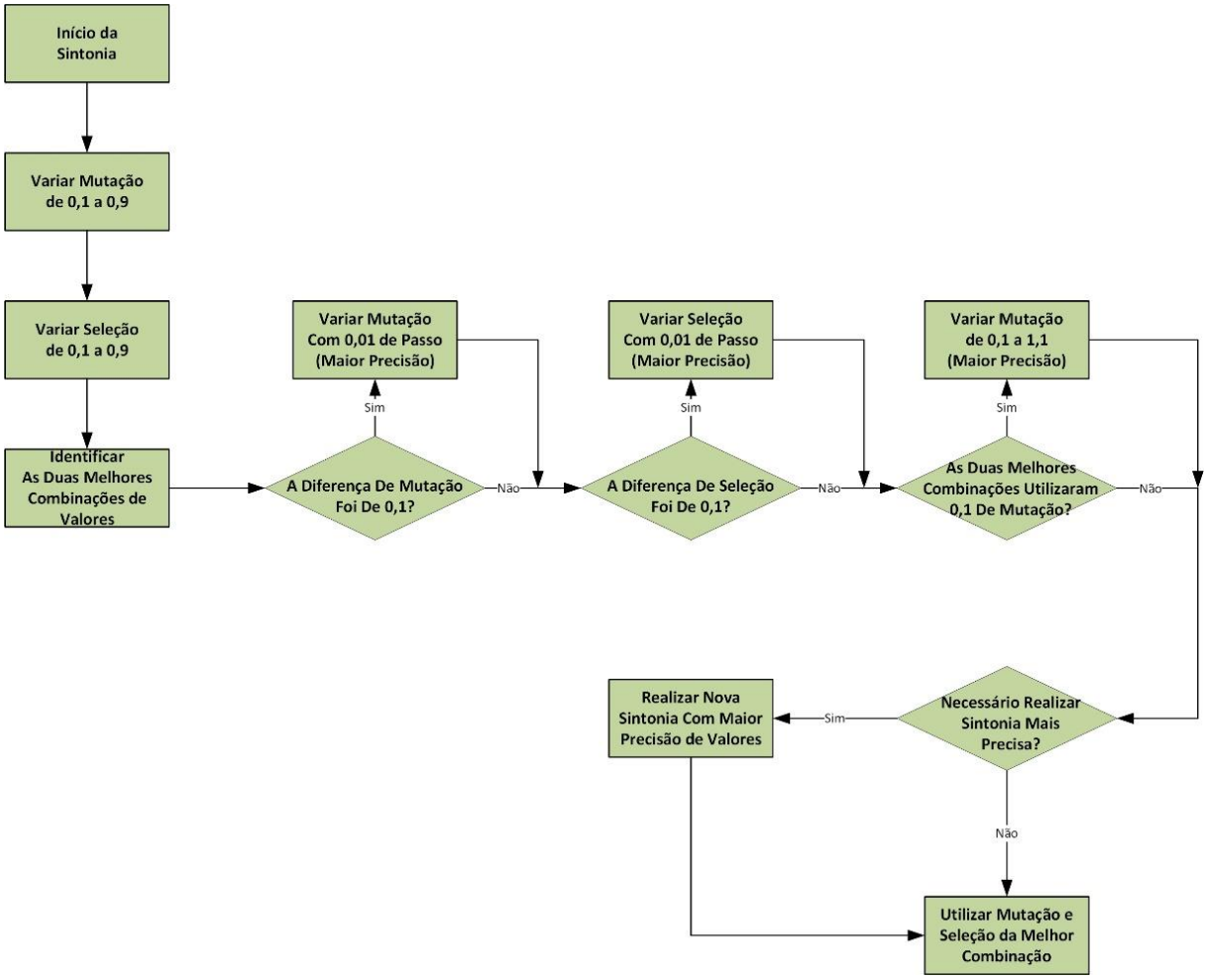

Figura 30 - Sintonia de Valores Para as Variáveis de Mutação e Seleção

Fonte: O autor, 2019.

Como observado no fluxograma anterior, o algoritmo constata se a distância dos valores utilizados pelo primeiro e segundo elementos é igual a granulação da primeira sintonia (0,1). Caso o intervalo seja igual a granulação, o algoritmo realiza uma segunda sintonia com granulação dez vezes menor (0,01) entre os valores utilizados.

Há também a possibilidade de os melhores elementos utilizarem a mesma granulação e esta ser igual a 0,1. Nestes casos, a granulação para a segunda sintonia variará de 0,1 a 1,1 em busca da melhor combinação de valores.

Este processo é apresentado em tempo real ao usuário através da tela de execução. Nessa apresentação pode ser observada a porcentagem concluída do processo e os valores de variação. Esse funcionamento pode ser observado na Figura 31.

Figura 31 - Execução do Processo de Sintonia

Fonte: O autor, 2019.

A funcionalidade de sintonia permite ao usuário encontrar a melhor configuração de variáveis para circuitos cujo comportamento de evolução é desconhecido. Isto facilita a evolução de circuitos pois isenta o usuário de solicitar a evolução dos circuitos para cada combinação dos parâmetros de mutação e seleção.

3 AVALIAÇÃO DE MÚLTIPLOS OBJETIVOS

Após constatar o funcionamento da plataforma na evolução de um objetivo, o algoritmo foi modificado para executar a evolução de dois objetivos. Neste modo de evolução, a aptidão de um circuito evoluído será determinada utilizando a saída observada e a potência consumida pelo circuito.

Essa avaliação e problemas cujo objetivo consiste na maximização e/ou minimização de variáveis são denominados Problemas de Otimização Multiobjetivo. Tais problemas usualmente possuem objetivos conflitantes, como por exemplo, o projeto de uma unidade de armazenamento em que se deseja maximizar a capacidade e minimizar o custo de fabricação (JONATHAN; PACHECO; 2010). Dessa forma, problemas como esse necessitam obter uma solução que não priorize a avaliação de um objetivo sobre os demais. Essa avaliação que busca a evolução sem que uma seja dominante sobre as demais pertence a uma linha de pesquisa denominada Otimização Multicritério.

A Otimização Multicritério possui grande variedade de procedimentos ao analisar a dominância das variáveis avaliadas. Uma classificação possível para os métodos utilizados os separam em quatro categorias:

- **Métodos Sem Preferência** – Não estabelecem uma preferência sobre os objetivos, bem como não utilizam informações prévias sobre as variáveis. Tais métodos buscam trabalhar na fronteira de soluções em que otimização de uma variável acarreta na piora sobre as demais variáveis.

- **Métodos A-Priori –** Usam informações prévias sobre as variáveis para determinar um critério de avaliação que pondere a importância de cada uma. Tal proporção poderá ser realizada através de médias ponderadas ou funções de pertinência, como será observado mais adiante. (BACELAR; 2017) (COELHO et. Al.; 2019)

- **Métodos A-Posteriori –** Permitem a obtenção de um conjunto de soluções ótimas que serão posteriormente escolhidas por um avaliador. Tal avaliação não necessita a especificação de restrições anteriores à evolução, mas demanda alto custo computacional.

- **Métodos Interativos –** Permitem o acréscimo ou alteração de pesos ou preferências no decorrer do processo de avaliação. O avaliador não necessita determinar os critérios utilizados antes do início da análise, podendo alterá-los no decorrer do processo. Tal metodologia permite a realização de uma busca guiada a soluções otimizadas, mas demanda acompanhamento e interferência constante do avaliador no processo de análise. (ZEBULUM; 2001)

Sendo o objetivo dessa plataforma uma análise independente de chamadas ao projetista, foi escolhida a metodologia a-priori. Desta forma, através de critérios previamente estabelecidos, a otimização será automaticamente avaliada e apresentada ao projetista de acordo com sua especificação. Para isso, foi escolhida a utilização da Lógica Fuzzy. Para um melhor entendimento de como implementar a Lógica Fuzzy na plataforma desenvolvida, um trabalho em conjunto foi realizado, trabalho este que resultou em um artigo (COELHO et. Al.; 2019).

3.1 Sistemas Fuzzy

A ideia básica na aplicação de um Sistema Fuzzy como avaliador é aproximar a análise de múltiplos objetivos à lógica de avaliação humana. Tal aproximação é realizada através da conversão de rótulos, ou conceitos qualitativos, a valores numéricos (SHUKLA; TRIPATHI; 2012). A interpretabilidade, natural de sistemas fuzzy, possibilita repassar preferência de modo simplificado ao sistema, utilizando uma linguagem compreensível e descomplicada.

O primeiro passo para a construção de um sistema fuzzy é a determinação das classificações possíveis para cada objetivo. Após determinados tais rótulos, estipula-se uma faixa dos valores aceitável para cada um. Por utilizar uma lógica de avaliação próxima ao raciocínio humano, essa delimitação permite a sobreposição de valores entre as categorias especificadas. Um exemplo da variável temperatura pode ser observado abaixo na Figura 32.

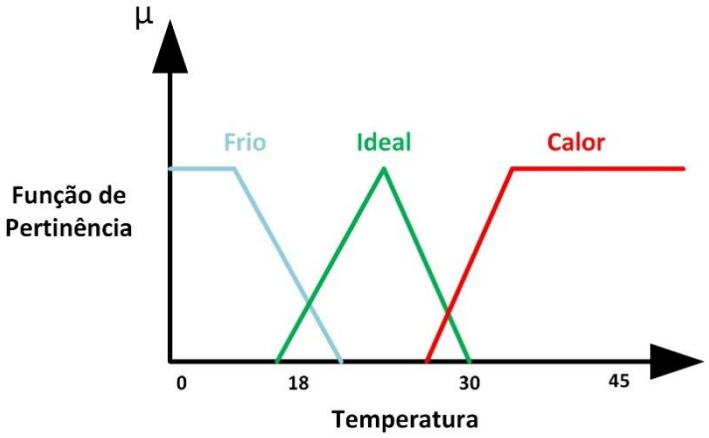

Figura 32 - Exemplo de Uma Variável Fuzzy

Fonte: O autor, 2019.

As curvas que demonstram a delimitação dos valores aos rótulos utilizados recebem o nome de Funções de Pertinência (TANSCHEIT; 2007). Dessa forma, cada objetivo deve ser categorizado em rótulos e cada rótulo terá uma curva de pertinência correspondente. Curvas de pertinência serão posteriormente utilizadas para quantificar o valor observado em um valor através de uma Função de Pertinência. Tais curvas de pertinência possuem formato triangular ou trapezoidal, formato esse definido pelo projetista.

De acordo com o valor do objetivo observado, ele receberá um grau de pertinência para cada classificação possível. Através do valor de pertinência observado, tal reposta será atribuída a uma das classificações existentes. Tais classificações serão utilizadas pela avaliação fuzzy no processo decisório, baseando-se em uma tabela de regras também especificada pelo usuário. A esse sistema de avaliação fuzzy é dado o nome de agregador fuzzy.

O sistema agregador fuzzy utilizado é do tipo Mamdani. Tal sistema é caracterizado por ser simplificado e não atribui graus de importância distintos para suas regras. Dessa forma, o processo decisório é mais interpretável, baseando-se diretamente nas regras que relacionam as categorias de cada variável com as classificações existentes para a saída. Uma tabela de correlação das classificações das entradas e da saída e o processo de criação de regras correspondentes serão apresentados adiante no trabalho.

Um sistema agregador fuzzy também foi utilizado em avaliações multiobjetivos nos trabalhos (BACELAR; 2017) e (FONSECA; 2017), (COELHO et. Al.; 2019). Este trabalho tem como foco a incorporação de uma avaliação multiobjetivo na plataforma já desenvolvida. Serão utilizadas funções de pertinência e regras de correlação genéricas para verificar o funcionamento básico da plataforma. Diversos trabalhos na área avaliam o melhor formato de

funções e conjuntos de regras para avaliações multiobjetivo (JIANG; YANG; 2016) (GACTO; ALCALÁ; HERRERA;2011) (LUCA et. Al.; 2015). Deseja-se conceber uma plataforma que possibilite a trabalhos futuros verificarem os melhores critérios a serem utilizados.

Uma vez definido o sistema de inferência fuzzy a ser utilizado, realiza-se a incorporação desse sistema na plataforma já criada. Tal incorporação deve ser realizada de modo que sua utilização e modificação devem ser facilmente customizáveis pelo usuário.

3.2 Incorporação da Técnica Multiobjetivo no Ambiente Evolutivo

A incorporação da avaliação Fuzzy ao algoritmo foi realizada e modificada diretamente pela ferramenta Fuzzy presente no MATLAB. Através desta ferramenta é possível a rápida criação das regras de um Sistema Fuzzy sem utilizar linhas extensas de códigos.

Após executada a função “fuzzy” do MATLAB, o programa retorna uma tela de criação para a definição das regras que serão utilizadas. Esta tela, observada na Figura 33, permite a determinação das variáveis de entrada utilizadas e as variáveis cuja saída será calculada utilizando a lógica Fuzzy.

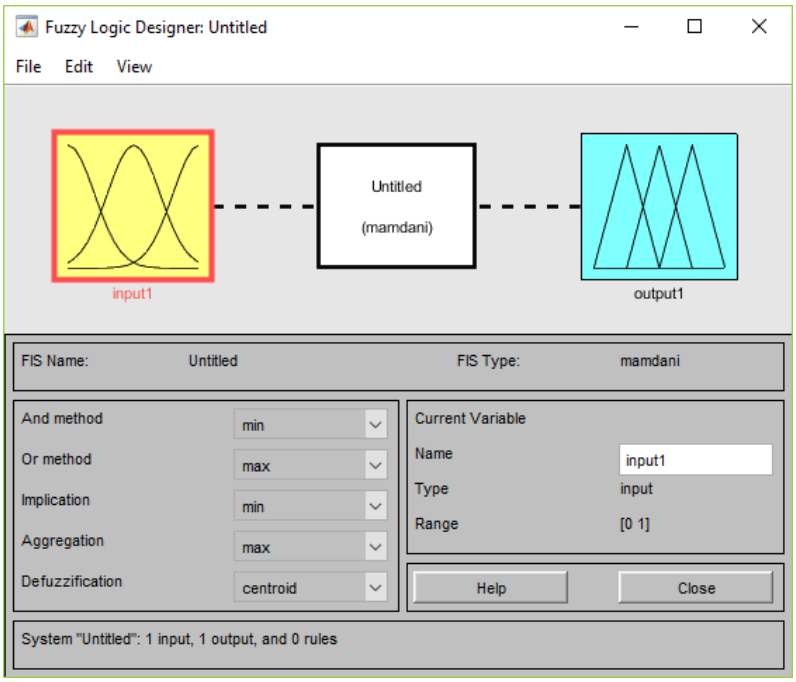

Figura 33 - Tela de Criação da Ferramenta Fuzzy

Fonte: O autor, 2019.

Com a utilização da ferramenta de criação, as variáveis utilizadas pela plataforma são geradas. Tais variáveis, mostradas na Figura 34, correspondem à compatibilidade com a saída desejada e à potência consumida pela topologia analisada.

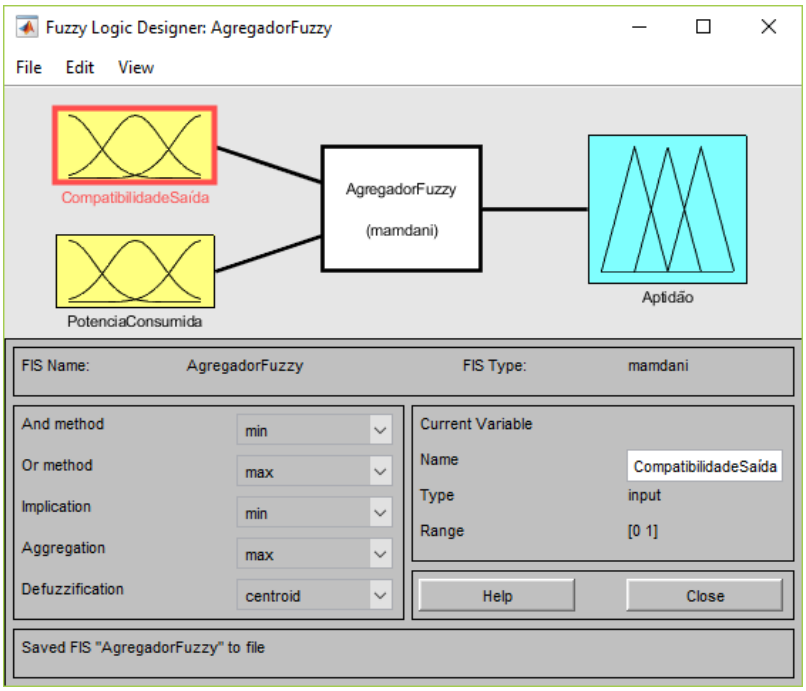

Figura 34 - Variáveis Utilizadas na Avaliação Fuzzy

Fonte: O autor, 2019.

A compatibilidade com a saída corresponde à aptidão do circuito para casos de um único objetivo. Esta compatibilidade, assim como nos casos mencionados anteriormente, assumirá valores entre zero a um e corresponde a uma aproximação de zero a cem por cento da saída almejada pela evolução.

Para a variável compatibilidade foram definidos três rótulos de classificação. Tais rótulos utilizam o valor da comparação com a saída desejada para determinar se a resposta observada obteve uma aproximação baixa, média ou alta.

No caso da variável que analisa a potência consumida, a mesma utilizará a solução conhecida para o circuito. Do mesmo modo que a variável compatibilidade, a potência consumida será atribuída a um consumo baixo, médio ou alto. Essa atribuição dos rótulos utilizará a solução conhecida para a saída.

Com base na solução conhecida pelo circuito, o algoritmo realiza uma chamada para a função de avaliação e quantifica a potência consumida por essa topologia conhecida. Em seguida, a função de avaliação verifica a potência consumida pela topologia avaliada durante a evolução. Através deste valor é realizada uma comparação com a potência conhecida. Essa

comparação determinará o grau de pertinência da resposta às classificações de consumo para os circuitos. As curvas de pertinência das variáveis analisadas podem ser observadas na Figura 35.

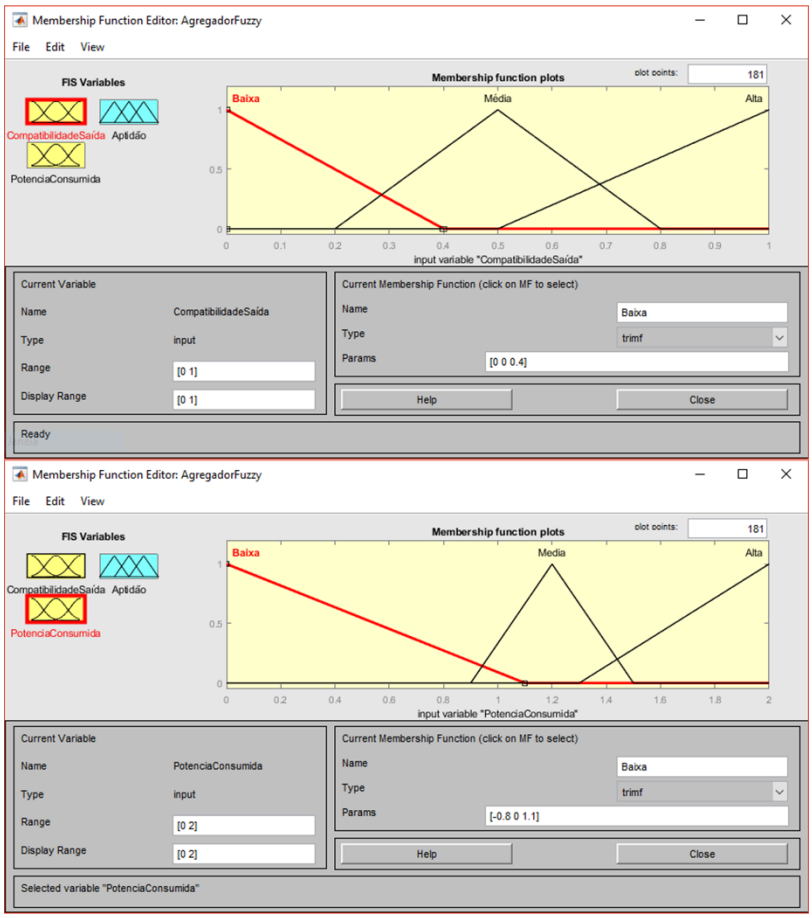

Figura 35 -Funções de Pertinência Utilizadas Pela Plataforma

Fonte: O autor, 2019.

Ressalta-se que as funções de pertinência, utilizadas no processo de incorporação e nos estudos de casos apresentados, foram determinadas de modo heurístico. Tais funções tem como objetivo de apresentar o processo de incorporação e verificar o funcionamento da plataforma. Um estudo mais detalhado das funções de pertinência e regras utilizadas compõe as avaliações necessárias às evoluções futuras realizadas pela plataforma.

Uma vez determinados os rótulos para as variáveis observadas, determinam-se as regras atribuídas às combinações destes rótulos. Os conjuntos utilizados na evolução realizada pela plataforma podem ser observadas na Tabela 7.

Tabela 7 – Matriz de Regras da Aptidão Utilizando a Compatibilidade e a Potência Consumida

	Potência Baixa	**Potência Média**	**Potência Alta**
Compatibilidade Baixa	Baixa	Baixa	Péssima
Compatibilidade Média	Média	Média	Baixa
Compatibilidade Alta	Ótima	Alta	Alta

Fonte: O autor, 2019.

Na tabela anterior, a aptidão é dividida em cinco rótulos: Péssima, Baixa, Média, Alta e Ótima. Tais rótulos são utilizados para compor as curvas de pertinência da saída. Tais curvas, observadas na Figura 36 são responsáveis por, avaliando a pertinência da resposta observada, quantificar a aptidão do circuito avaliado.

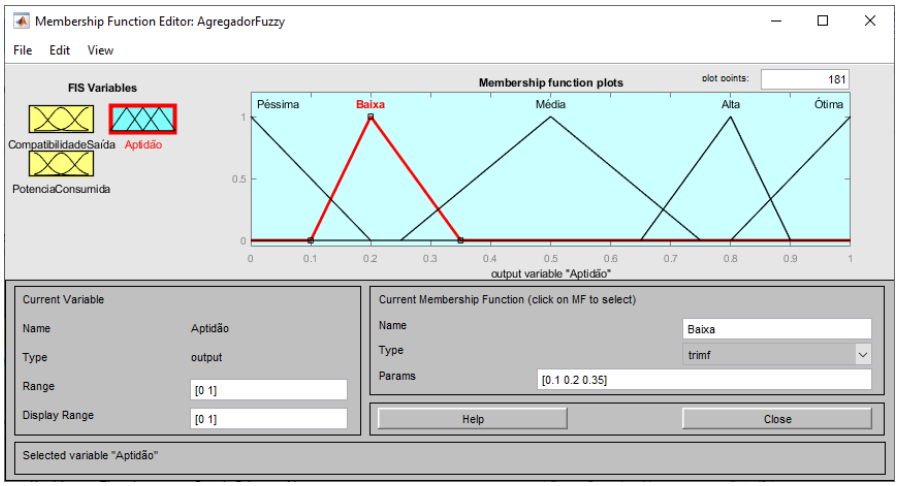

Figura 36 - Funções de Pertinência Utilizadas Para a Aptidão

Fonte: O autor, 2019.

Uma vez determinadas as funções de pertinência para as entradas e para as saídas, tais classificações devem ser relacionadas. Para realizar essa correlação, as regras anteriormente apresentadas são adicionadas ao sistema fuzzy, como mostrado na Figura 37.

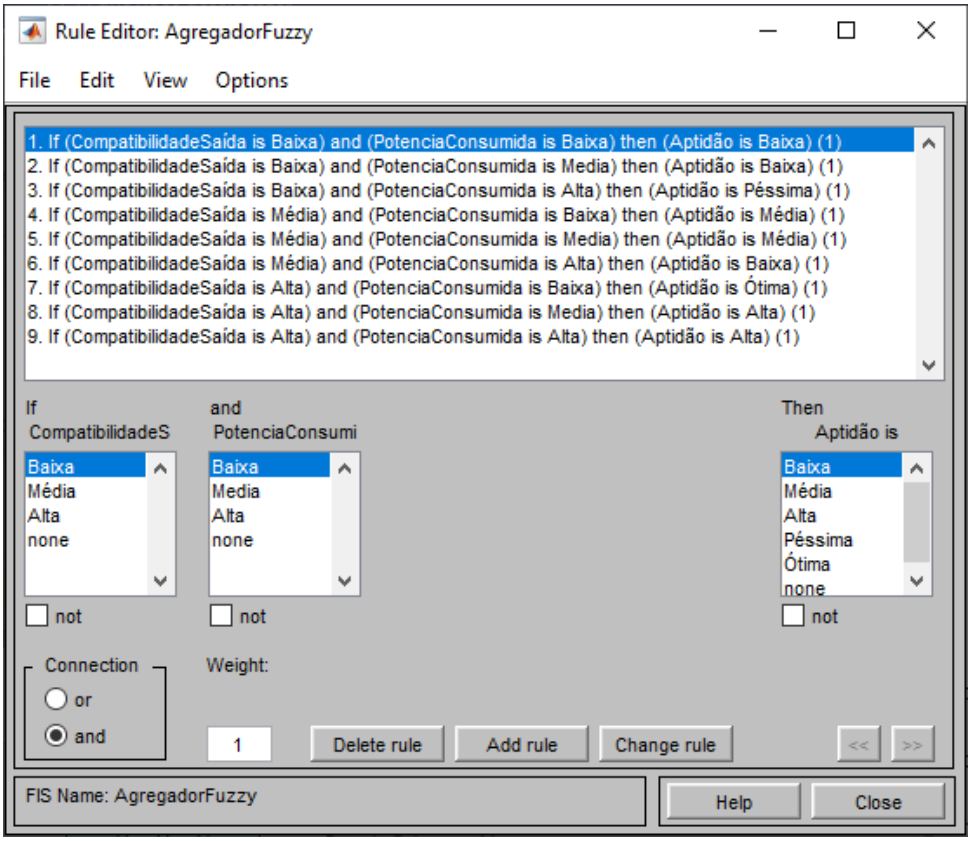

Figura 37 - Criação das Regras Fuzzy Utilizando Ferramenta Fuzzy do Matlab

Fonte: O autor, 2019.

Uma vez adicionadas à ferramenta Fuzzy do MATLAB, as curvas de pertinência podem ser modificadas com o objetivo de apresentarem a melhor atribuição de aptidão para os circuitos avaliados. Essa verificação pode ser realizada durante o processo de evolução ou, de modo mais simples, pelo próprio editor de regras Fuzzy.

As curvas e regras apresentadas anteriormente tem como objetivo verificar o funcionamento básico da plataforma e apresentar as ferramentas disponíveis para o usuário. Para trabalhos futuros, uma determinação mais criteriosa de tais parâmetros deverá ser realizada ao se avaliar o impacto de cada modificação.

Através da opção ”Visualizar Regras” é possível alterar o valor das variáveis utilizadas e observar sua pertinência e, consequentemente, a aptidão ao final da avaliação. Esta apresentação de resultados, observada na Figura 38, possibilita uma rápida verificação do agregador utilizado com as definições realizadas. Tal verificação permite, também, verificar o impacto na resposta final de eventuais alterações realizadas.

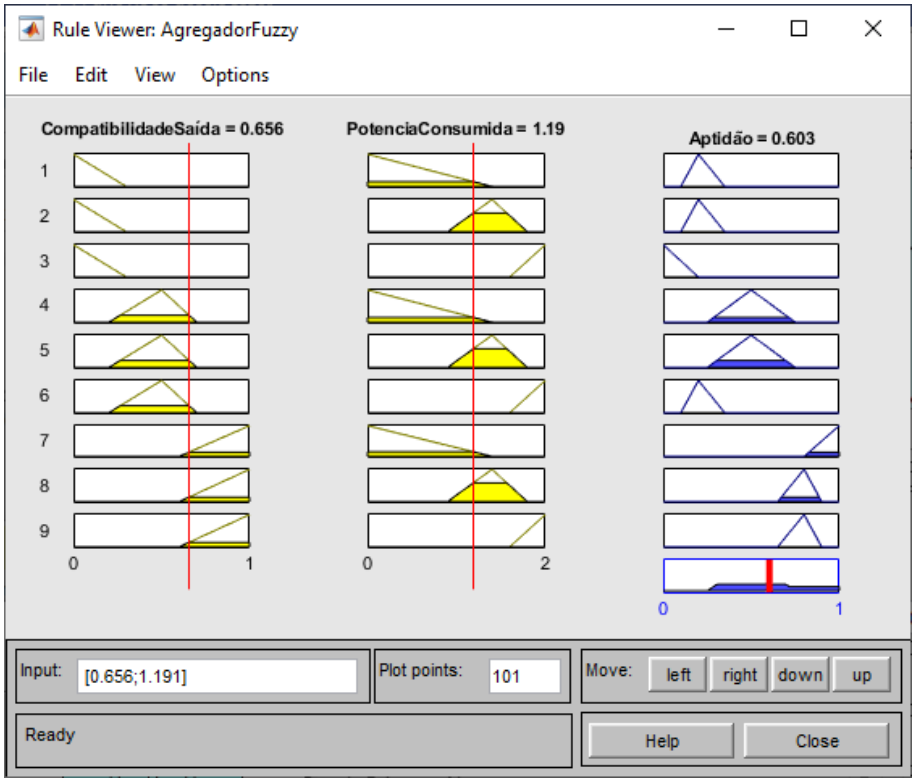

Figura 38 - Visualização Do Impacto das Curvas Utilizadas no Valor da Aptidão

Fonte: O autor, 2019.

Após definidos todos os parâmetros de análise Fuzzy, o conjunto de regras é exportado e salvo no mesmo diretório que contém as funções do MATLAB. Tais regras permanecerão externas à plataforma até o momento de serem necessárias ao processo de evolução.

3.3 Aplicação no Ambiente Evolutivo

Uma vez definido o conjunto de regras utilizadas, elas serão importadas para a plataforma durante seu funcionamento. A importação é realizada através da função “Readfis” do MATLAB. Esta função é responsável por importar o conjunto de regras contido no arquivo do tipo “.fis” e disponibilizá-las na plataforma.

Uma vez contidas na plataforma, a função “Evalfis” é utilizada para calcular a aptidão. Tal cálculo será realizado utilizando os valores das variáveis de compatibilidade e potência como observado pela Equação 7.

$$Aptidão = Evalfis([Compatibilidade\ Potência], Regras),$$
$$Regras = Readfis('Arquivo.fis') \quad (7)$$

Onde:
Aptidão = Valor de aptidão dado às topologias
Compatibilidade = Compatibilidade da saída observada comparada à saída desejada
Potência = Comparação da potência consumida pela topologia e a potência conhecida
Arquivo.fis = Arquivos contendo os parâmetros Fuzzy definidos na Seção anterior.

Esse modo de utilização dos parâmetros do agregador fuzzy facilitam a atualização da plataforma. Como tais parâmetros são importados pela função de avaliação no momento da avaliação e não definidos de forma fixa no código, o usuário pode atualizar as curvas de pertinência sem alterar o código da plataforma.

Dessa forma, qualquer atualização desejada no modo de avaliação do sistema fuzzy pode ser realizada diretamente no arquivo que contêm as regras. Isso permite uma atualização rápida e baseada no circuito a ser evoluído pelo usuário. Essa capacidade possibilita uma maior versatilidade da plataforma ao adequar sua avaliação aos parâmetros dados pelo usuário.

4 ESTUDOS DE CASO

Uma vez definidos os parâmetros e as funções utilizadas no desenvolvimento da plataforma, foram realizados estudos de casos para verificar seu funcionamento. Tal análise será realizada para cada um dos modos de operação que compõe a plataforma e avaliada de acordo com a complexidade dos circuitos utilizados. Para todos os estudos de casos a taxa de crossover utilizada foi de 85%.

4.1 Divisor de Tensão

O primeiro circuito avaliado pela plataforma foi um circuito divisor de tensão básico. Este circuito, observado na Figura 39, será utilizado para confirmar o funcionamento da plataforma e verificar o impacto de determinadas variáveis no processo de evolução.

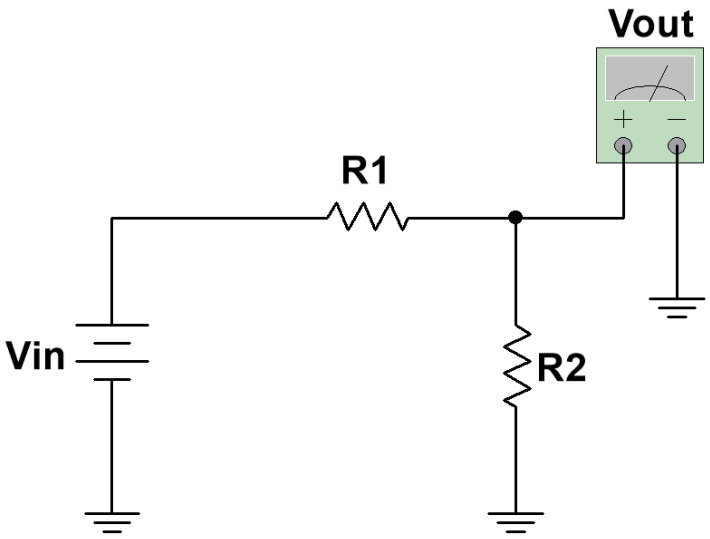

Figura 39 - Circuito Divisor de Tensão

Fonte: O autor, 2019.

Essa avaliação será realizada para os três modos que compõe a plataforma desenvolvida. Uma vez realizada, é possível aumentar a complexidade dos circuitos evoluídos e verificar o comportamento da evolução realizada pela plataforma.

4.1.1 Evolução por Variação de Valores

A primeira evolução foi realizada no modo variação de valores de componentes. Utilizando a equação do espaço de busca, definida na Seção 3.6.1, determina-se um espaço de busca igual a 14.403 variações. Nesta primeira evolução, foram atribuídos valores fixos para os parâmetros relacionados com seleção e mutação e observada a evolução dos indivíduos com o passar das gerações. Os parâmetros utilizados foram 20 gerações com 25 indivíduos. A taxa de crossover foi fixada em 0,85, igual a das demais evoluções, e a taxa de seleção por normalização geométrica e a taxa de mutação são fixas. Tal evolução pode ser observada na Figura 40.

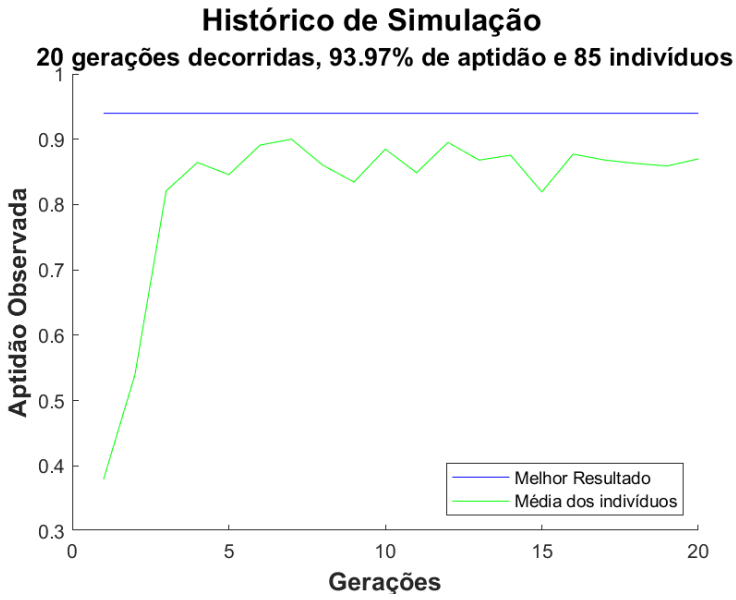

Figura 40 - Evolução Utilizando Mutação e Seleção Fixos

Fonte: O autor, 2019.

Novas simulações foram realizadas para avaliar a seleção de parâmetros. Tal processo foi realizado utilizando-se dois pares de valores para “Gerações” e “Indivíduos” utilizados durante a sintonia. O uso de diferentes pares destas grandezas visa determinar a melhor configuração para se realizar a sintonia de valores e, posteriormente, verificar o tempo necessário pelo processo de sintonia para obter uma boa combinação dos valores de seleção e mutação. Os pares de valores utilizados durante a sintonia e os respectivos desempenhos podem ser observados na Tabela 8.

Tabela 8 - Resultados Observados Após Sintonia de Valores

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,3	0,11	100%
7	5	0,8	0,09	100%

Fonte: O autor, 2019.

Os dois pares de resultados, observados na Figura 41 e na Figura 42, obtiveram 100% de desempenho para a evolução de valores. Ou seja, ambos obtiveram valores de componentes comerciais que resultaram em saídas totalmente compatíveis com a resposta almejada pela evolução.

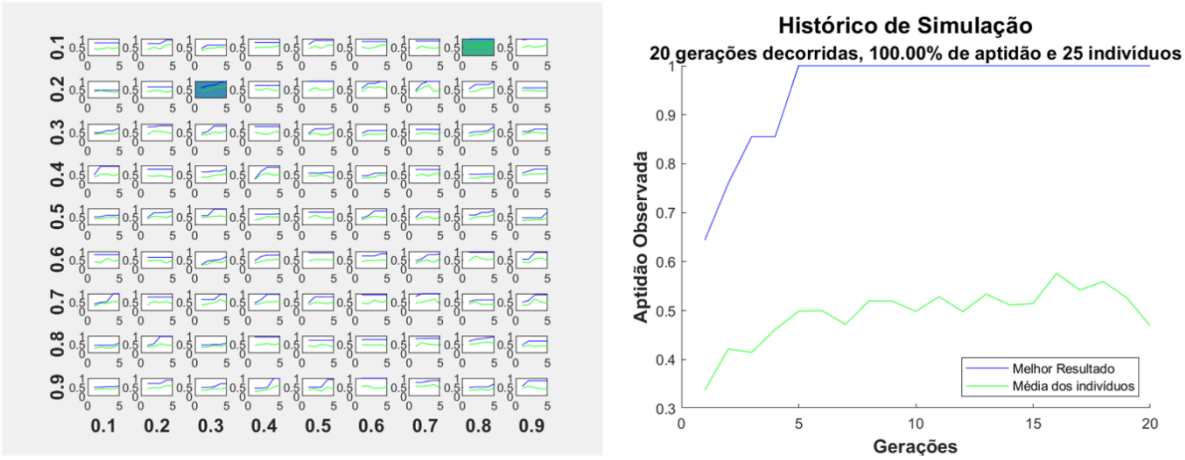

Figura 41 - Evolução Observada Após Sintonia Utilizando 5 Gerações de 5 Indivíduos

Fonte: O autor, 2019.

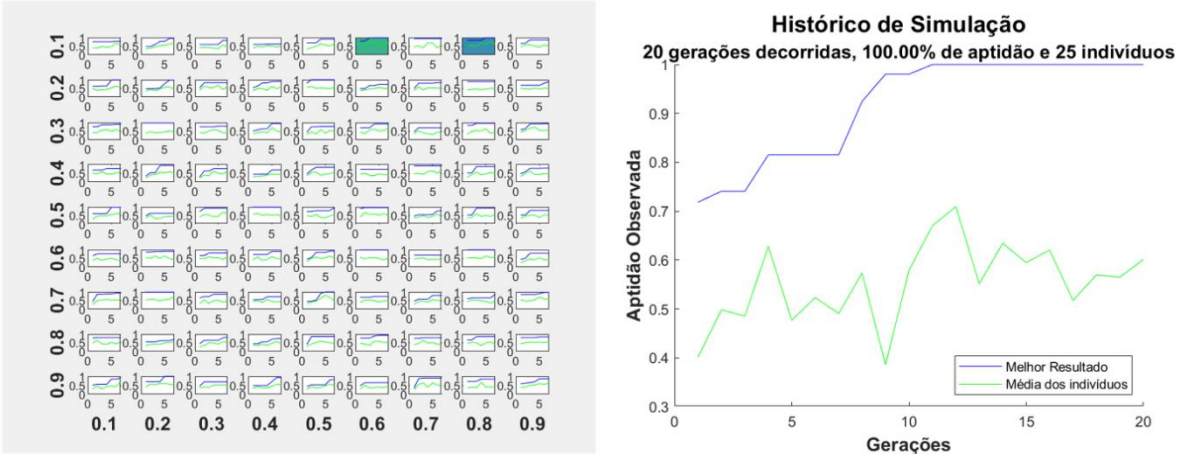

Figura 42 - Evolução Observada Após Sintonia Utilizando 7 Gerações de 5 Indivíduos

Fonte: O autor, 2019.

A evolução realizada para cada um dos casos resultou em duas topologias possíveis para o circuito desejado. A primeira topologia, obtida utilizando a combinação de cinco gerações de cinco indivíduos para a sintonia de valores, pode ser observada na Figura 43; a resposta apresentada pode ser observada na Figura 44.

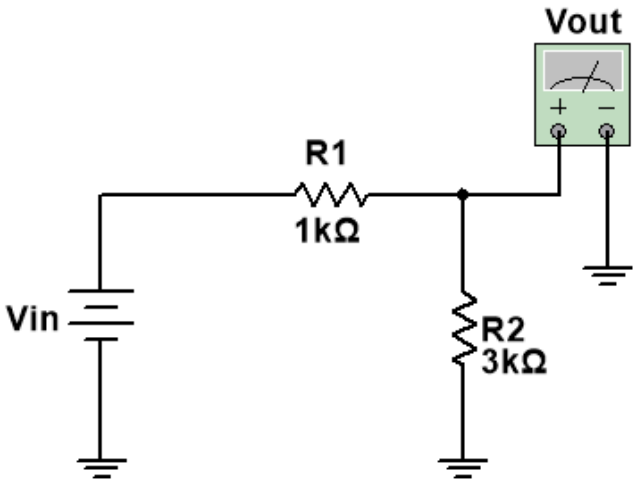

Figura 43 - Topologia Obtida Utilizando 5 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

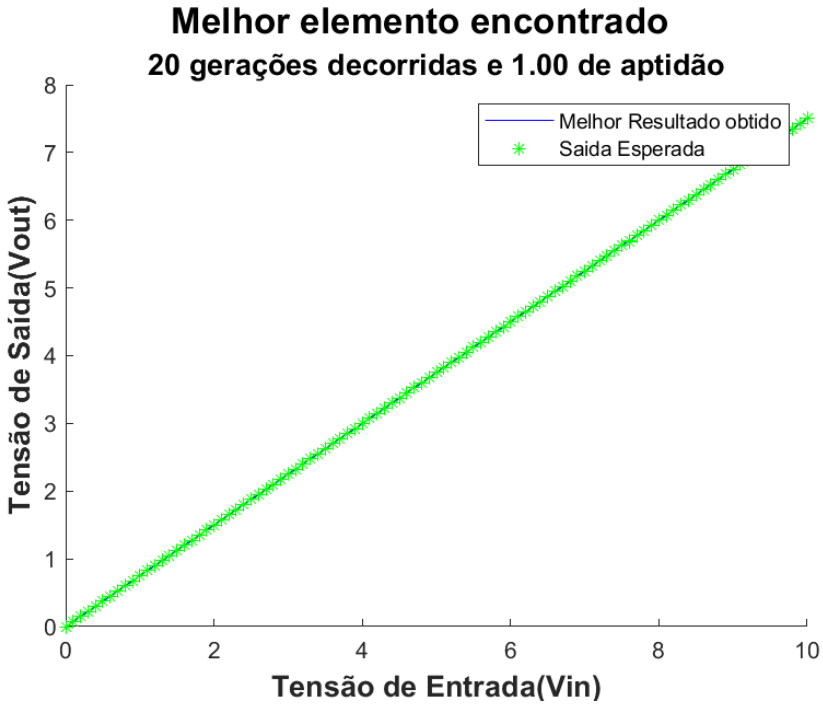

Figura 44 - Resposta Observada Utilizando 5 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

Ao final do processo de evolução, após a apresentação das curvas de evolução e da resposta observada, o algoritmo gerou um relatório de execução com os dados pertinentes ao processo. Tal relatório, armazenado na pasta com os demais arquivos gerados, pode ser observado abaixo na Tabela 9.

Tabela 9 - Relatório de Evolução Utilizando 5 Gerações de 5 Indivíduos na Sintonia

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Melhor Resultado	0.6442	0.7601	0.8554	0.8554	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Média dos Elementos	0.3375	0.4216	0.4147	0.4615	0.4990	0.4999	0.4719	0.5199	0.5194	0.4985	0.5285	0.4979	0.5338	0.5117	0.5144	0.5759	0.5421	0.5593	0.5261	0.4688
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erro 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
População:	25	indivíduos																		
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erros de Simulação Críticos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	1348,359367	Segundos																		
	22,47265612	Minutos																		
	0,374544269	Horas																		
Tempo de Evolução	265,5771563	Segundos																		
	4,426285938	Minutos																		
	0,073771432	Horas																		

Fonte: O autor, 2019.

A segunda topologia, obtida utilizando a combinação de sete gerações de cinco indivíduos para a sintonia de valores, pode ser vista na Figura 45. A saída dessa topologia pode ser observada na Figura 46.

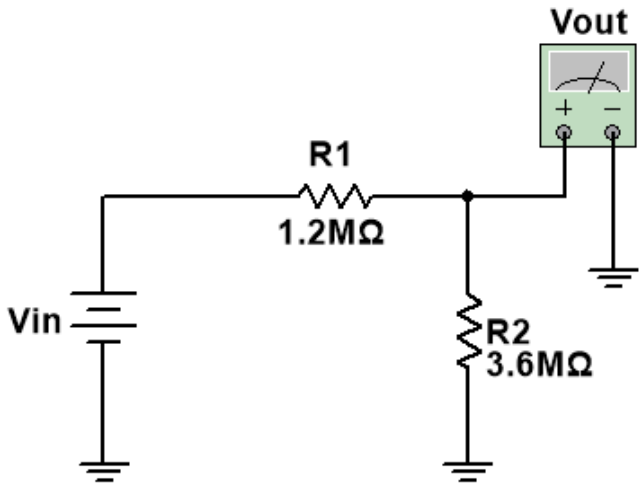

Figura 45 - Topologia Obtida Utilizando 7 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

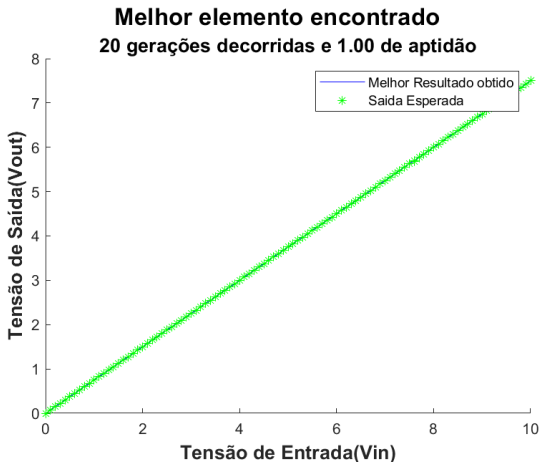

Figura 46 - Resposta Observada Utilizando 7 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

Ao final do processo de evolução, após a apresentação das curvas de evolução e da resposta observada, o algoritmo gerou um relatório de execução com os dados pertinentes ao processo. Tal relatório, salvo na pasta com os demais arquivos gerados, pode ser observado abaixo na Tabela 10.

Tabela 10 - Relatório de Evolução Utilizando 7 Gerações de 5 Indivíduos na Sintonia

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Melhor Resultado	0.7185	0.7408	0.7408	0.8153	0.8153	0.8153	0.8153	0.9247	0.9804	0.9804	1	1	1	1	1	1	1	1	1	1
Média dos Elementos	0.4013	0.4983	0.4853	0.6279	0.4769	0.5232	0.4907	0.5735	0.3860	0.5790	0.6703	0.7092	0.5515	0.6343	0.5950	0.6204	0.5174	0.5697	0.5650	0.6011
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erro 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
População:	25	indivíduos																		
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erros de Simulação Críticos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	1950,144436	Segundos																		
	32,50240727	Minutos																		
	0,541706788	Horas																		
Tempo de Evolução	272,3737178	Segundos																		
	4,539561963	Minutos																		
	0,075659366	Horas																		

Fonte: O autor, 2019.

Após a execução satisfatória da evolução no modo de variação de valores dos componentes, a mesma topologia será utilizada nos próximos dois modos de evolução. Tais evoluções buscarão alcançar soluções compatíveis com a solução desejada, mas usando a

evolução por adição de componentes (para casos de falhas de circuitos fixos) ou a evolução por alteração completa de componentes (para casos em que todos os parâmetros da topologia são alterados).

4.1.2. Evolução por Adição de Componentes

A evolução por Adição de Componentes é utilizada para casos em que existe um circuito fixo em funcionamento, mas em que a resposta não corresponde à desejada. Tal modo de evolução pode ser entendido como uma evolução a falhas, na qual uma resposta indesejada é observada em um circuito e o algoritmo de evolução deve adicionar componentes a este circuito com o objetivo de melhora-la.

Na evolução da topologia do divisor de tensão, é simulada a queima de um resistor, o qual deixa de fazer parte do circuito (circuito aberto) e, consequentemente, influencia na saída observada. A partir daí o algoritmo adicionará componentes à topologia e avaliará tal adição na resposta observada. Ao final da evolução busca-se uma topologia adicional que aproxime a resposta pós falha do circuito da resposta desejada anteriormente.

O espaço de busca desse tipo de evolução, como explicado anteriormente na Seção 3.6.1, é determinado pela quantidade máxima de nós, componentes e de tipos utilizados. Para esta evolução será possível utilizar no máximo dez componentes, cinco nós e todos serão resistores.

Do mesmo modo que na evolução anterior, o processo de evolução foi realizado utilizando-se a sintonia para dois pares de valores das variáveis gerações de teste e indivíduos de teste. Os pares utilizados foram os que obtiveram melhor desempenho na evolução por valores do circuito. A tabela com os pares e seus respectivos desempenhos podem ser observados na Tabela 11.

Tabela 11 - Sintonia de Valores Para Evolução Para Divisor de Tensão No Modo Adição de Componentes

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,85	0,5	99,84%
7	5	0,38	0,01	99,91%

Fonte: O autor, 2019.

Com o objetivo de comparar o impacto da utilização de correção do cromossomo durante a avaliação, a mesma evolução foi realizada desativando tais ajustes no cromossomo. A tabela com os pares de sintonias e seus respectivos desempenhos sem a função de correção pode ser observada na Tabela 12.

Tabela 12 - Sintonia de Valores no Modo Adição de Componentes Sem Correção do Cromossomo

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,6	0,1	99,09%
7	5	0,4	0,8	99,19%

Fonte: O autor, 2019.

Conforme observado ao se comparar as tabelas acima, a não utilização de um sistema de correção durante a evolução do cromossomo interfere diretamente no processo de sintonia e no valor do desempenho final da evolução. Tal diferença, apesar de pequena para um circuito simples como um divisor de tensão, torna-se crítica na evolução de circuitos mais complexos. Na seção 5.8.2 tal diferença será apresentada de forma quantitativa por meio dos relatórios de evolução apresentados ao final das evoluções. A curva de evolução da simulação com melhor desempenho pode ser observada na Figura 47.

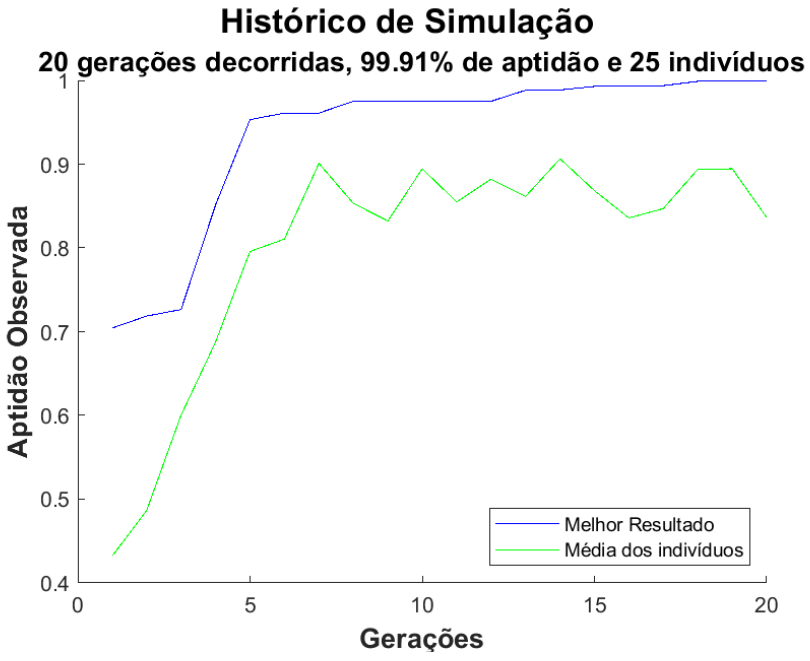

Figura 47 - Curva de Evolução Para Divisor de Tensão No Modo Adição de Componentes

Fonte: O autor, 2019.

Utilizando a topologia que obteve melhor desempenho na evolução deste circuito, obtém-se uma compatibilidade de 99,91% com a resposta observada. Esta resposta e a topologia evoluída podem ser observadas na Figura 48 e na Figura 49, respectivamente.

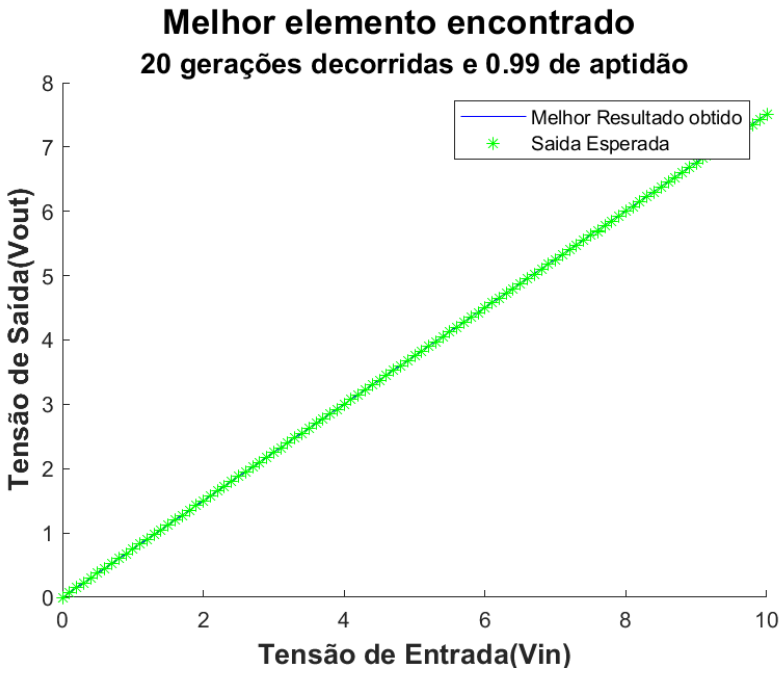

Figura 48 - Resposta Observada Para Divisor de Tensão No Modo Adição de Componentes

Fonte: O autor, 2019.

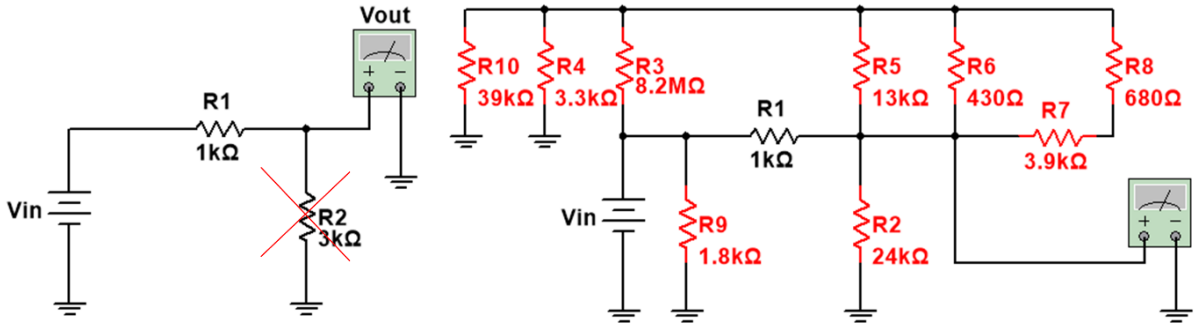

Figura 49 - Topologia Evoluída Para Divisor de Tensão No Modo Adição de Componentes

Fonte: O autor, 2019.

Essa topologia pode ser observada, também, na netlist exportada pelo algoritmo ao final da evolução. Esse arquivo apresenta a representação do indivíduo como cromossomo e sua topologia no formato compreendido pelo simulador de circuitos. Uma parte desse arquivo contendo tais informações pode ser observado abaixo na Figura 50.

```
CKT_DivisorTensao_Adicao_ComCorr_Sint_7e5_Resistores

*
R1 1 2 1.000000E+03
R2 0 2 2.400000E+04
R3 1 5 8.200000E+06
R4 5 0 3.300000E+03
R5 2 5 1.300000E+04
R6 2 5 4.300000E+02
R7 3 2 3.900000E+03
R8 5 3 6.800000E+02
R9 0 1 1.800000E+03
R10 0 5 3.900000E+04

*--------------------------------

*O modo de evolução realizado é o de adição de componentes.
*Circuito fixo utilizado: 1 1 1 2 0 2 0 0 2
*Tipos de componentes: R

*O circuito possui 1 Componentes: 1 Resistores, 0 Mosfets, 0 BJTs, 0
AmpOps, 0 Fontes DC e 0 Capacitores.

*Circuito variável utilizado: 9 1 0 2 5 48 22 0 1 1 5 0 115 2 6 1 5
0 0 62 21 18 1 2 5 5 18 22 5 1 2 5 0 76 6 5 1 3 2 5 72 20 2 1 5 3 3
101 12 7 1 0 1 4 32 29 30 1 0 5 2 73 4 25 1 1 2 1 65 15 23
*Tipos de componentes: R

*O circuito possui 9 Componentes: 9 Resistores, 0 Mosfets, 0 BJTs, 0
AmpOps, 0 Fontes DC e 0 Capacitores..

*--------------------------------

*Modelos Utilizados:
```

Figura 50 - Arquivo Com Netlist do Melhor Indivíduo da Evolução por Adição (Divisor de Tensão)

Fonte: O autor, 2019.

Tal evolução gerou um relatório de evolução com as variáveis de interesse para futuras verificações. Tais variáveis podem ser observadas na Tabela 13.

Tabela 13 - Relatório de Execução Para Divisor de Tensão No Modo Adição de Componentes

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Melhor Resultado	0.7044	0.7187	0.7265	0.8514	0.9532	0.9610	0.9610	0.9752	0.9752	0.9752	0.9752	0.9752	0.9882	0.9886	0.9928	0.9934	0.9938	0.9989	0.9989	0.9991
Média dos Elementos	0.4322	0.4867	0.6005	0.6877	0.7958	0.8106	0.9009	0.8531	0.8322	0.8944	0.8549	0.8821	0.8614	0.9065	0.8684	0.8358	0.8470	0.8939	0.8946	0.8364
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erro 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
População:	25	indivíduos																		
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
Erros de Simulação Críticos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	4365,167689	Segundos																		
	72,75279482	Minutos																		
	1,21254658	Horas																		
Tempo de Evolução	288,0165552	Segundos																		
	4,80027592	Minutos																		
	0,080004599	Horas																		

Fonte: O autor, 2019.

Pode-se observar que a topologia evoluída, apesar de possuir alta compatibilidade, possui a quantidade máxima de componentes determinada para a evolução. Tal resposta

justifica-se pelo fato de a evolução ter sido realizada para apenas um objetivo, a compatibilidade da saída. Caso a evolução fosse realizada para múltiplos objetivos considerando, por exemplo, a potência consumida, ou número de componentes, tal topologia não possuiria uma aptidão tão alta na avaliação. Tal evolução será realizada novamente, na seção 5.6, em uma evolução multiobjetivo visando diminuir a potência.

4.1.3. Evolução por Alteração Completa de Componentes

O próximo modo de evolução corresponde ao modo de Alteração Completa de Componentes. Nesse modo, buscam-se topologias alternativas através da variação completa das variáveis em questão, ou seja, topologias não conhecidas diretamente para o problema.

Como discutido anteriormente na Seção 3.6.1, o espaço de busca para esse tipo de evolução considera a quantidade máxima de componentes utilizada, a quantidade máximas de nós, a quantidade de valores e as tensões de alimentação no caso dos amplificadores operacionais; da mesma forma que a evolução por adição de componentes. Entretanto, tal evolução também possibilita observar a tensão de saída em cada nó que compõe o circuito; com o objetivo de melhorar o processo de busca por uma solução.

Tabela 14 - Tabela de Sintonia de Valores Para Variação Completa (Divisor de Tensão)

Correção do Cromossomo	Utilização de uma Solução Conhecida	Sintonia		Resultados		
		Gerações	Indivíduos	Seleção	Mutação	Desempenho
Não	Não	5	5	0,9	0,1	100%
		7	5	0,5	0,09	100%
	Sim	5	5	0,9	0,2	92,47%
		7	5	0,5	0,09	100%
Sim	Não	5	5	0,2	0,1	100%
		7	5	0,1	0,3	89,91%
	Sim	5	5	0,6	0,4	94,32%
		7	5	0,8	0,2	98,34%

Como observado na tabela anterior, a correção do cromossomo durante a topologia não garante a melhor evolução de uma topologia. Dessa forma, será considerado o melhor resultado a topologia com menor tempo de evolução, maior desempenho e que não realizou a correção do cromossomo. Logo, a evolução sem correção, sem solução inicial e que realizou a sintonia de valores com cinco gerações de cinco indivíduos é considerada a melhor evolução. A parte do arquivo netlist com a descrição do melhor indivíduo e a topologia correspondente podem ser observadas abaixo na Figura 51 e na Figura 52, respectivamente.

```
CKT_DivisorTensao_Completa_SemCorr_SemMelhor_Sint_5e5
_ResistoresFinal

*
R1 5 5 5.100000E+02
R2 4 4 3.900000E+06
R3 1 2 6.200000E+05
R4 4 2 1.500000E+05
R5 4 1 8.200000E+03
R6 5 1 6.200000E+05
R7 5 0 6.200000E+05
R8 3 1 1.200000E+06


*--------------------------------

*O modo de evolução realizado é o de variação total de componentes.
*Circuito variável utilizado: 8 1 5 5 4 86 22 22 1 4 4 1 75 3 16 1 1
2 1 99 18 28 1 4 2 3 24 18 23 1 4 1 2 112 3 30 1 5 1 5 99 28 5 1 5 0
5 99 15 8 1 3 1 0 15 17 25 1 2 4 4 87 9 0 1 5 4 5 68 29 21 5
*Tipos de componentes: R

*O circuito possui 8 Componentes: 8 Resistores, 0 Mosfets, 0 BJTs, 0
AmpOps, 0 Fontes DC e 0 Capacitores.

*--------------------------------

*Modelos Utilizados:
```

Figura 51 - Arquivo Com Netlist do Melhor Indivíduo da Evolução Completa (Divisor de Tensão)

Fonte: O autor, 2019.

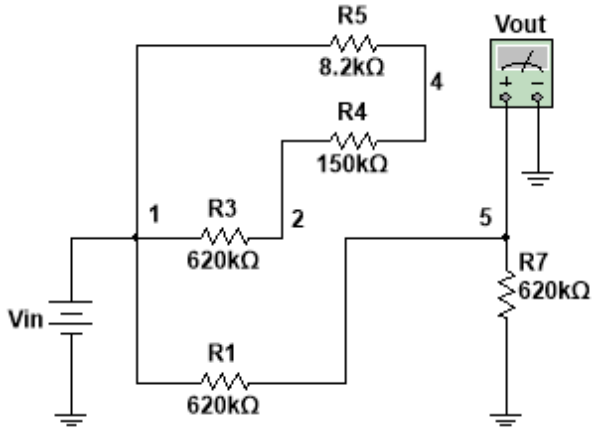

Figura 52 - Topologia Evoluída Para Divisor de Tensão no Modo de Evolução Completo

Fonte: O autor, 2019.

Analisando-se o arquivo netlist e a topologia do circuito, observa-se que o melhor indivíduo corresponde a apenas dois resistores. Isso mostra a necessidade de uma avaliação posterior da topologia evoluída.

4.2. Amplificador

Após a realização da evolução para o circuito divisor de tensão, o algoritmo é utilizado para evoluir a topologia de um circuito amplificador. Tal evolução visa observar se o algoritmo é capaz de evoluir topologias para circuitos mais complexos. Uma topologia básica de um circuito amplificador pode ser observada abaixo na Figura 53.

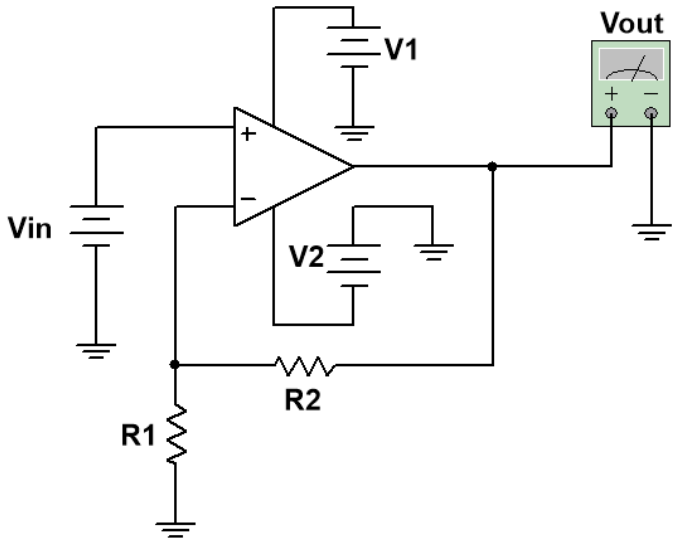

Figura 53 - Circuito Amplificador
Fonte: O autor, 2019.

4.2.1. Evolução por Variação de Valores

A primeira evolução realizada consiste na evolução de valores da topologia básica do circuito amplificador de ganho três. Utilizando a equação do espaço de busca, definida na Seção 3.6.1, determina-se um espaço de busca igual a 15.361 variações.

Do mesmo modo que na evolução para o divisor de tensão, a sintonia de valores para parâmetros foi realizada com diferentes pares de valores de gerações e de indivíduos. A partir do desempenho destes pares é possível observar qual combinação apresenta o melhor desempenho para o processo de evolução. Os pares de valores utilizados durante a sintonia e os respectivos desempenhos podem ser observados na Tabela 15.

Tabela 15 - Resultados Observados Após Sintonia de Valores

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,9	0,9	97%
7	5	0,7	0,08	99,66%
10	5	0,9	0,12	100%

Fonte: O autor, 2019.

Conforme observado na tabela anterior, o algoritmo obteve 100% compatibilidade com a resposta desejada utilizando apenas valores comerciais de componentes. Os dois melhores pares de resultados, com 100% de desempenho na evolução, podem ser observados na Figura 54 e na Figura 55.

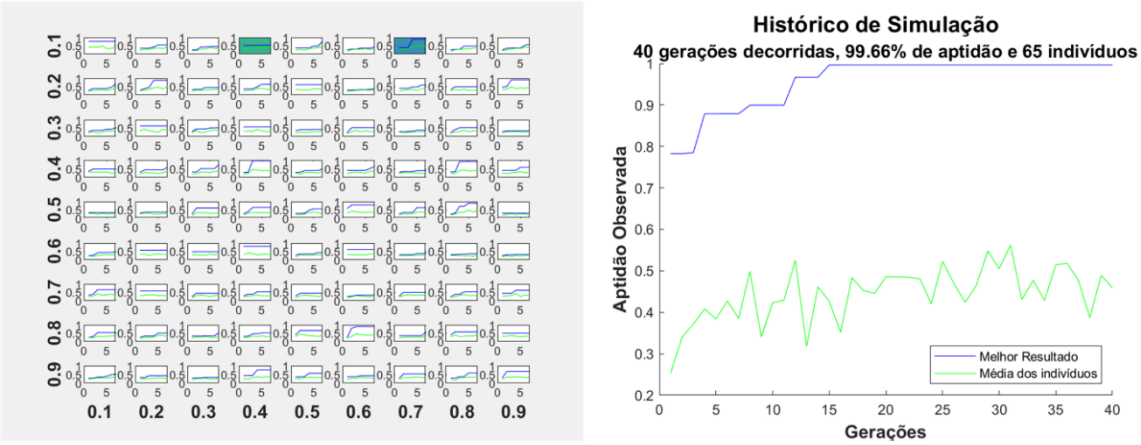

Figura 54 - Evolução Observada Após Sintonia Utilizando 7 Gerações de 5 Indivíduos

Fonte: O autor, 2019.

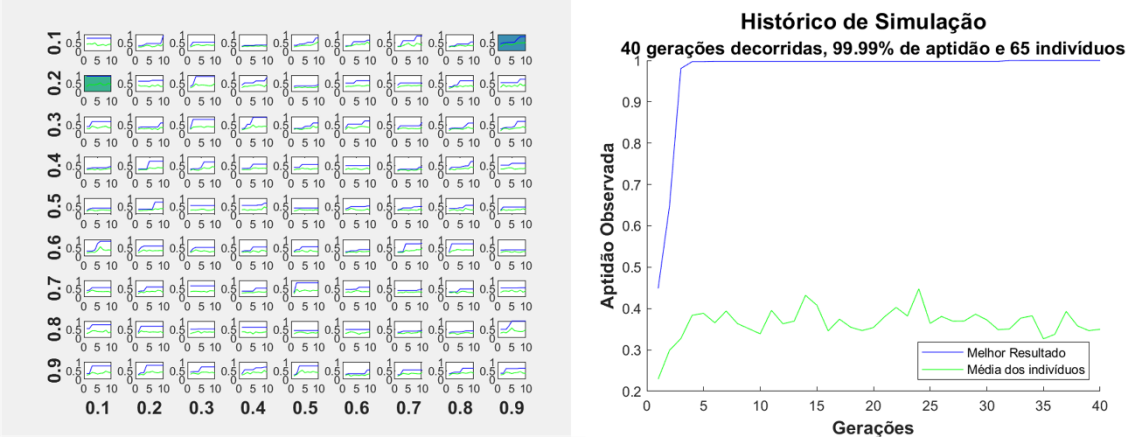

Figura 55 - Evolução Observada Após Sintonia Utilizando 10 Gerações de 5 Indivíduos

Fonte: O autor, 2019.

A evolução realizada para cada um dos casos resultou em duas configurações de valores possíveis para o circuito desejado. A primeira, obtida utilizando a combinação de sete gerações de cinco indivíduos para a sintonia de valores, pode ser observada na Figura 56 e a resposta apresentada pela mesma pode ser observada na Figura 57.

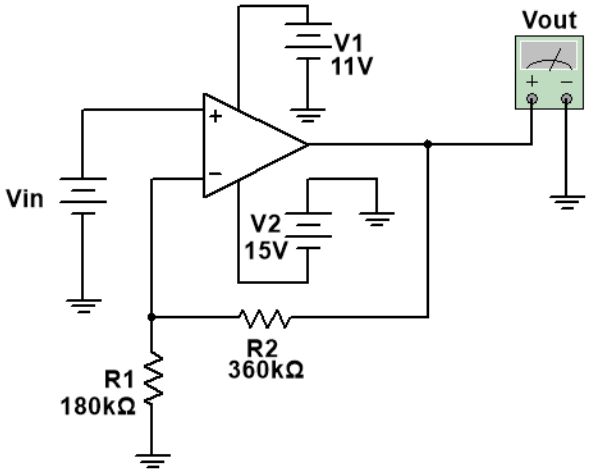

Figura 56 - Topologia Obtida Utilizando 7 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

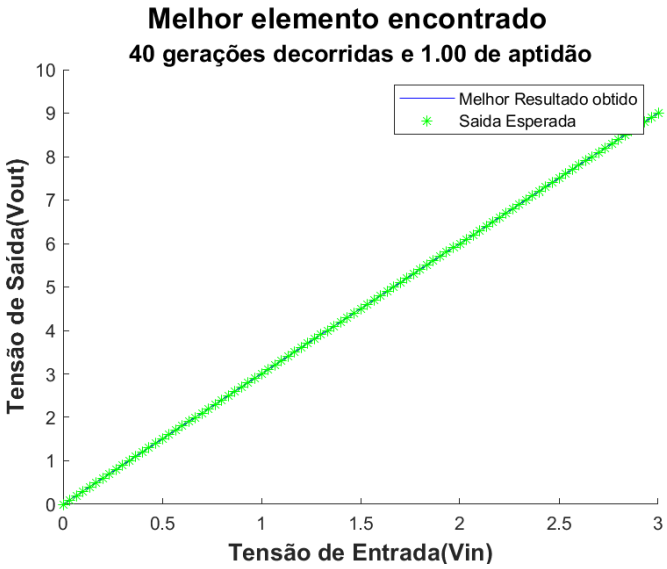

Figura 57 - Resposta Observada Utilizando 7 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

Ao final do processo de evolução, após a apresentação das curvas de evolução e da resposta observada, o algoritmo apresentou um relatório de execução com os dados pertinentes ao processo. Tal relatório, que foi armazenado na pasta junto com os demais arquivos gerados, pode ser observado abaixo na Tabela 16.

Tabela 16 - Relatório de Evolução Utilizando 7 Gerações de 5 Indivíduos na Sintonia

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Melhor Resultado	0.7827	0.7827	0.7850	0.8787	0.8788	0.8789	0.8789	0.8897	0.8897	0.8997	0.8997	0.9670	0.9671	0.9671	0.9965	0.9965	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966
Média dos Elementos	0.2533	0.3396	0.3715	0.4079	0.3839	0.4272	0.3851	0.4986	0.3408	0.4231	0.4291	0.5250	0.3178	0.4617	0.4262	0.3518	0.4834	0.4528	0.4455	0.4857	0.4858	0.4846	0.4808	0.4208	0.5230	0.4696	0.4237	0.4662	0.5477	0.5048	0.5624	0.4305	0.4773	0.4286	0.5147	0.5183	0.4762	0.3867	0.4890	0.4589
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erro 1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 2	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 4	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 7	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
População:	65	indivíduos																																						
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erros de Simulação Críticos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	2003,357274 Segundos																																							
	33,38928789 Minutos																																							
	0,556488132 Horas																																							
Tempo de Evolução	1458,476952 Segundos																																							
	24,3079492 Minutos																																							
	0,405132487 Horas																																							

A segunda topologia, obtida utilizando a combinação de dez gerações de cinco indivíduos para a sintonia de valores, pode ser observada na Figura 58 e a resposta pode ser observada na Figura 59.

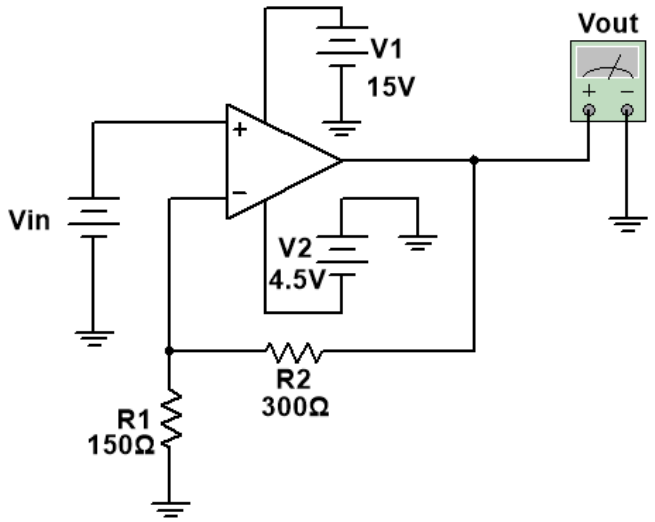

Figura 58 - Topologia Obtida Utilizando 10 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

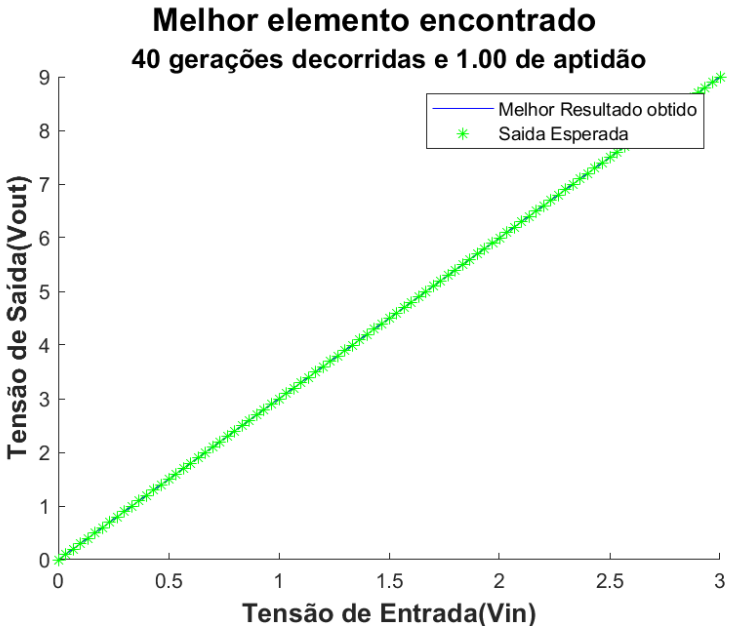

Figura 59 - Resposta Observada Utilizando 10 Gerações de 5 indivíduos na Sintonia

Fonte: O autor, 2019.

Do mesmo modo que na evolução anterior, após a apresentação das curvas de evolução e da resposta observada, o algoritmo criou o relatório de execução com os dados pertinentes ao processo. Este relatório pode ser observado na Tabela 17.

Tabela 17 - Relatório de Evolução Utilizando 10 Gerações de 5 Indivíduos na Sintonia

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Melhor Resultado	0.4487	0.6482	0.9805	0.9972	0.9972	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9978	0.9998	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
Média dos Elementos	0.2294	0.2990	0.3273	0.3838	0.3885	0.3656	0.3941	0.3636	0.3513	0.3388	0.3956	0.3633	0.3695	0.4319	0.4082	0.3460	0.3743	0.3547	0.3468	0.3542	0.3808	0.4028	0.3922	0.4476	0.3642	0.3953	0.3695	0.3698	0.3863	0.3725	0.3490	0.3507	0.3771	0.3824	0.3267	0.3378	0.3931	0.3582	0.3465	0.3498
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erro 1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 2	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 4	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Erro 7	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
População:	65	indivíduos																																						
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erros de Simulação Críticos	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:		0 Segundos																																						
		0 Minutos																																						
		0 Horas																																						
Tempo de Evolução		1442,368819 Segundos																																						
		24,03948032 Minutos																																						
		0,400658005 Horas																																						

Ambas as topologias, apesar de apresentarem 100% de compatibilidade com a saída desejada, possuem grande variação na ordem de grandeza dos resistores utilizados. Em um primeiro momento, em que apenas a compatibilidade da saída é observada, o algoritmo não diferencia tais ordens de grandeza utilizadas.

Entretanto, quando o mesmo evoluir para múltiplos objetivos, a ordem de grandeza afetará diretamente a aptidão observada. Tal diferença influenciará diretamente no valor de potência consumida pelo circuito e, consequentemente, na aptidão dada a cada possível solução que compõe a população sendo evoluída.

4.2.2. Evolução por Adição de Componentes

Após a realização da evolução de uma topologia para o circuito amplificador, simularemos uma situação de falha no circuito evoluído. Essa simulação testará a plataforma buscando uma solução para a falha através do modo Adição de Componentes.

A falha simulada será a queima de um de seus resistores, mais precisamente o resistor com valor de 180 kΩ. Após a queima desse resistor, a conexão dos nós desse componente é considerada inexistente. A partir do circuito resultante, o algoritmo adicionará componentes a topologia e avaliará tal adição na resposta observada. Ao final da evolução busca-se uma topologia adicional que seja capaz de corrigir o modo de funcionamento do circuito amplificador.

Do mesmo modo que nas evoluções realizadas anteriormente, o processo de evolução foi realizado utilizando-se a sintonia com o par de valores que obteve o melhor desempenho na evolução anterior, ou seja, sete gerações com cinco indivíduos. Definidas as variáveis de sintonia, será avaliado o desempenho da evolução com e sem a correção do cromossomo durante a evolução. Tal comparação visa avaliar o impacto dessa função no desempenho da evolução. O impacto dessa função no aparecimento de erros durante o processo será foco da discussão ao decorrer da dissertação. A tabela comparativa com os desempenhos de cada caso pode ser observada abaixo na Tabela 18.

Tabela 18 - Comparação de Desempenho da Evolução do Amplificador (Modo Adição de Componentes)

Correção do Cromossomo	Seleção	Mutação	Desempenho
Não	0,3	0,01	75,40%
Sim	0,5	0,5	99,66%

Fonte: O autor, 2019.

Como observado na tabela acima, a correção do cromossomo interfere diretamente no desempenho observado. O processo de sintonia e a curva de evolução podem ser observadas na Figura 60. A curva resultante dessa evolução pode ser observada em seguida na Figura 61.

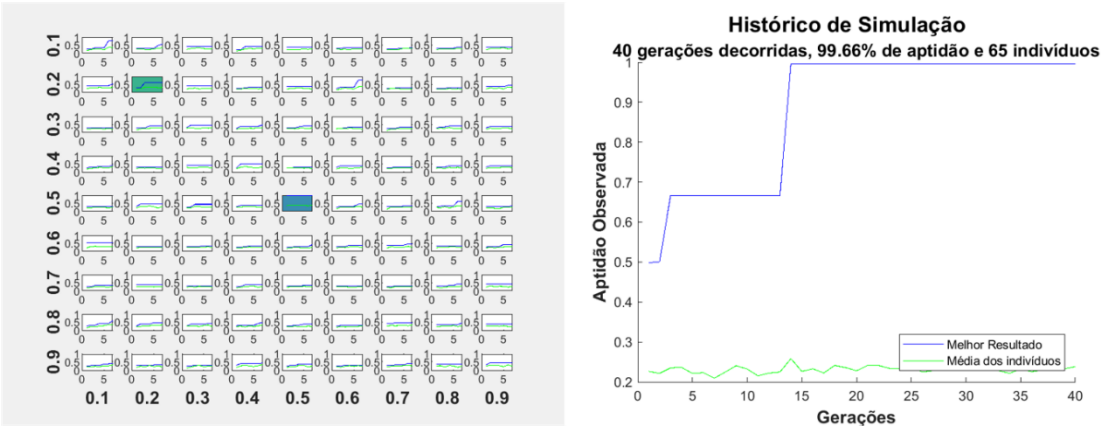

Figura 60 - Evolução de Circuito Amplificador no Modo Adição de Componentes

Fonte: O autor, 2019.

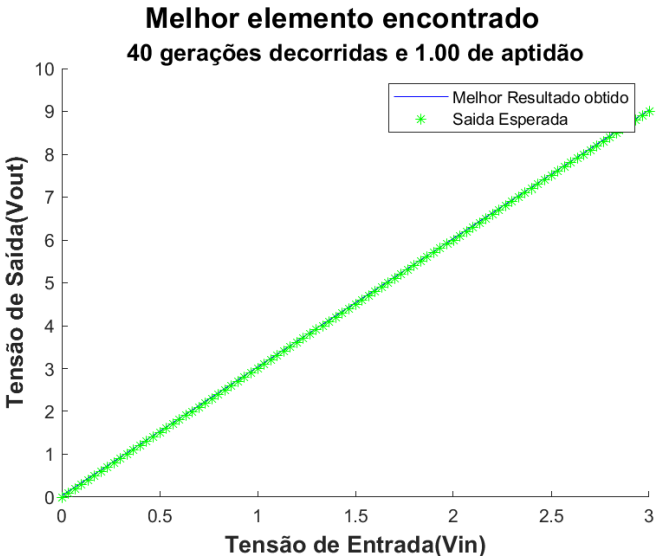

Figura 61 - Resposta Observada Para Circuito Amplificador no Modo Adição de Componentes

Fonte: O autor, 2019.

A netlist com a descrição do circuito evoluído e a topologia correspondentes podem ser observados abaixo na Figura 62 e na Figura 63, respectivamente. É possível observar que a evolução realizada obteve como resposta a topologia anteriormente utilizada. Tal resposta foi obtida adicionando outro resistor com o mesmo valor do resistor queimado.

```
CKT_Amplificador_Adicao_ComCorr_Sint_7e5_Resistores

*
R1 2 3 3.600000E+05
XAmpOp1 1 2 3 Vp1 Vn1 OPAMP2
*Fontes Amp Op
VpXAmpOp1  Vp1 0 DC   +11V
VnXAmpOp1  Vn1 0 DC   -15V
R2 2 0 1.800000E+05


*--------------------------------

*O modo de evolução realizado é o de adição de componentes.
*Circuito fixo utilizado: 2 1 2 3 0 69 0 0 2 1 2 3 42 22 30 3
*Tipos de componentes: RA

*O circuito possui 2 Componentes: 1 Resistores, 0 Mosfets, 0 BJTs, 1
AmpOps, 0 Fontes DC e 0 Capacitores.

*Circuito variável utilizado: 1 1 2 0 2 34 19 1 2 2 2 0 85 21 26 1 0
2 4 63 17 14 1 5 5 3 120 8 7 1 5 3 1 50 20 24 2 3 5 1 116 13 14 2 5
5 1 45 7 17 2 2 5 5 120 0 20 2 5 0 0 47 19 26 1 3 0 5 8 13 12
*Tipos de componentes: RA

*O circuito possui 1 Componentes: 1 Resistores, 0 Mosfets, 0 BJTs, 0
AmpOps, 0 Fontes DC e 0 Capacitores..

*--------------------------------

*Modelos Utilizados:
```

Figura 62 - Netlist do Melhor Indivíduo Para Circuito Amplificador (Adição de Componentes)

Fonte: O autor, 2019.

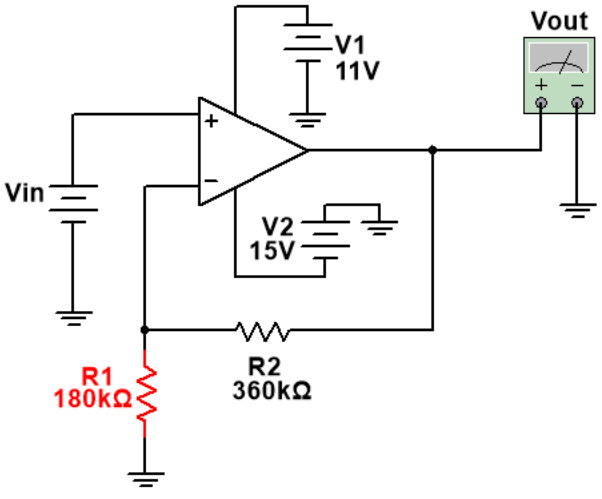

Figura 63 - Topologia Evoluída Para Circuito Amplificador no Modo Adição de Componentes

Fonte: O autor, 2019.

4.2.3. Evolução por Alteração Completa de Componentes

Após a evolução por adição de componentes, realizar-se-á a evolução no modo Alteração de Componentes. Nessa evolução será evoluída toda a topologia em busca de uma solução adequada. Essa busca poderá ou não utilizar uma solução de partida, que neste caso será a topologia evoluída na evolução de valores.

A saída objetivo dessa evolução será um amplificador que duplique a tensão se entrada. A solução de partida conhecida, amplificador com ganho três, poderá ou não ser usada; sendo utilizada na comparação de desempenho. Semelhante ao modo Adição de Componentes, a função de correção do cromossomo poderá ou não ser utilizada neste modo. A configuração de sintonia para os valores de mutação e seleção são determinados pelo melhor desempenho nas evoluções anteriores, ou seja, sete gerações de cinco indivíduos.

Com estes valores definidos, será analisada a possibilidade de correção do cromossomo durante a evolução e a possibilidade de utilizar uma solução conhecida durante a evolução. A tabela com as respectivas situações e seus desempenhos podem ser observados na Tabela 19.

Tabela 19 - Desempenho de Evoluções no Modo Completo Para o Circuito Amplificador

Correção do Cromossomo	Utilização de uma Solução Conhecida	Seleção	Mutação	Desempenho
Não	Não	0,9	0,1	48,04%
Não	Sim	0,9	0,01	63,92%
Sim	Não	0,9	0,1	83,80%
Sim	Sim	0,6	0,3	82,53%

Fonte: O autor, 2019.

Como observado na tabela anterior, o desempenho da evolução foi superior no caso sem a utilização de uma solução de partida. A curva de evolução que obteve tal resultado pode ser observado abaixo na Figura 64. Através da curva de evolução constata-se a possibilidade de melhoria do desempenho observado pelo aumento das gerações utilizadas.

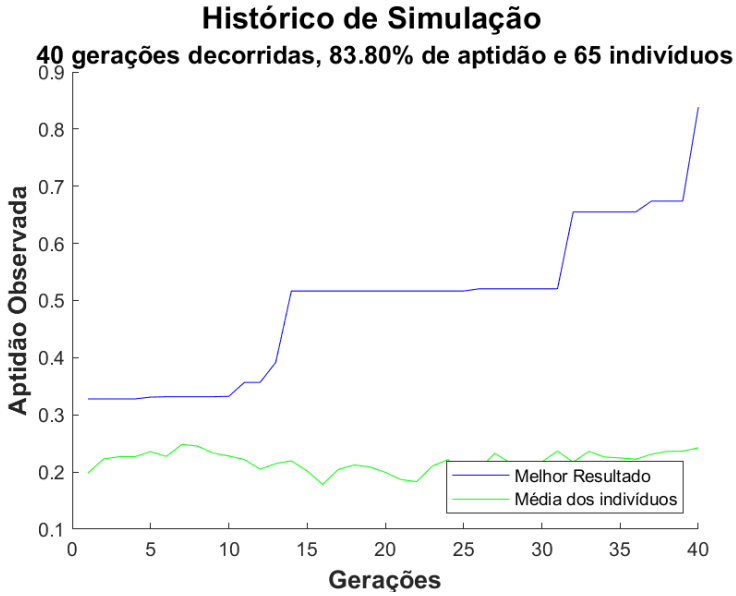

Figura 64 - Curva de Evolução de Circuito Amplificador no Modo Variação Completa de Componentes

Fonte: O autor, 2019.

Com o objetivo de verificar o desempenho com um número maior de gerações, a evolução é realizada novamente utilizando os parâmetros obtidos novamente. A curva de evolução utilizando um número maior de gerações pode ser observada na Figura 65. O aumento do número de gerações melhorou a aptidão de 83% para 94%. Dessa forma, a ocorrência de melhoras abruptas próximas ao final da simulação é um indicativo de que o aumento do número de gerações pode resultar na melhora da resposta observada.

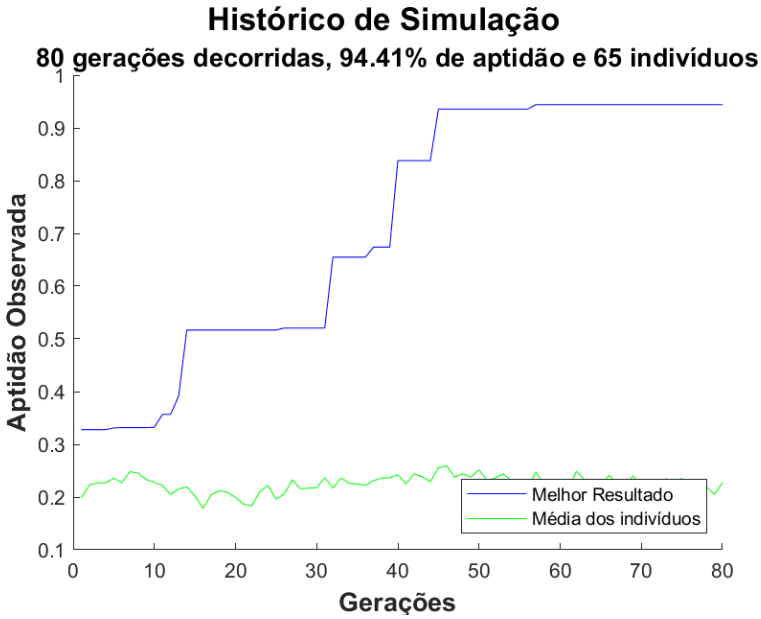

Figura 65 - Curva de Evolução de Circuito Amplificador Atualizada (Variação Completa de Componentes)

Fonte: O autor, 2019.

A topologia evoluída e sua resposta observada ao final do processo de evolução podem ser observadas na Figura 66 e na Figura 67, respectivamente.

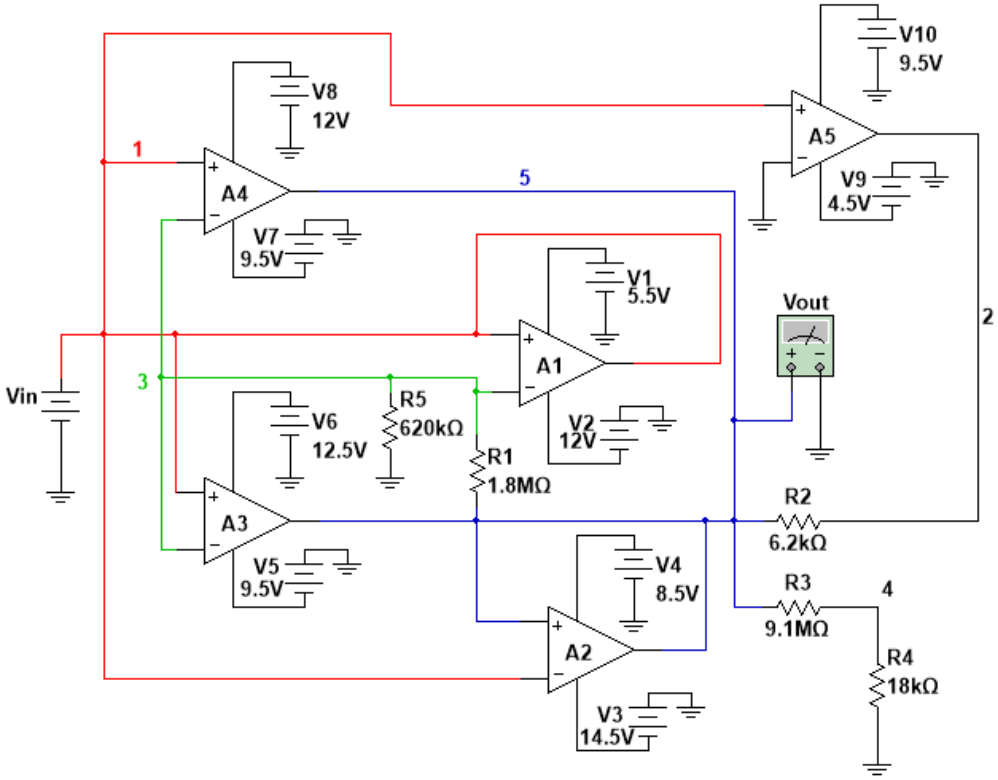

Figura 66 - Topologia Evoluída Para o Circuito Amplificador (Variação Completa de Componentes)

Fonte: O autor, 2019.

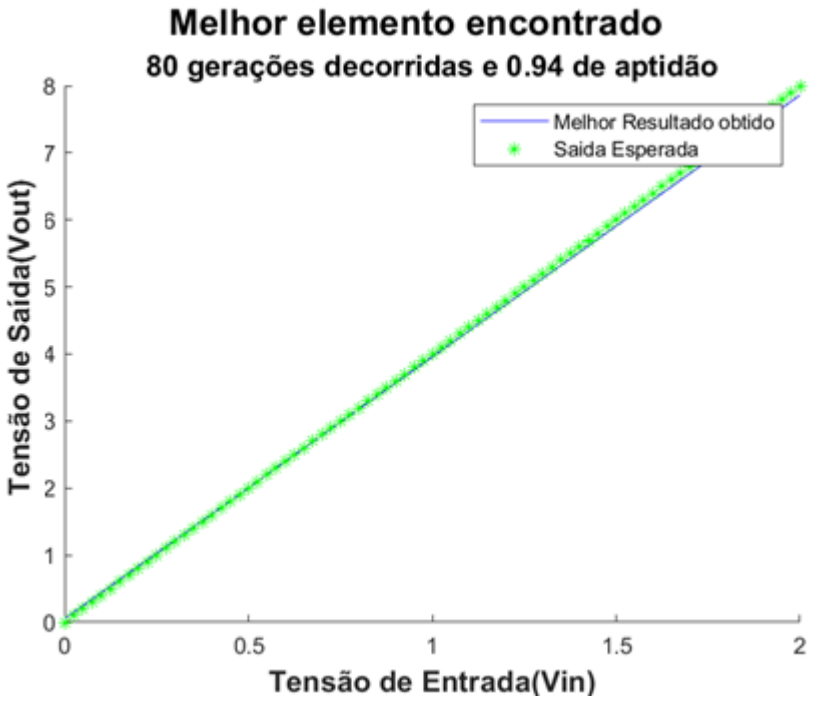

Figura 67 - Saída Observada Para o Circuito Amplificador (Variação Completa de Componentes)

Fonte: O autor, 2019.

Pode-se observar que o circuito evoluído não se assemelha à topologia conhecida para o circuito. Entretanto, a saída apresentada por essa topologia é bem próxima da desejada. Isso comprova que a plataforma é capaz de realizar buscas a topologias não usuais, mas que satisfaçam uma determinada saída desejada. Uma parte do arquivo gerado ao final da evolução pode ser observado abaixo na Figura 68. Tal arquivo contém a topologia e o cromossomo referente ao melhor indivíduo, possibilitando sua futura utilização.

```
CKT_Amplificador_Completa_ComCorr_SemMelhor_09e01_Resistores2

*
XAmpOp1 1 3 1 Vp1 Vn1 OPAMP2
*Fontes Amp Op
VpXAmpOp1  Vp1 0 DC   +5.500000e+00V
VnXAmpOp1  Vn1 0 DC   -12V
R1 5 3 1.800000E+06
XAmpOp2 5 1 5 Vp2 Vn2 OPAMP2
*Fontes Amp Op
VpXAmpOp2  Vp2 0 DC   +8.500000e+00V
VnXAmpOp2  Vn2 0 DC   -1.450000e+01V
XAmpOp3 1 3 5 Vp3 Vn3 OPAMP2
*Fontes Amp Op
VpXAmpOp3  Vp3 0 DC   +1.250000e+01V
VnXAmpOp3  Vn3 0 DC   -9.500000e+00V
R2 5 2 6.200000E+03
R3 5 4 9.100000E+06
XAmpOp4 1 3 5 Vp4 Vn4 OPAMP2
*Fontes Amp Op
VpXAmpOp4  Vp4 0 DC   +12V
VnXAmpOp4  Vn4 0 DC   -9.500000e+00V
XAmpOp5 1 0 2 Vp5 Vn5 OPAMP2
*Fontes Amp Op
VpXAmpOp5  Vp5 0 DC   +9.500000e+00V
VnXAmpOp5  Vn5 0 DC   -4.500000e+00V
R4 0 4 1.800000E+04
R5 3 0 6.200000E+05


*------------------------------

*O modo de evolução realizado é o de variação total de componentes.
*Circuito variável utilizado: 10 2 1 3 1 6 11 24 1 5 3 1 35 7 21 2 5
1 5 19 17 29 2 1 3 5 77 25 19 1 5 2 5 97 24 30 1 5 4 0 120 23 24 2 1
3 5 107 24 19 2 1 0 2 34 19 9 1 0 4 5 33 25 13 1 3 0 3 99 26 18 5
*Tipos de componentes: RA

*O circuito possui 10 Componentes: 5 Resistores, 0 Mosfets, 0 BJTs,
5 AmpOps, 0 Fontes DC e 0 Capacitores.

*------------------------------

*Modelos Utilizados:
```

Figura 68 - Netlist Evoluída Para o Circuito Amplificador (Variação Completa de Componentes)

Fonte: O autor, 2019.

4.3. Função Z

Após a realização da evolução para o circuito amplificador, o algoritmo é utilizado para evoluir a topologia de um circuito gerador de Função Z. Tal circuito recebe esse nome devido a sua forma de onda similar a letra Z. A topologia básica de um circuito gerador de Função Z pode ser observada abaixo na Figura 69.

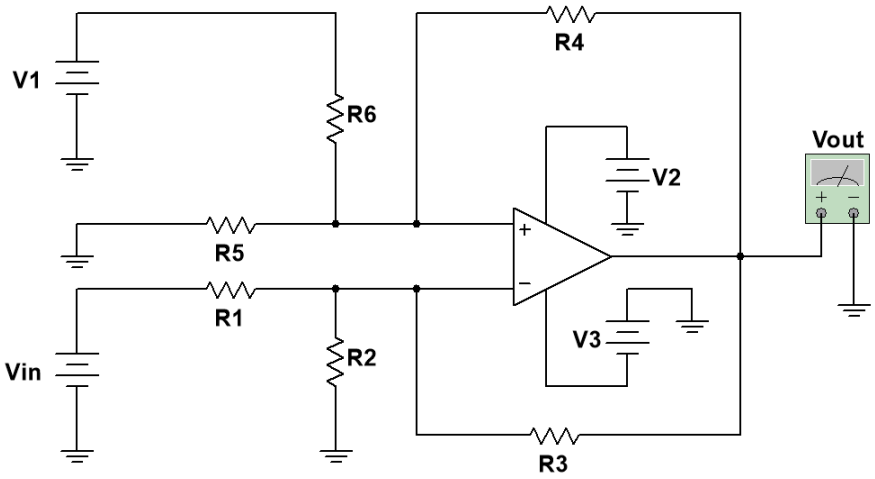

Figura 69 - Circuito Gerador de Função de Pertinência Z
Fonte: O autor, 2019.

4.3.1. Evolução por Variação de Valores

A primeira evolução realizada consiste na evolução de valores da topologia básica para o circuito gerador da Função Z. Utilizando a equação do espaço de busca, definida na Seção 3.6.1, determina-se um espaço de busca igual a 2.985.984.000.992 variações.

Do mesmo modo que na evolução para o divisor de tensão, a sintonia de valores para parâmetros foi realizada com diferentes pares de valores de gerações e de indivíduos. A partir do desempenho destes pares é possível observar qual combinação apresenta o melhor desempenho para o processo de evolução. Os pares de valores utilizados durante a sintonia e os respectivos desempenhos podem ser observados na Tabela 20.

Tabela 20 - Resultados Observados Após Sintonia de Valores

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,4	0,07	95,53%
7	5	0,7	0,8	64,52%
10	5	0,8	0,11	71,60%

Fonte: O autor, 2019.

Conforme observado na tabela anterior, o algoritmo obteve 96% compatibilidade com a resposta desejada utilizando apenas valores comerciais de componentes. O processo de sintonia e a curva de evolução que obteve melhor desempenho podem ser observados na Figura 70 e na Figura 71, respectivamente.

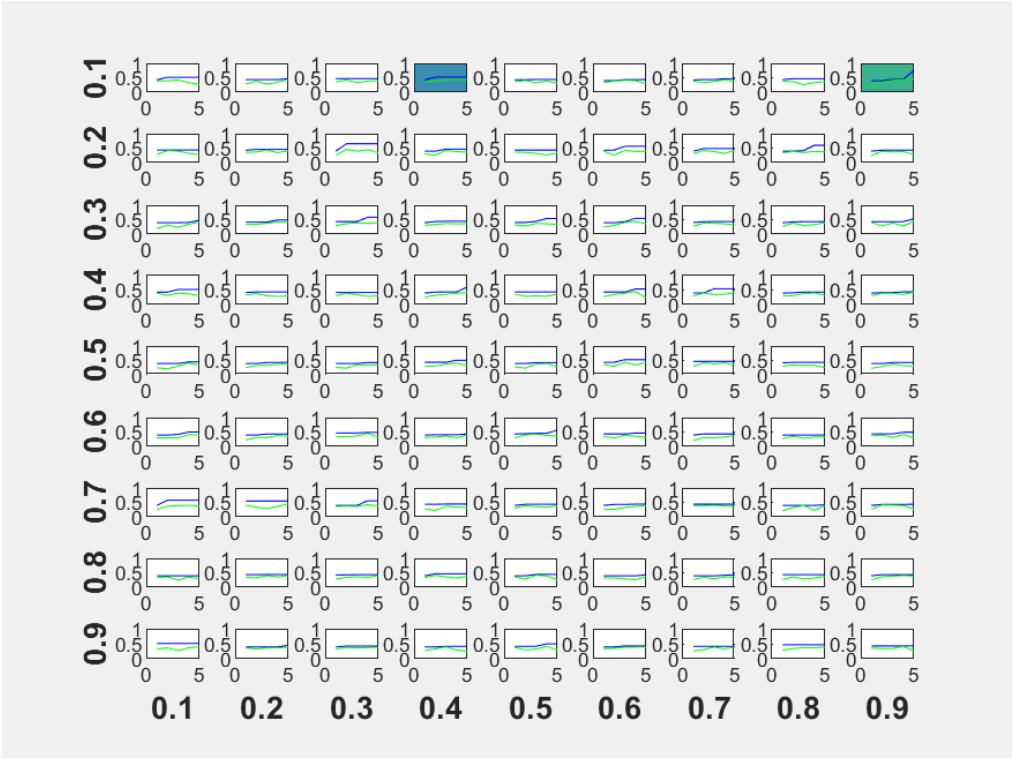

Figura 70 - Processo de Sintonia Para Evolução de Valores da Função Z

Fonte: O autor, 2019.

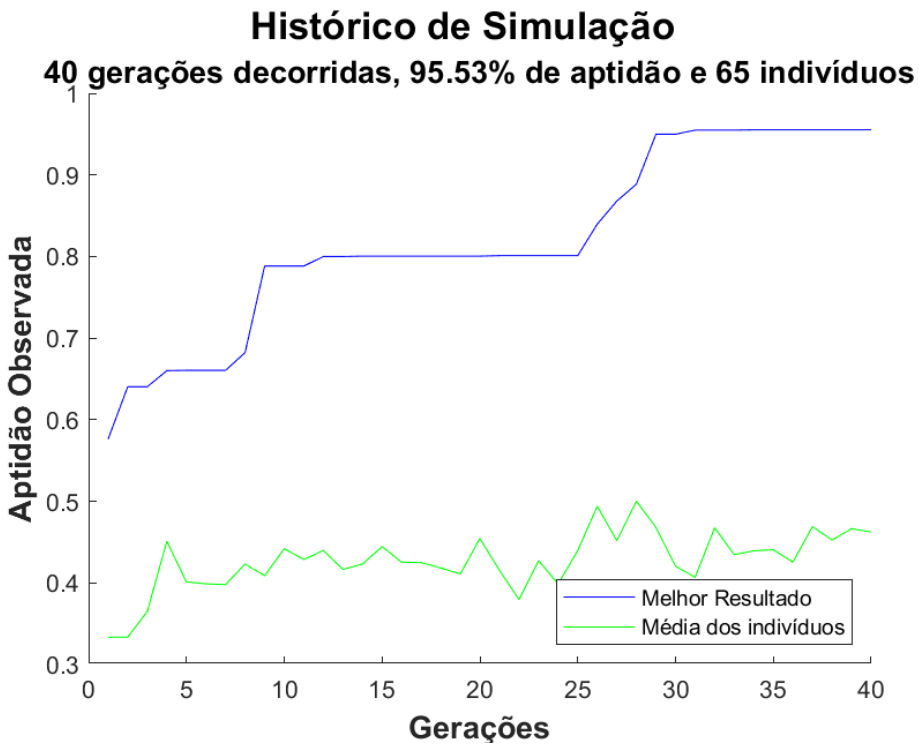

Figura 71 - Curva de Evolução de Valores da Função Z

Fonte: O autor, 2019.

Tal evolução resultou em uma topologia de valores comerciais cuja saída se aproxima da saída desejada. Este conjunto de valores pode ser observado na Figura 72. A respectiva saída observada, que obteve 96% de compatibilidade com a desejada, também pode ser observada na Figura 73.

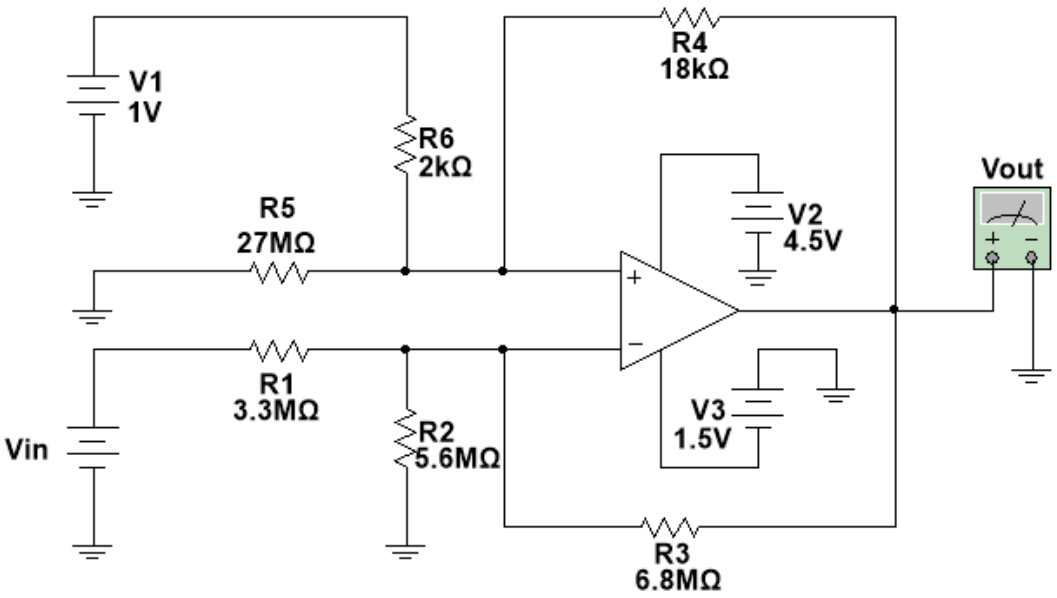

Figura 72 - Topologia Obtida Para Função Z Utilizando Variação de Valores

Conforme observado na saída do melhor elemento, a resposta aproxima-se bastante da saída desejada. Tal saída foi obtida pela utilização de componentes com valores comerciais e considerando o impacto da alimentação do amplificador operacional na resposta final.

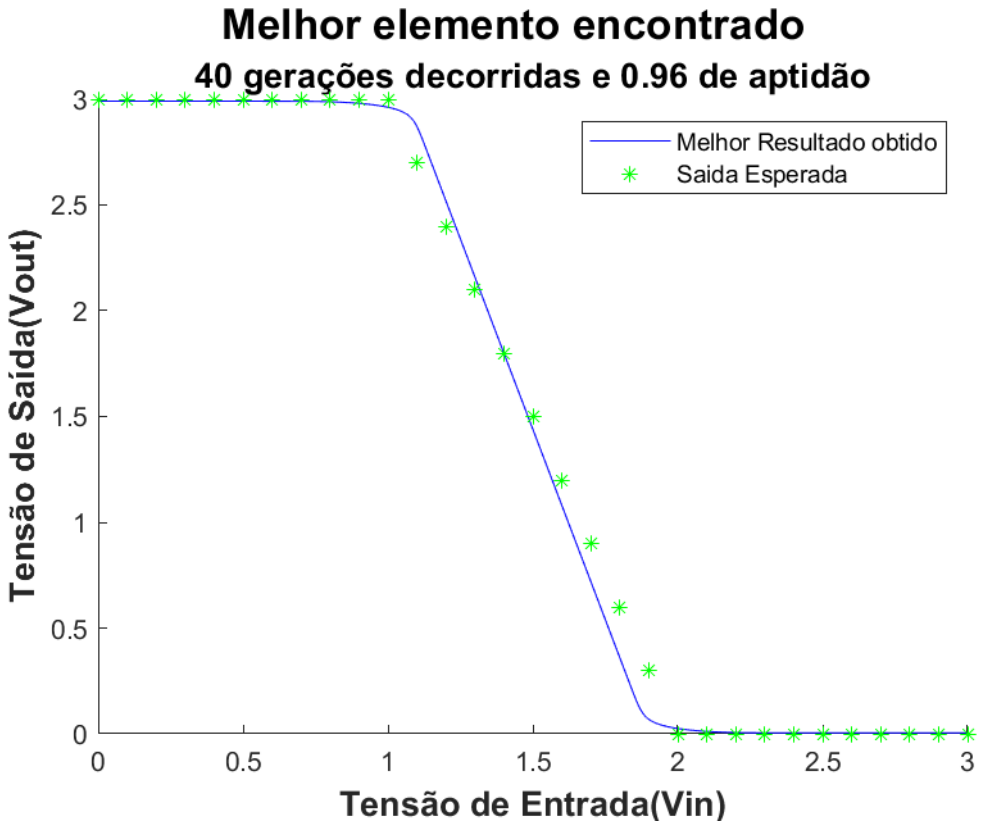

Figura 73 - Saída Observada Para Função Z Pela Variação de Valores

Fonte: O autor, 2019.

A evolução também gerou um relatório com um resumo de todo processo. Este relatório pode ser observado abaixo na Tabela 21 e apresenta grandezas de interesse como o tempo de evolução e o tempo necessário para realizar a sintonia de valores.

Tabela 21 - Relatório de Evolução Para Função Z no Modo Variação de Valores

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Melhor Resultado	0.5763	0.6404	0.6404	0.6601	0.6605	0.6605	0.6605	0.6823	0.7883	0.7883	0.7883	0.8000	0.8000	0.8005	0.8005	0.8005	0.8005	0.8005	0.8005	0.8005	0.8011	0.8011	0.8011	0.8011	0.8011	0.8397	0.8680	0.8887	0.9499	0.9499	0.9549	0.9549	0.9549	0.9553	0.9553	0.9553	0.9553	0.9553	0.9553	0.9553
Média dos Elementos	0.3335	0.3336	0.3650	0.4510	0.4012	0.3989	0.3979	0.4232	0.4088	0.4420	0.4286	0.4398	0.4166	0.4231	0.4445	0.4254	0.4249	0.4183	0.4111	0.4544	0.4155	0.3796	0.4273	0.3988	0.4396	0.4939	0.4518	0.5003	0.4679	0.4206	0.4068	0.4675	0.4347	0.4393	0.4406	0.4254	0.4691	0.4525	0.4664	0.4624
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erro 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
População:	65	indivíduos																																						
Erros de Simulação Críticos	0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	1362,526613 Segundos																																							
	22,70877689 Minutos																																							
	0,378479615 Horas																																							
Tempo de Evolução	1371,014372 Segundos																																							
	22,85023954 Minutos																																							
	0,380837326 Horas																																							

4.3.2. Evolução por Adição de Componentes

Após a realização da evolução de uma topologia para o circuito amplificador, simular-se-á uma situação de falha no circuito evoluído. Essa simulação testará a plataforma buscando uma solução para a falha através do modo Adição de Componentes.

A falha simulada será a queima de um de seus resistores, mais precisamente o de 27 MΩ. Após a queima, a conexão dos nós desse componente é considerada inexistente. A partir do circuito resultante, o algoritmo adicionará componentes a topologia e avaliará tal adição na resposta observada. Ao final da evolução busca-se uma topologia adicional que seja capaz de corrigir o modo de funcionamento do circuito amplificador.

Do mesmo modo que nas evoluções realizadas anteriormente, o processo de evolução foi realizado utilizando-se a sintonia com o par de valores que obteve o melhor desempenho na evolução anterior, ou seja, sete gerações com cinco indivíduos. Definidas as variáveis de sintonia, realizaremos o desempenho da evolução com e sem a correção do cromossomo durante a evolução. Tal comparação visa avaliar o impacto dessa função no desempenho da evolução. O impacto dessa função no aparecimento de erros durante o processo será foco da discussão ao decorrer da dissertação. A tabela comparativa com os desempenhos de cada caso pode ser observada abaixo na Tabela 22.

Tabela 22 - Desempenho de Evolução (Circuito Gerador de Função Z no Modo Adição)

Correção do Cromossomo	Seleção	Mutação	Desempenho
Não	0,1	0,8	95,55%
Sim	0,9	0,2	98,03%

Fonte: O autor, 2019.

Como observado na tabela acima, a correção do cromossomo interfere diretamente no desempenho observado. O processo de sintonia e a curva de evolução podem ser observadas na Figura 74. A curva resultante dessa evolução pode ser observada em seguida na Figura 75.

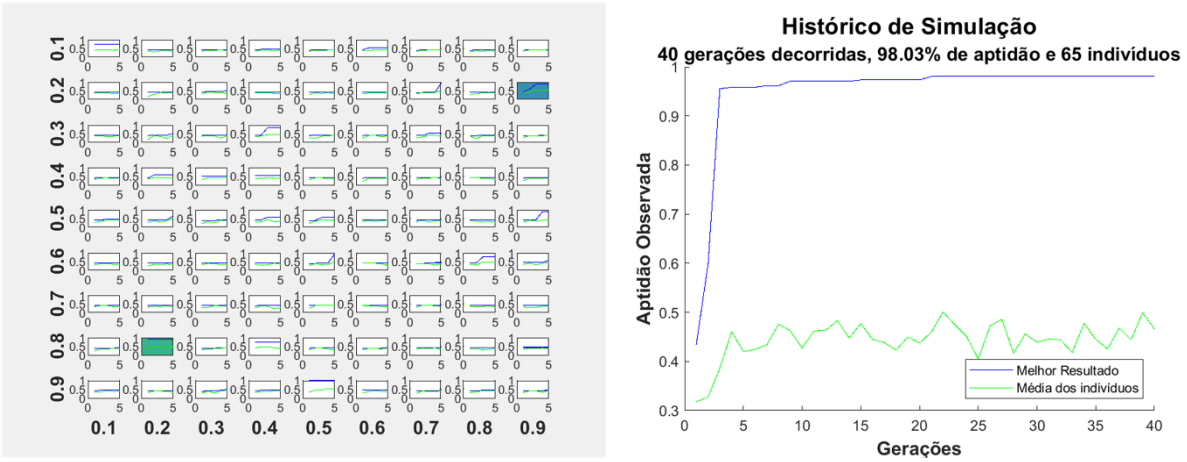

Figura 74 - Curva de Evolução do Circuito Gerador de Função Z (Modo Adição de Componentes)

Fonte: O autor, 2019.

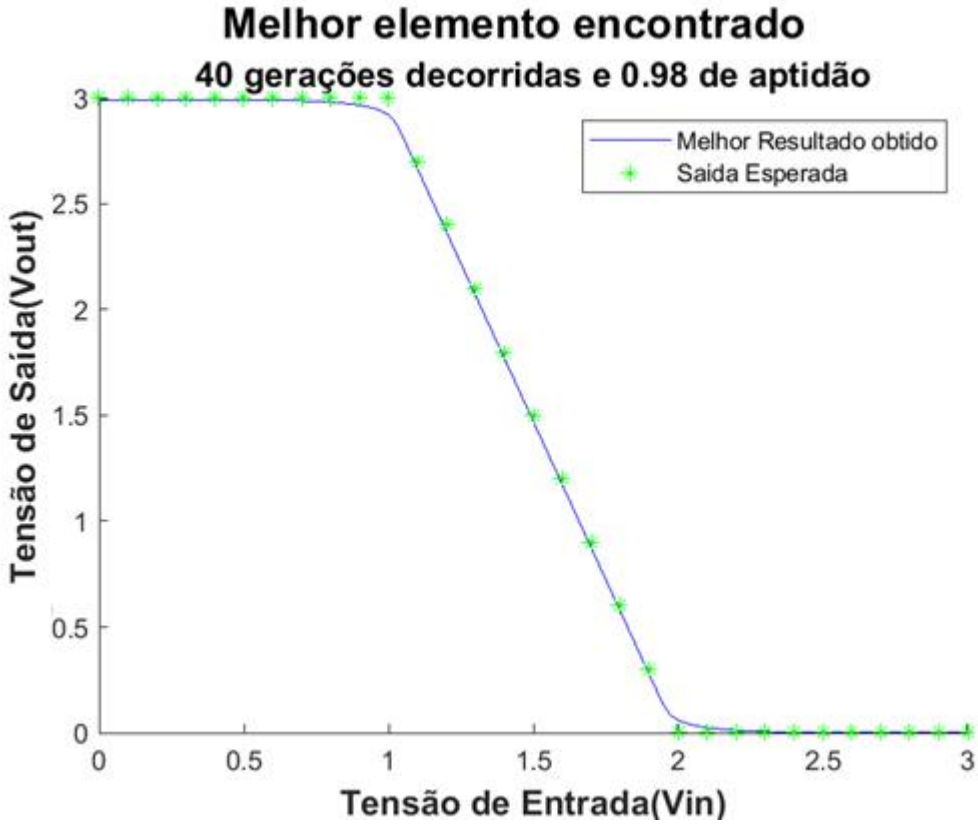

Figura 75 - Resposta Observada Para Circuito Gerador de Função Z (Modo Adição de Componentes)

Fonte: O autor, 2019.

Após a evolução o melhor indivíduo é apresentado, no MATLAB, e exportado em um arquivo netlist; contendo sua representação em valores de cromossomos e suas conexões e valores de componentes. Uma parte do arquivo netlist pode ser observado na Figura 76. Em seguida, uma comparação entre o circuito com falha e o corrigido também pode ser observada na Figura 77.

```
CKT_FuncaoZ_Adicao_ComCorr_Sint_5e5_Resistores

*
R1 1 2 3.300000E+06
R2 2 0 5.600000E+06
R3 2 4 6.800000E+06
R4 3 4 1.800000E+04
XAmpOp1 3 2 4 Vp1 Vn1 OPAMP2
*Fontes Amp Op
VpXAmpOp1  Vp1 0 DC   +4.500000e+00V
VnXAmpOp1  Vn1 0 DC   -1.500000e+00V
R5 3 5 2.000000E+03
V1 5 0 dc 1V
R6 3 1 5.600000E+04
R7 5 3 9.100000E+03
R8 3 5 9.100000E+06


*--------------------------------

*O modo de evolução realizado é o de adição de componentes.
*Circuito fixo utilizado: 7 1 1 2 0 65 0 0 1 2 0 0 95 0 0 1 2 4 0
105 0 0 1 3 4 0 33 0 0 2 3 2 4 1 9 3 1 3 5 0 37 0 0 3 5 0 0 1 0 0 4
*Tipos de componentes: RAV

*O circuito possui 7 Componentes: 5 Resistores, 0 Mosfets, 0 BJTs, 1
AmpOps, 1 Fontes DC e 0 Capacitores.

*Circuito variável utilizado: 3 1 3 1 5 93 16 28 1 5 3 4 117 27 24 1
3 5 3 120 12 3 1 0 0 1 93 1 8 1 5 2 5 59 26 5 1 2 1 4 56 18 1 1 5 5
1 100 24 20 1 0 4 4 118 28 0 1 4 0 4 12 30 20 1 5 5 5 91 7 14
*Tipos de componentes: RAV

*O circuito possui 3 Componentes: 3 Resistores, 0 Mosfets, 0 BJTs, 0
AmpOps, 0 Fontes DC e 0 Capacitores..
```

Figura 76 - Netlist Evoluída Para o Circuito Função Z(Adição de Componentes)

Fonte: O autor, 2019.

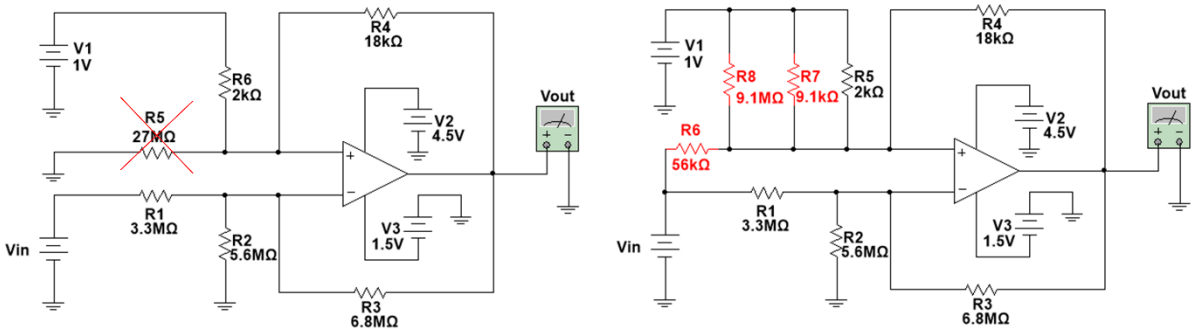

Figura 77 - Topologia Evoluída Para o Circuito Função Z(Adição de Componentes)

Fonte: O autor, 2019.

4.3.3. Evolução por Alteração Completa de Componentes

Após evoluir o circuito por adição de componentes, realizaremos a evolução no modo Alteração de Componentes. Nessa evolução será evoluída toda a topologia em busca de uma solução adequada. Essa busca poderá ou não utilizar uma solução de partida, que neste caso será a topologia evoluída na evolução de valores.

A saída objetivo dessa evolução será um circuito gerador de função z tensão máxima de entrada e de saída igual a cinco volts. A topologia anterior avaliada poderá ou não ser utilizada, topologia cuja tensão máxima de entrada e saída são iguais a três volts. Assim como no modo Adição de Componentes, a função de correção do cromossomo poderá ou não ser utilizada neste modo. A configuração de sintonia para os valores de mutação e seleção são determinados pelo melhor desempenho nas evoluções anteriores, ou seja, cinco gerações de cinco indivíduos.

Com estes valores definidos, será analisada a possibilidade de correção do cromossomo durante a evolução e a possibilidade de utilizar uma solução conhecida durante a evolução. A tabela com as respectivas situações e seus desempenhos podem ser observados na Tabela 23.

Tabela 23 - Desempenho de Evoluções (Circuito Função Z no Modo Completo)

Correção do Cromossomo	Utilização de uma Solução Conhecida	Seleção	Mutação	Desempenho
Não	Não	0,7	0,9	44,51%
Não	Sim	0,79	0,01	82,81%
Sim	Não	0,7	0,03	66,44%
Sim	Sim	0,8	0,1	57,93%

Fonte: O autor, 2019.

Como observado na tabela anterior, o desempenho da evolução foi superior no caso sem a utilização de correção e com uma solução de partida. A curva de evolução que obteve tal resultado pode ser observado abaixo na Figura 78. É possível observar que não utilizar a correção do cromossomo resultou em um melhor desempenho. Tal comportamento pode representar que a correção limite a capacidade exploratória do algoritmo utilizando-se poucas

gerações. A discussão do impacto de tais funções faz parte dos trabalhos futuros possibilitados pela plataforma desenvolvida.

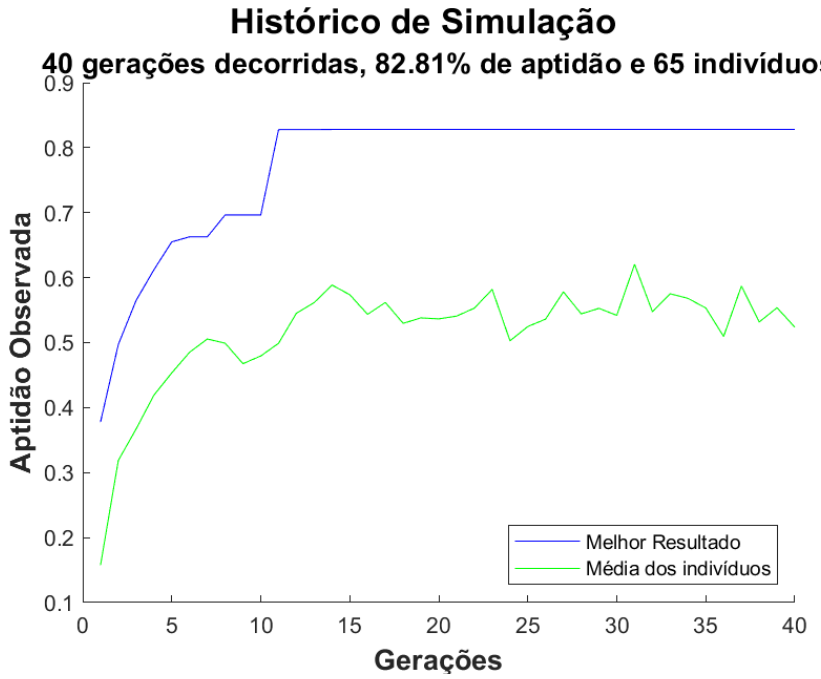

Figura 78 - Curva de Evolução Para Circuito Gerador de Função Z (Variação Completa de Componentes)

Fonte: O autor, 2019.

A topologia evoluída pelo processo de evolução pode ser observada abaixo na Figura 79. Tal topologia pode ser comparada com a netlist salva ao final da evolução, apresentada abaixo na Figura 80.

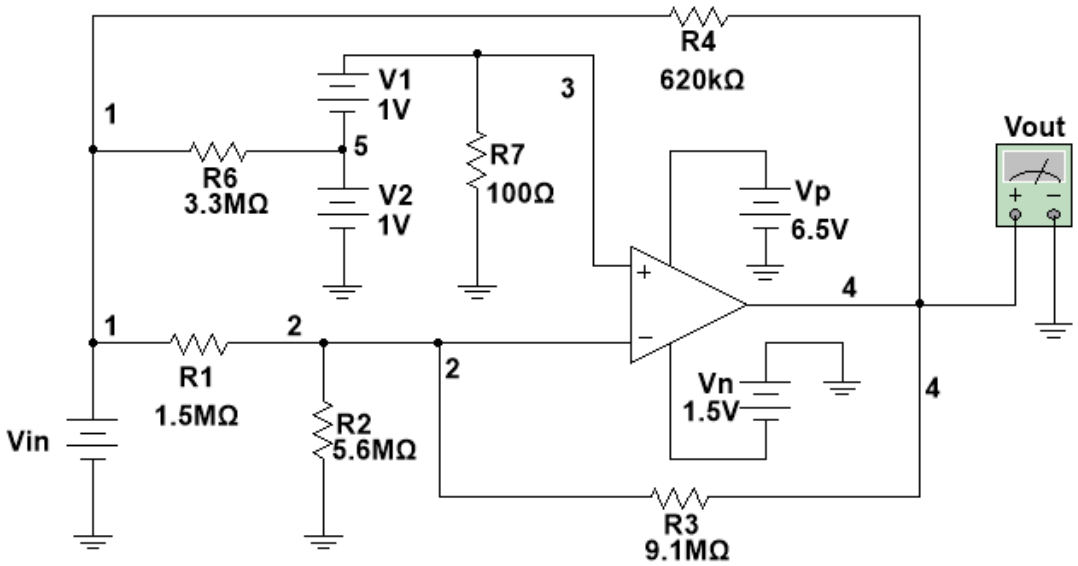

Figura 79 - Topologia Evoluída Para Circuito Gerador de Função Z (Variação Completa de Componentes)

Fonte: O autor, 2019.

```
CKT_FuncaoZ_Completa_SemCorr_ComMelhor_Sint_5e5_Resistores2

*
R1 1 2 1.500000E+06
R2 2 0 5.600000E+06
R3 2 4 9.100000E+06
R4 1 4 6.200000E+05
XAmpOp1 3 2 4 Vp1 Vn1 OPAMP2
*Fontes Amp Op
VpXAmpOp1  Vp1 0 DC   +6.500000e+00V
VnXAmpOp1  Vn1 0 DC   -1.500000e+00V
R5 5 5 2.400000E+04
V1 3 5 dc 1V
V2 5 0 dc 1V
R6 5 1 3.300000E+06
R7 3 0 1.000000E+02


*--------------------------------

*O modo de evolução realizado é o de variação total de componentes.
*Circuito variável utilizado: 10 1 1 2 0 25 2 0 1 2 0 2 95 2 0 1 2 4
1 120 2 4 1 1 4 4 99 3 4 2 3 2 4 24 13 3 1 5 5 1 48 8 0 3 3 5 4 1 0
10 3 5 0 4 1 0 16 1 5 1 2 65 3 5 1 3 0 1 1 20 1 4
*Tipos de componentes: RAV

*O circuito possui 10 Componentes: 7 Resistores, 0 Mosfets, 0 BJTs,
1 AmpOps, 2 Fontes DC e 0 Capacitores.

*--------------------------------
```

Figura 80 - Netlist Evoluída Para Circuito Gerador de Função Z (Variação Completa de Componentes)
Fonte: O autor, 2019.

Tal topologia gerou uma saída com 83% de compatibilidade com a saída desejada. A comparação entre tais formas de onda pode ser observada abaixo na Figura 81. Tal resposta, apesar de não exata comparada com a saída desejada, apresenta uma resposta bem próxima a desejada pela evolução. A alteração de parâmetros, como o número de gerações e indivíduos de sintonia e/ou evolução, pode resultar em melhores desempenhos para esse e outros circuitos analisados. O objetivo deste trabalho, entretanto, é possibilitar a ferramenta para trabalhos futuros que verifiquem a melhor combinação de parâmetros para cada caso estudado.

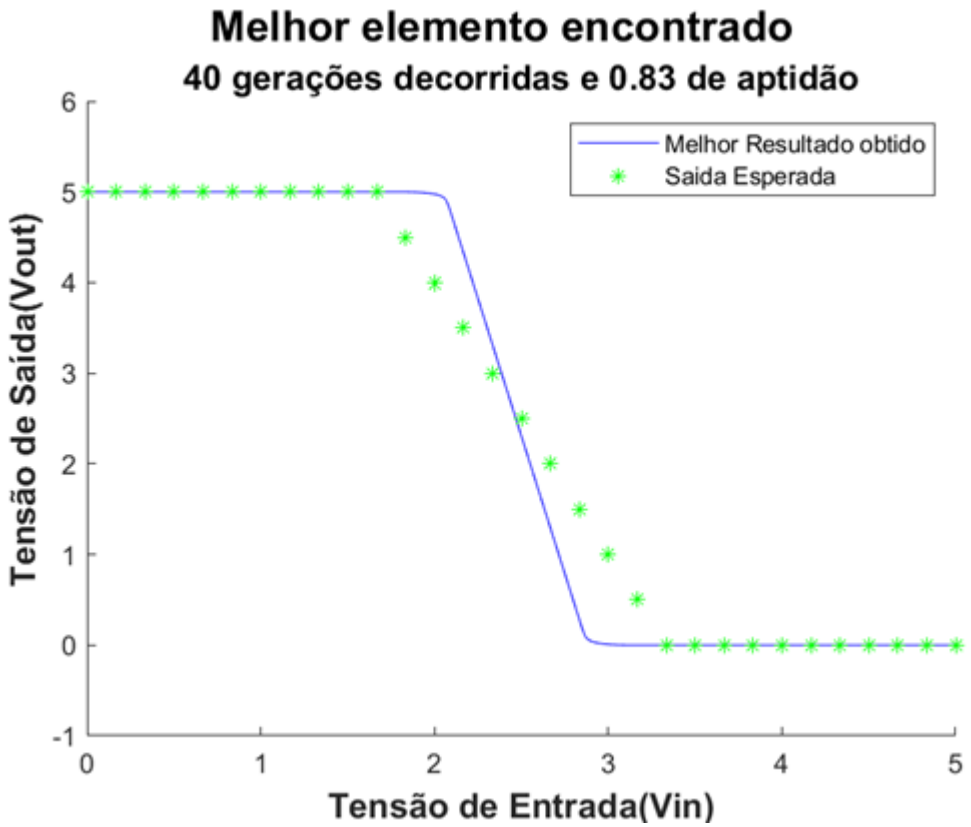

Figura 81 - Resposta Evoluída Para Circuito Gerador de Função Z (Variação Completa de Componentes)

Fonte: O autor, 2019.

4.4. Filtro Passa Baixa

Após a realização das evoluções para os circuitos de tensão contínua, o algoritmo é utilizado para evoluir circuitos no domínio da frequência. O primeiro circuito a ser evoluído dessa maneira é um filtro passa baixa. Esse circuito é muito utilizado em aplicações que se deseja remover ruídos que interfiram nos sinais analisados. Dessa forma, é estabelecida uma faixa de frequência utilizada no estudo e sinais acima desse valor são descartadas. A topologia básica de um filtro passa baixa pode ser observada abaixo na Figura 82.

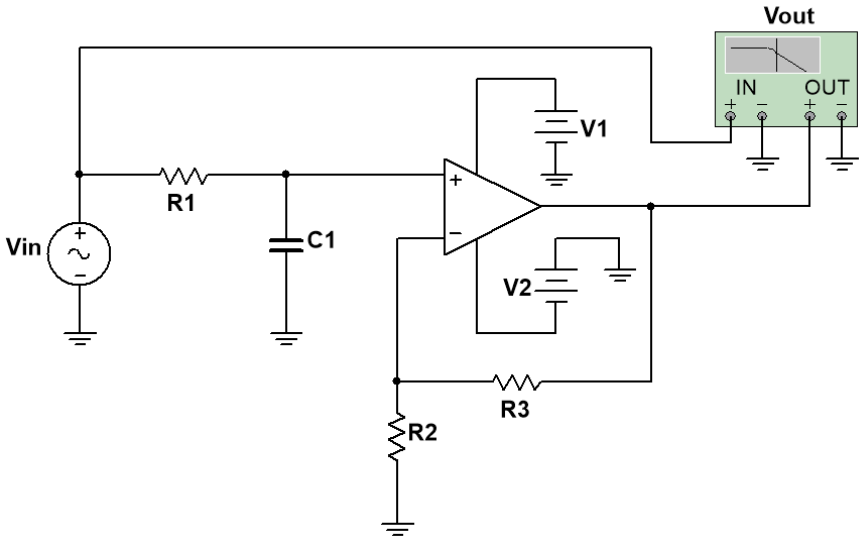

Figura 82 - Circuito de um Filtro Passa Baixa

Fonte: O autor, 2019.

4.4.1. Evolução por Variação de Valores

O primeiro modo de evolução consiste na evolução de valores da topologia básica para o filtro passa baixa. Utilizando a equação do espaço de busca, definida na Seção 3.6.1, determina-se um espaço de busca igual a 1.729.081 variações.

Do mesmo modo que nas evoluções realizadas anteriormente, a sintonia de valores inicial foi realizada para diferentes pares de valores de gerações e de indivíduos. A partir do desempenho destes pares é possível observar qual combinação apresenta o melhor desempenho para o processo de evolução. Os pares de valores utilizados durante a sintonia e os respectivos desempenhos obtidos podem ser observados na Tabela 24.

Tabela 24 - Sintonia de Valores (Passa Baixa no Modo Variação de Valores)

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,6	0,12	96,85%
7	5	0,1	0,11	84,11%
10	5	0,8	0,03	96,80%

Fonte: O autor, 2019.

Conforme observado na tabela anterior, o algoritmo obteve melhor desempenho, aproximadamente de 97% compatibilidade, com a resposta desejada utilizando cinco gerações de cinco indivíduos durante o processo de sintonia. O processo de sintonia e a curva de evolução podem ser observados na Figura 83 e na Figura 84, respectivamente.

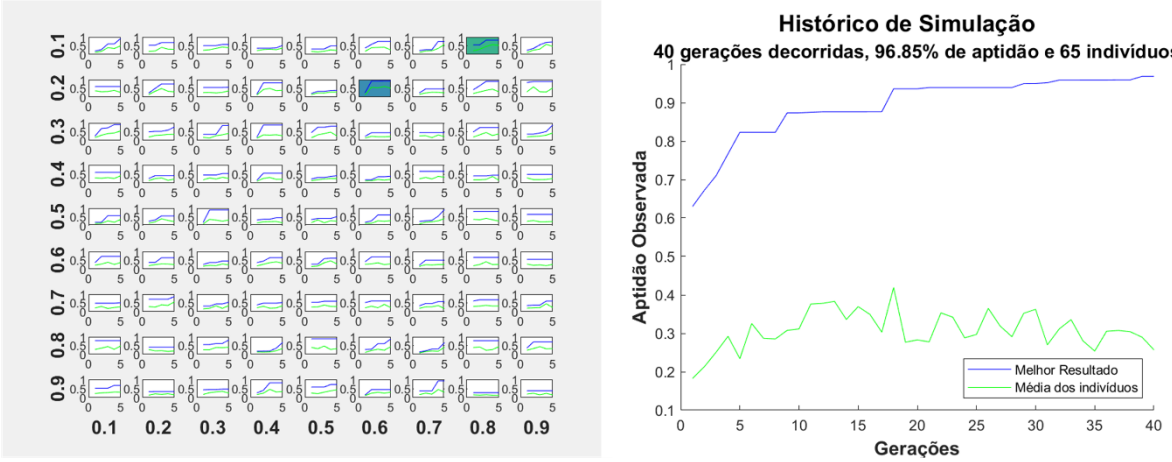

Figura 83 - Evolução Observada Após Sintonia Utilizando 5 Gerações de 5 Indivíduos

Fonte: O autor, 2019.

Entretanto, apesar de uma compatibilidade de 96,85% com a saída desejada, a resposta observada na saída evoluída não foi satisfatória. Esse resultado, observado na Figura 84, é explicado pelo fato de o LTSpice utilizar uma amostragem variável cujo passo é menor em locais com maiores variações do sinal. Dessa forma, a saída comparada possui uma maior amostragem em frequências com maior variação, dando prioridade a tais regiões.

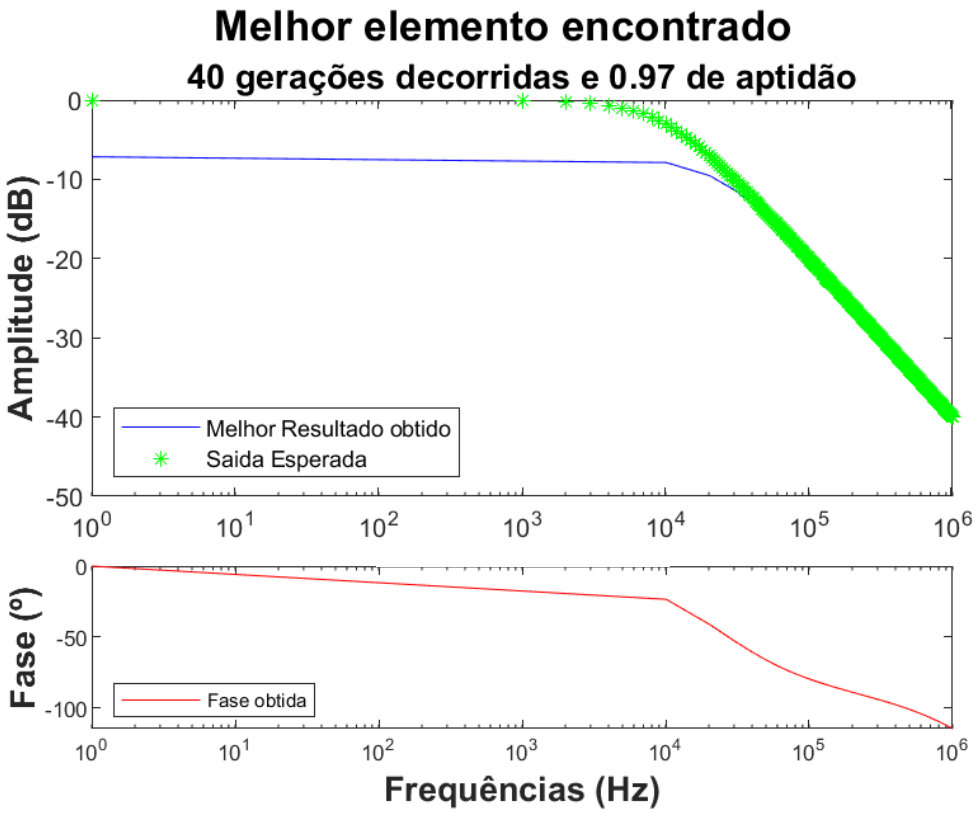

Figura 84 - Saída Não Satisfatória Apesar De Aptidão Elevada

Fonte: O autor, 2019.

Com o objetivo de evitar este tipo de erro novamente, foi utilizada, nesta evolução e nas demais do domínio da frequência, uma distribuição maior para pontos nas regiões de frequência com menor variação. Dessa forma, o cálculo da aptidão não prioriza uma região em relação a outra, considerando cada região da forma de onda igualmente. Para a evolução do circuito passa faixa em questão, foi utilizada a mesma quantidade de pontos para a parte monótona da curva e para a parte que possui maior variação na magnitude.

Realizando-se novamente a evolução, utilizando a metodologia anterior, foi executada a sintonia para diferentes pares de valores de gerações e de indivíduos. Tais pares de valores e seus respectivos desempenhos podem ser observados na Tabela 25.

Tabela 25 - Resultados Observados Após Sintonia de Valores

Sintonia		Resultados		
Gerações	Indivíduos	Seleção	Mutação	Desempenho
5	5	0,1	0,1	99,40%
7	5	0,6	0,04	98,68%
10	5	0,9	0,03	96,94%

Conforme observado na tabela anterior, o algoritmo obteve um desempenho de aproximadamente de 99% compatibilidade com a resposta desejada. Tal desempenho foi obtido realizando o processo de sintonia de valores utilizando cinco gerações de cinco indivíduos. Após a realização deste processo, obteve-se os valores de 0,1 para a mutação (Horizontal) e 0,1 para a seleção (Vertical). A tela de resultado do processo de sintonia pode ser observada abaixo na Figura 85.

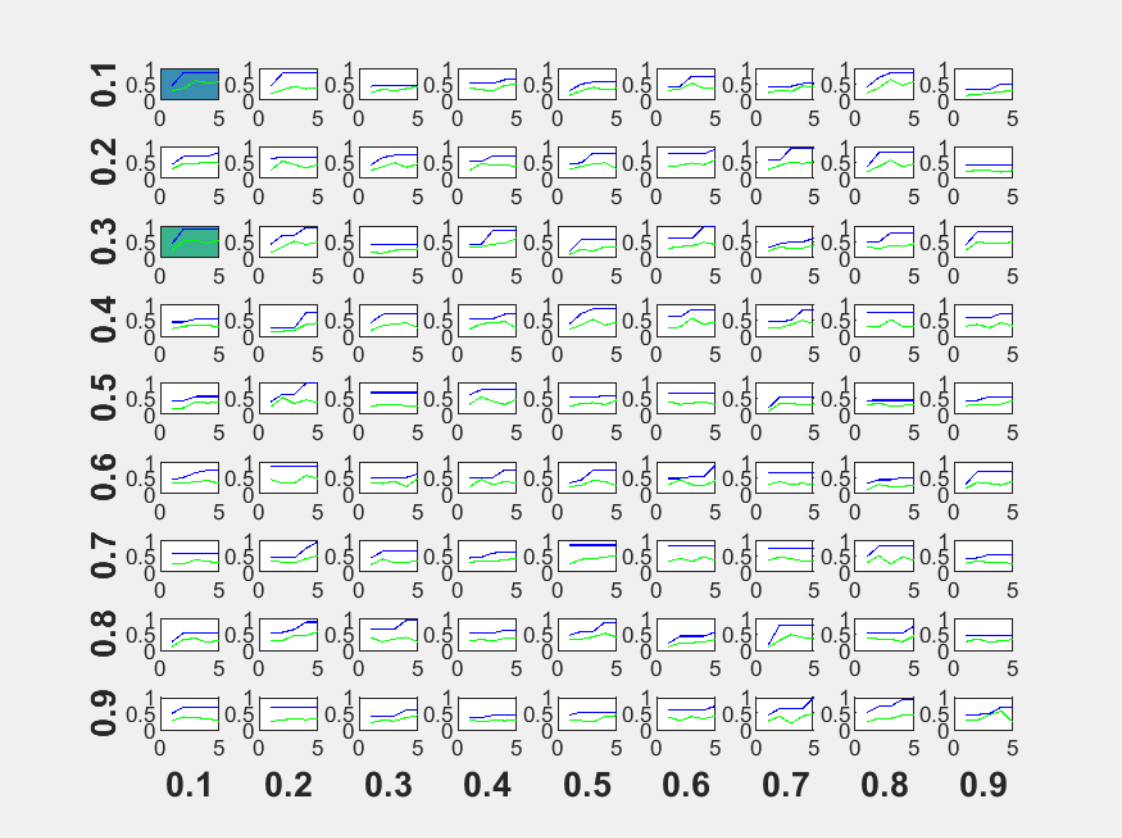

Figura 85 - Tela de Sintonia de Valores Para Evolução do Filtro Passa Baixa no Modo Variação de Valores
Fonte: O autor, 2019.

Uma vez determinados os fatores de seleção e de mutação, a evolução é realizada utilizando a quantidade de gerações e indivíduos determinados anteriormente. Para as evoluções do filtro passa baixa no modo de valores foram utilizadas quarenta gerações com sessenta e cinco indivíduos. A curva de evolução e a resposta observada pela melhor solução podem ser observados na Figura 86 e na Figura 87, respectivamente.

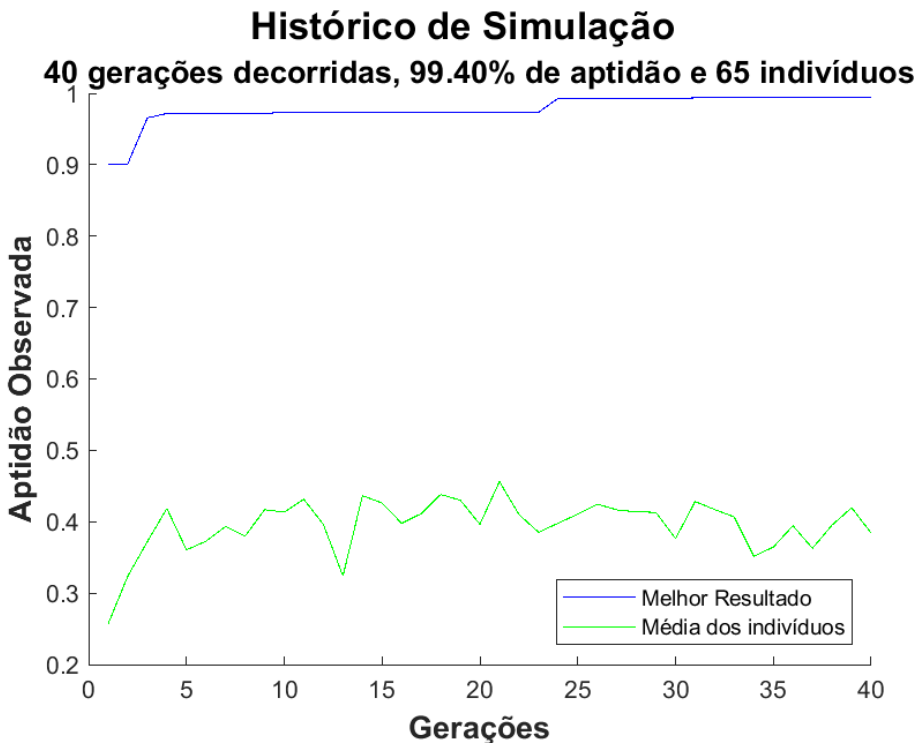

Figura 86 - Curva de Evolução do Filtro Passa Baixa no Modo Variação de Valores

Fonte: O autor, 2019.

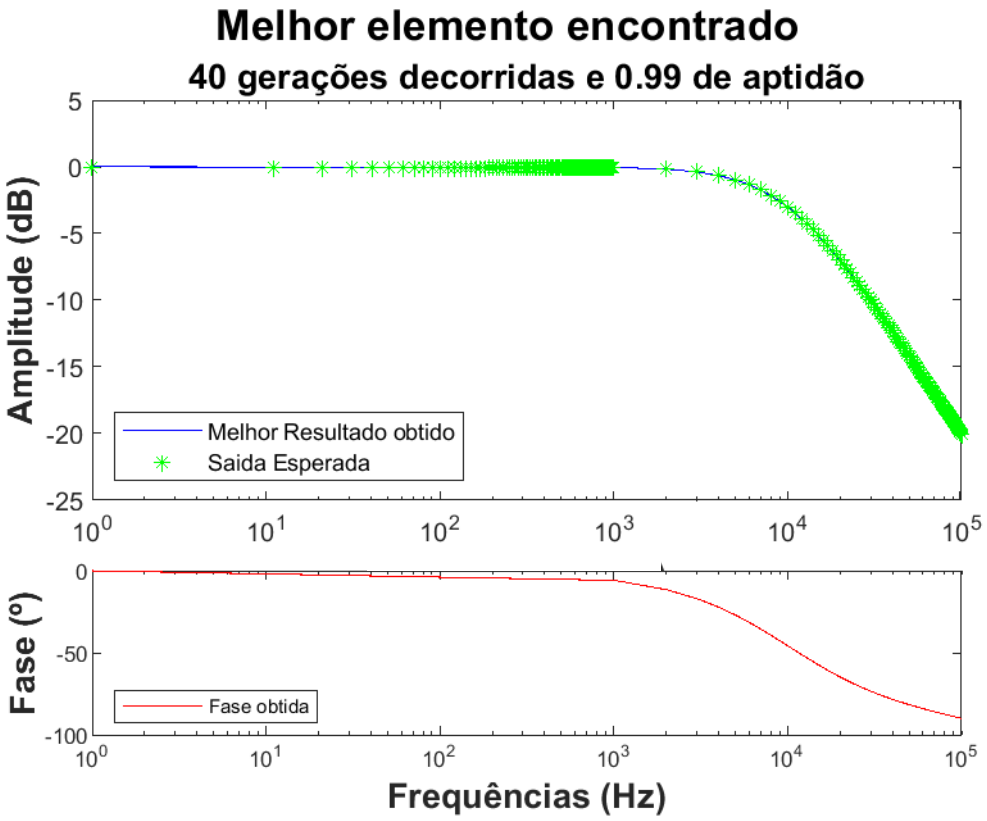

Figura 87 - Resposta Observada Para Evolução do Filtro Passa Baixa no Modo Variação de Valores

Fonte: O autor, 2019.

Após o processo de evolução, é determinado uma topologia ao usuário; cuja aptidão é de 99% de compatibilidade com a resposta desejada. A topologia obtida após esse processo de evolução pode ser observada na Figura 88.

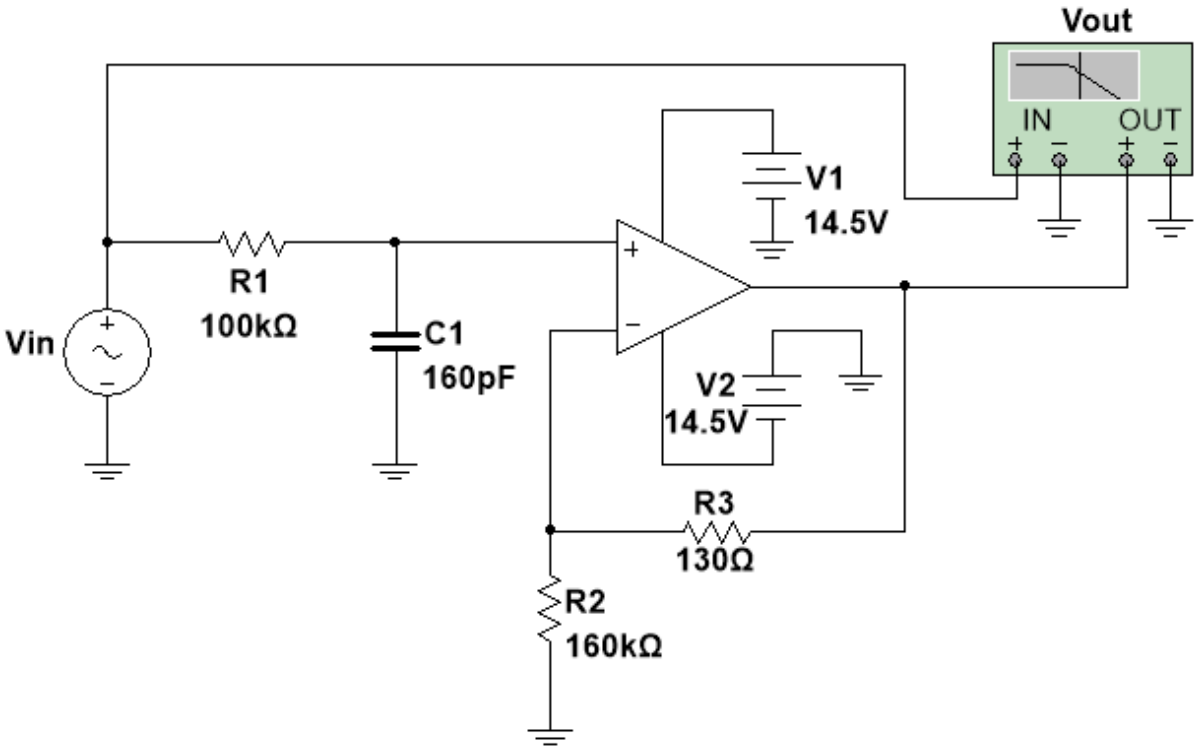

Figura 88 - Topologia Evoluída Para Filtro Passa Baixa no Modo Variação de Valores
Fonte: O autor, 2019.

Após o processo de evolução também se obteve o relatório do processo realizado. Nesse relatório pode-se observar a evolução dos valores no decorrer da simulação e observar variáveis de interesse como o tempo necessário para a evolução. Tais informações podem ser observadas abaixo na Tabela 26.

Tabela 26 - Relatório de Evolução Para Filtro Passa Baixa no Modo Variação de Valores

Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Melhor Resultado	0.9010	0.9010	0.9655	0.9718	0.9718	0.9718	0.9718	0.9718	0.9718	0.9718	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9735	0.9929	0.9929	0.9929	0.9929	0.9929	0.9929	0.9929	0.9936	0.9936	0.9936	0.9936	0.9936	0.9936	0.9937	0.9937	0.9940	0.9940
Média dos Elementos	0.2569	0.3235	0.3720	0.4185	0.3607	0.3726	0.3932	0.3800	0.4168	0.4135	0.4314	0.3963	0.3241	0.4364	0.4261	0.3979	0.4113	0.4382	0.4301	0.3964	0.4561	0.4100	0.3852	0.3980	0.4105	0.4245	0.4167	0.4138	0.4131	0.3763	0.4282	0.4171	0.4068	0.3516	0.3645	0.3944	0.3628	0.3954	0.4195	0.3842
Erros por Geração	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erro 1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 6		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Erro 7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
População:		65 indivíduos																																						
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22	G23	G24	G25	G26	G27	G28	G29	G30	G31	G32	G33	G34	G35	G36	G37	G38	G39	G40
Erros de Simulação Críticos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Porcentagem de Críticos	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo de Sintonia:	1470,024093 Segundos																																							
	24,50040155 Minutos																																							
	0,408340026 Horas																																							
Tempo de Evolução	1640,963953 Segundos																																							
	27,34939921 Minutos																																							
	0,45582332 Horas																																							

4.4.2. Evolução por Adição de Componentes

Após a realização da evolução de uma topologia pelo modo variação de valores, simularemos uma situação de falha no circuito evoluído. Essa simulação testará a plataforma buscando uma solução para a falha através do modo Adição de Componentes.

A falha simulada será a queima de um de seus resistores, mais precisamente o resistor com valor de 160 kΩ. Após a queima desse resistor, a conexão dos nós desse componente é considerada inexistente. A partir do circuito resultante, o algoritmo adicionará componentes a topologia e avaliará tal adição na resposta observada.

Do mesmo modo que nas evoluções realizadas anteriormente, o processo de evolução foi realizado utilizando-se a sintonia com o par de valores que obteve o melhor desempenho na evolução anterior, ou seja, cinco gerações com cinco indivíduos. Definidas as variáveis de sintonia, realizaremos o desempenho da evolução com e sem a correção do cromossomo durante a evolução. A tabela comparativa com os desempenhos de cada caso pode ser observada abaixo na Tabela 27.

Tabela 27 - Desempenho de Evolução do Filtro Passa Baixa (Modo Adição de Componentes)

Correção do Cromossomo	Seleção	Mutação	Desempenho
Não	0,7	0,2	99,89%
Sim	0,4	0,01	99,91%

Fonte: O autor, 2019.

Como observado na tabela acima, a correção do cromossomo interfere, mesmo que pouco, no desempenho observado. A topologia resultante dessa evolução pode ser observada em seguida na Figura 89. A resposta resultante também pode ser observada na Figura 90.

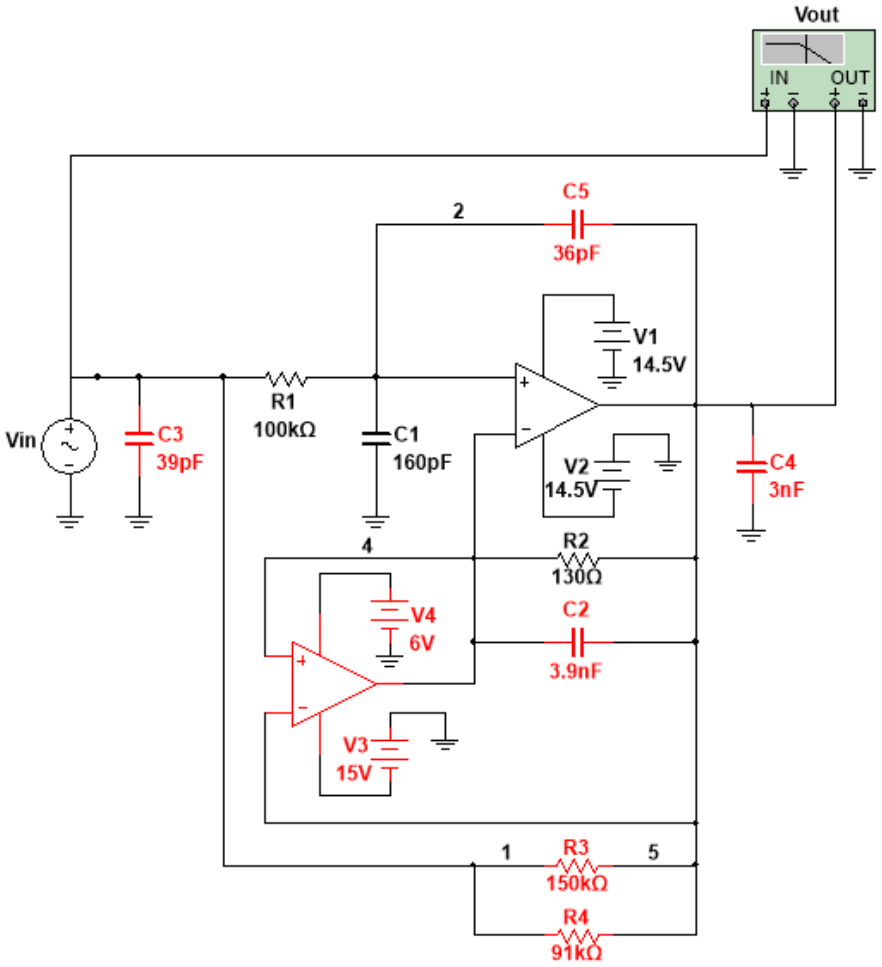

Figura 89 - Topologia Evoluída Para Filtro Passa Baixa (Modo Adição de Componentes)

Fonte: O autor, 2019.

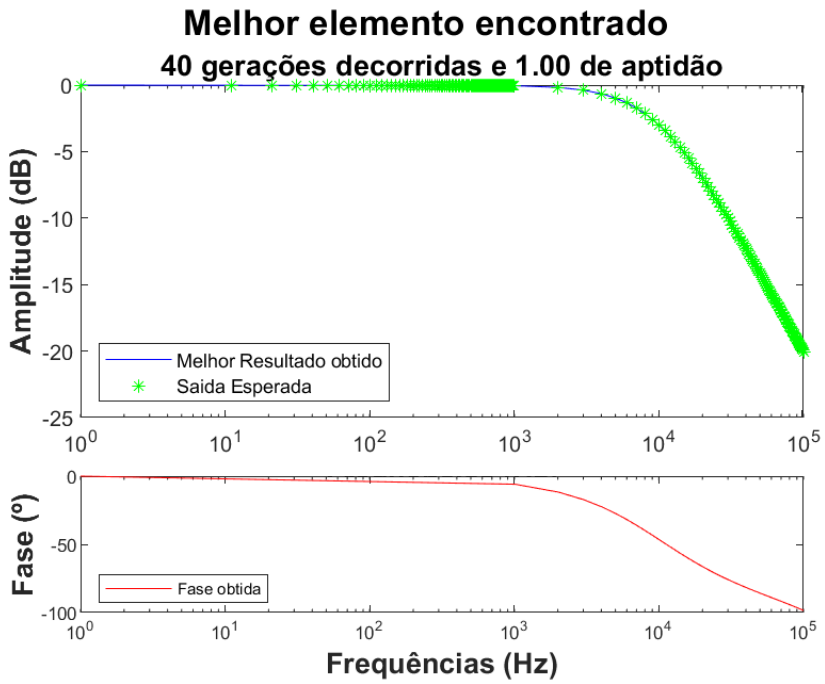

Figura 90 - Resposta Evoluída Para Filtro Passa Baixa no Modo Adição de Componentes

Fonte: O autor, 2019.

4.4.3. Evolução por Alteração Completa de Componentes

Após evoluir o circuito por adição de componentes, realizaremos a evolução no modo Alteração Completa de Componentes. Nessa evolução será evoluída toda a topologia em busca de uma solução adequada.

A saída objetivo dessa evolução será um filtro passa baixa com frequência de corte diferente da utilizada anteriormente, passando da frequência de corte de 1000 Hz para a frequência de 3000 Hz. Observando a eficiência da plataforma nos casos anteriores, a evolução não usará solução inicial. A curva de evolução observada pela evolução pode ser vista abaixo na Figura 91. A topologia evoluída também pode ser observada abaixo na Figura 92.

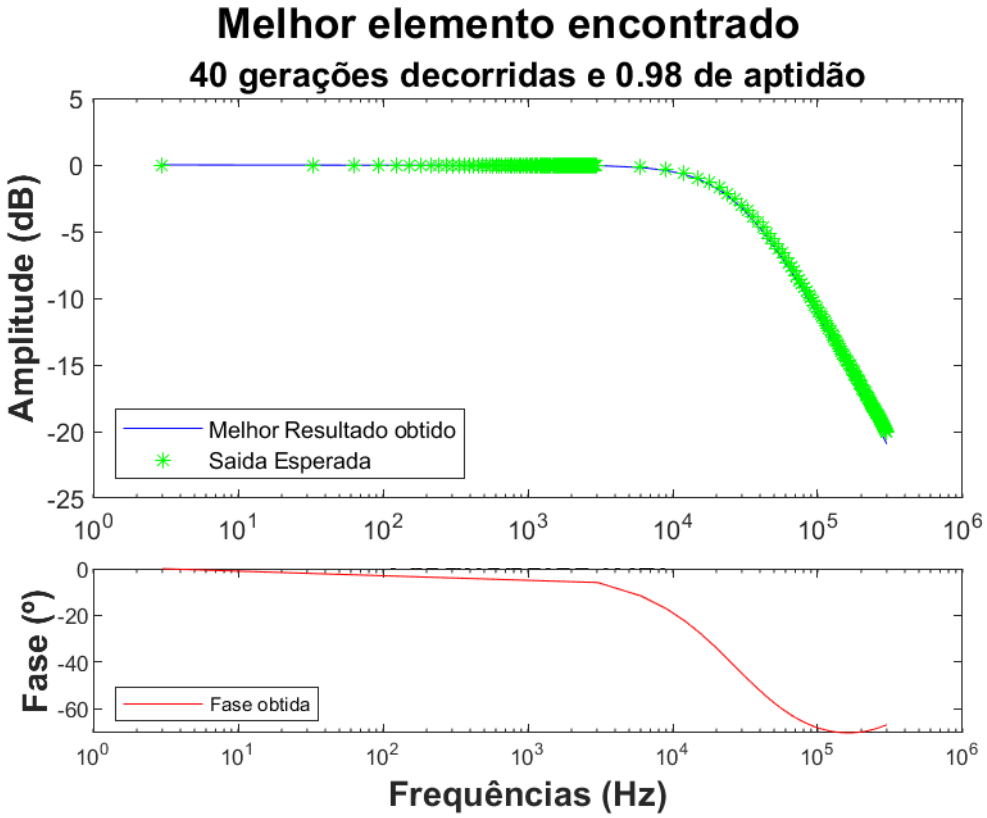

Figura 91 - Resposta Observada Para Filtro Passa Baixa (Modo Alteração Completa de Componentes)

Fonte: O autor, 2019.

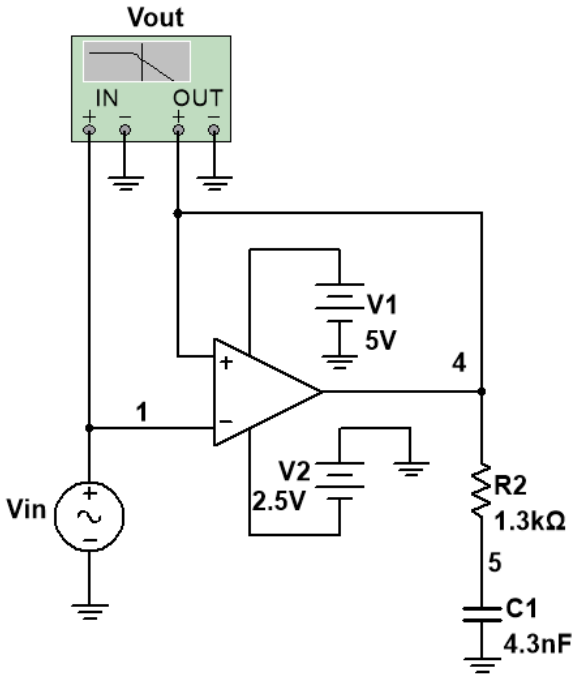

Figura 92 - Topologia Evoluída Para Filtro Passa Baixa (Modo Alteração Completa de Componentes)
Fonte: O autor, 2019.

4.5. Divisor Tensão Com Duplo Objetivo

Comprovada a eficiência da plataforma ao evoluir um único objetivo, realiza-se a evolução de evoluções que considerem dois objetivos; a compatibilidade com a saída e a potência consumida.

Para essa evolução serão mantidos os critérios e a metodologia utilizados nas evoluções anteriores. Dessa forma, realiza-se o processo de sintonia assim como nas demais evoluções. Essa comparação visa observar se o comportamento observado nas evoluções com apenas um objetivo é mantido na evolução com dois objetivos. Uma vez que a evolução multiobjetivo utiliza a solução semente como parâmetro de comparação da potência, as comparações são realizadas verificando-se o impacto da correção do cromossomo durante o processo de evolução. A tabela correspondente de desempenhos pode ser observada na Tabela 28.

Tabela 28 - Sintonia de Valores Para Evolução Com Duplo Objetivo

Correção do Cromossomo	Sintonia		Resultados		
	Gerações	Indivíduos	Seleção	Mutação	Desempenho
Não	5	5	0,2	0,1	93,67%
	7	5	0,6	0,3	93,67%
Sim	5	5	0,9	0,8	93,65%
	7	5	0,7	0,18	93,58%

Fonte: O autor, 2019.

Como observado na tabela anterior, o valor do desempenho indica uma saída próxima, mas não exata, comparando-se com a saída desejada. Entretanto, ao observar a curva observada na saída do melhor elemento constata-se que a plataforma obteve a saída desejada.

A saída observada mostra uma compatibilidade de 100% com a saída, como mostrado na Figura 93. Dessa forma, esse valor inferior do desempenho é devido à plataforma buscar minimizar ao máximo a potência consumida pelo circuito.

Outra constatação possível é que a correção do cromossomo não impactou significativamente no desempenho observado. Tal comportamento pode justificar-se devido à baixa complexidade do circuito evoluído. O impacto da correção do cromossomo em circuitos mais complexos compõe os trabalhos futuros possibilitados pela plataforma desenvolvida.

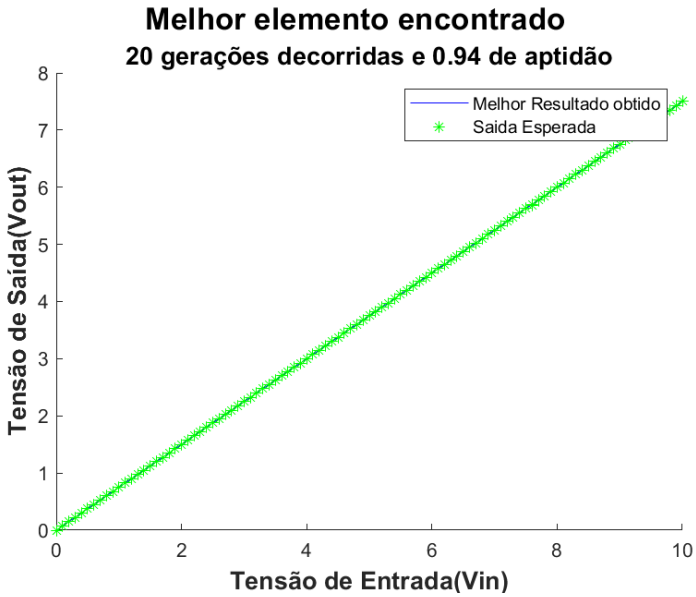

Figura 93 - Saída Observada Para a Evolução Com Duplo Objetivo

Fonte: O autor, 2019.

Ao observar a topologia evoluída, mostrada na Figura 94, constata-se a utilização de componentes com maior valor que na evolução realizada para apenas um objetivo. Dessa forma, a evolução com o objetivo de minimizar a potência consumida foi realizada de modo satisfatório.

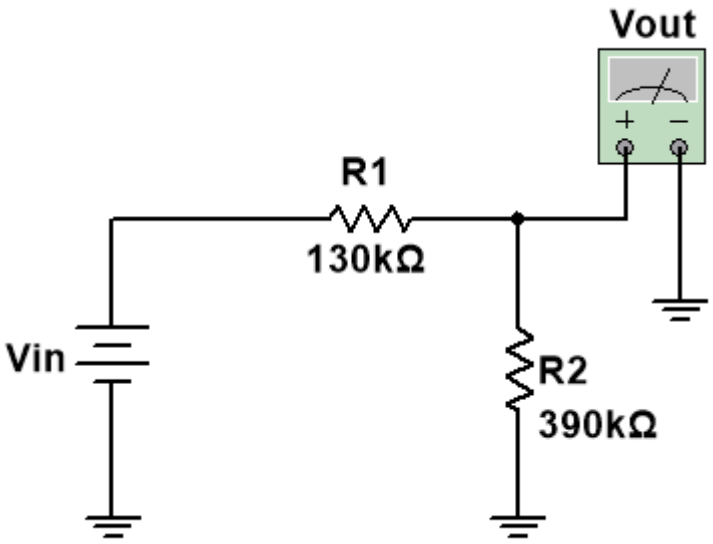

Figura 94 - Topologia Evoluída Após Evolução Com Duplo Objetivo

Fonte: O autor, 2019.

A curva de evolução para a aquisição dessa topologia pode ser observada abaixo na Figura 95.

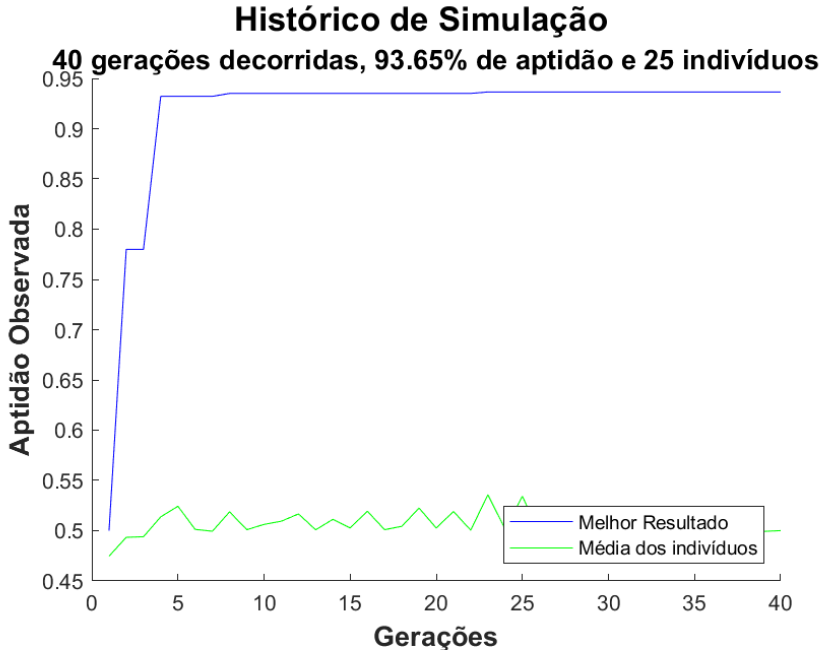

Figura 95 - Curva de Evolução Para Divisor De Tensão Com Multiobjetivo

Fonte: O autor, 2019.

4.6. Amplificador Com Duplo Objetivo

Após evoluir satisfatoriamente o circuito divisor de tensão, realiza-se a evolução do circuito amplificador utilizando a avaliação multiobjetivo. Tal avaliação visa observar a função em circuitos mais complexos, evoluindo inclusive as tensões de alimentação do amplificador em busca de uma topologia que satisfaça a saída com o mínimo de potência consumida.

Do mesmo modo que os estudos anteriores, o processo de sintonia é realizado e, posteriormente, a evolução é realizada com o melhor par de variáveis obtido. Ao final do processo de evolução uma resposta com 92% de aptidão foi obtida pela plataforma. Tal curva de evolução pode ser observada abaixo na Figura 96.

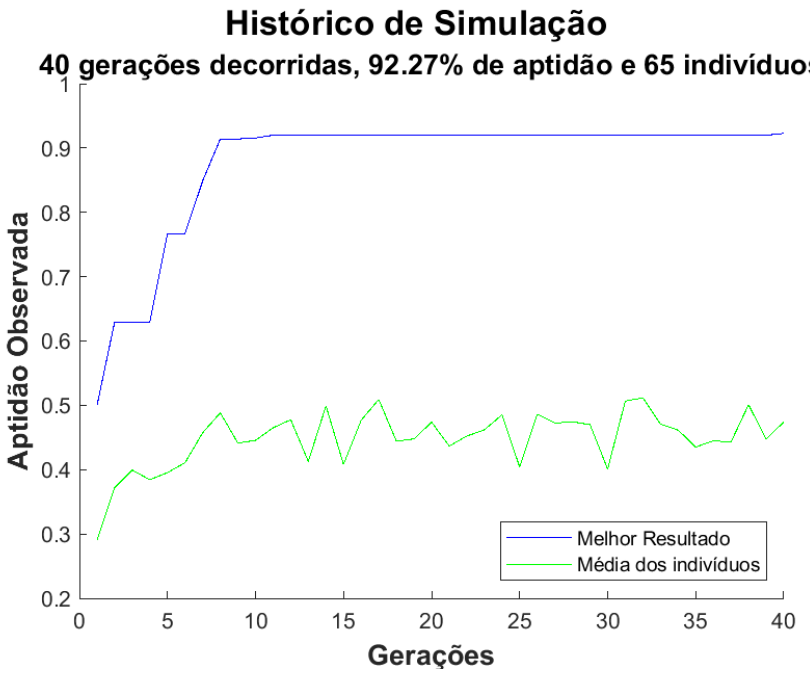

Figura 96 - Curva de Evolução Para Amplificador Com Multiobjetivo

Fonte: O autor, 2019.

Entretanto, ao observar a saída obtida observou-se que a compatibilidade não estava de acordo com o esperado, como observado abaixo na Figura 97. Tal resposta expôs a necessidade de alteração das regras de avaliação fuzzy utilizadas.

Em virtude da estrutura de criação da plataforma, tal atualização pode ser realizada diretamente no arquivo que contêm as regras fuzzy utilizadas. Com os arquivos de regras atualizado, a plataforma utilizará tais regras durante a evolução e nenhuma alteração no código fonte da mesma será necessária.

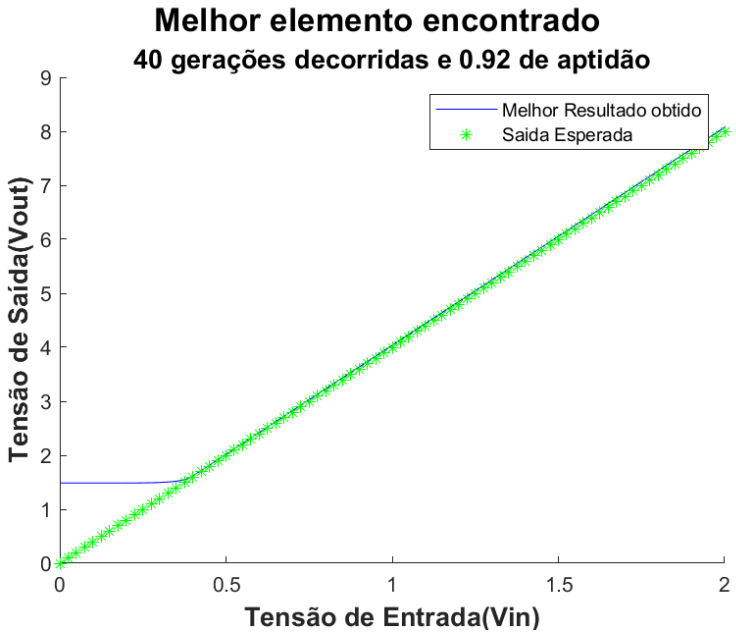

Figura 97 - Resposta Observada Para Amplificador Com Multiobjetivo

Fonte: O autor, 2019.

Tal resposta ocorreu devido a configuração utilizada pela avaliação fuzzy priorizar uma maior economia de potência ao invés de uma melhor compatibilidade entre a saída observada e a saída desejada. Após a atualização das regras fuzzy, facilitada pela estrutura utilizada pela plataforma, a execução foi novamente realizada. Tal execução resultou na resposta observada na Figura 98.

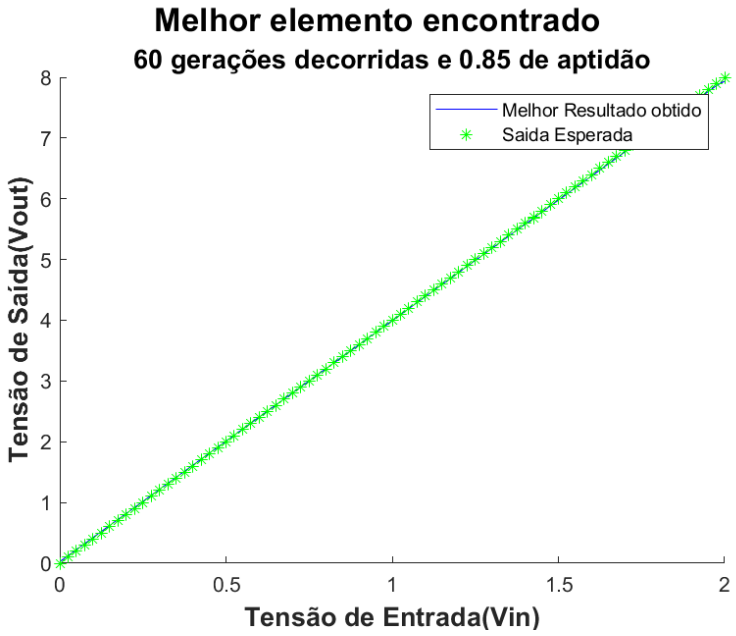

Figura 98 - Resposta Atualizada Para Amplificador Com Multiobjetivo

Fonte: O autor, 2019.

Pode-se observar que a alteração das regras fuzzy resultaram na modificação do desempenho observado ao final da evolução. Apesar de apresentar um valor menor para o desempenho, a evolução obteve uma saída exata em comparação com a desejada. A execução também obteve valores de resistores próximos a maior faixa disponível (MΩ), como esperado para uma evolução multiobjetivo que considere a potência consumida. A topologia evoluída pode ser vista na Figura 99.

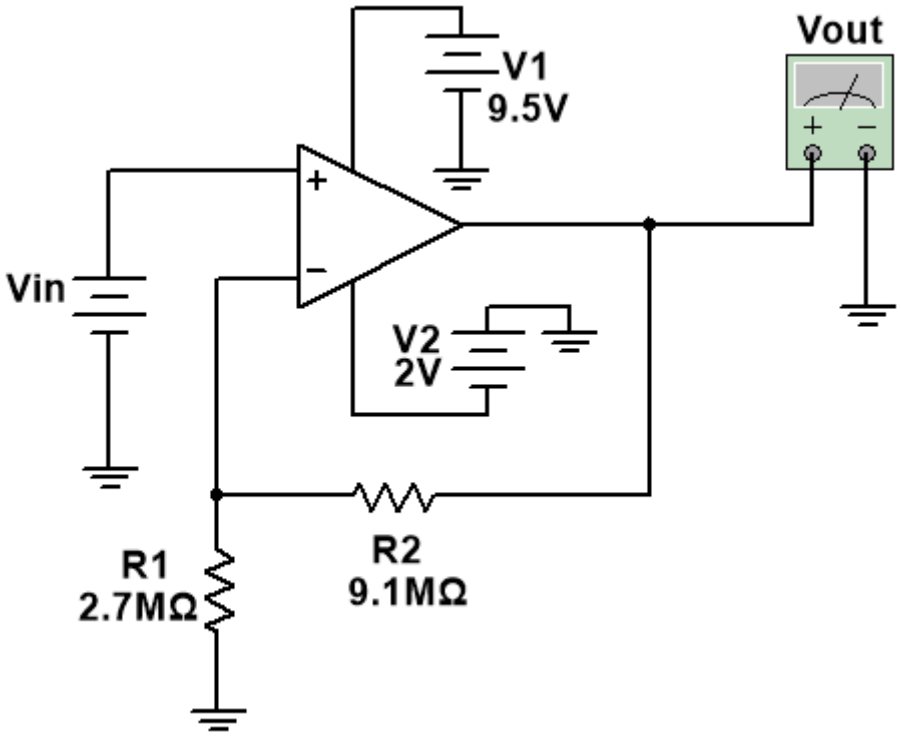

Figura 99 - Topologia Evoluída Para Amplificador Com Multiobjetivo
Fonte: O autor, 2019.

Verifica-se, também, que os valores de alimentação para o amplificador foram evoluídos como esperado. Sendo a saída um sinal com variação de 0V a 8V, as alimentações foram evoluídas na fronteira de não saturação do sinal sem comprometer a potência consumida.

Ressalta-se que, para a evolução realizada, foi dada prioridade à confirmação do correto funcionamento da plataforma ao invés da melhor escolha de parâmetros. O exemplo realizado buscou mostrar a fácil alteração dos critérios fuzzy utilizados e a obtenção de uma resposta apropriada para um número fixo de gerações.

Uma melhor determinação do impacto de cada parâmetro e das regras fuzzy apropriadas compõe os trabalhos futuros que foram possibilitados pelo desenvolvimento da plataforma apresentada no decorrer deste trabalho.

5. CONSIDERAÇÕES FINAIS

5.1.Conclusões

Com base nos quatorze estudos de caso apresentados, conclui-se que a plataforma foi desenvolvida de acordo com os objetivos definidos inicialmente. Os resultados alcançados para cada um dos modos de operação existentes, com diferentes circuitos de complexidades e tipos de análises distintas, indicam boa flexibilidade para os mais diversos experimentos.

A plataforma concebida permite ao projetista uma rápida e fácil mudança na especificação do circuito a ser evoluído sem alterações complexas no código. A alteração das topologias analisadas, as entradas utilizadas e o tipo de entrada podem ser facilmente modificadas a partir das informações passadas à plataforma. É papel da plataforma a criação dos arquivos necessários para a comunicação com o simulador e dos arquivos resultados dessa evolução, sem necessidade da interferência pelo usuário.

A plataforma também permite uma rápida customização da evolução em utilizar uma solução semente e em realizar correções no cromossomo durante sua avaliação. Tais alterações simplificadas permitem ao projetista analisar o impacto dessas ações na resposta evoluída e simplificam a escolha do comportamento a ser selecionado para uma evolução e para circuitos similares.

Outra função que facilita a evolução de topologias é sua capacidade de sintonia de valores. Através dessa função, ativada ou desativada pelo usuário, a plataforma avalia o comportamento da evolução para diferentes combinações das variáveis de seleção e mutação de indivíduos, utilizando a que obtiver a melhor média da população para a evolução. Isso diminui a chance de ocorrer casos de a evolução estagnar em valores não satisfatórios e, consequentemente, a necessidade de o usuário realizar sucessivas evoluções completas até obter uma combinação de valores que resulte em uma resposta satisfatória.

A estagnação poderia ser mitigada com parâmetros adaptativos de crossover e, principalmente, de mutação. Entretanto, essa função de sintonia de valores facilita a comparação de parâmetros. A escolha do processo de sintonia para as taxas de seleção normalizada e a taxa de mutação buscou um balanceamento entre o caráter determinístico e o caráter exploratório das evoluções da plataforma, respectivamente.

Com o objetivo de otimizar o processo de evolução, diminuindo o tempo de processamento, todos os estudos de caso foram realizados utilizando um disco virtual na memória RAM do computador. Esse tipo de disco é denominado “Ram Disk” e otimiza a velocidade de acesso e escrita de arquivos durante os acessos. Essa otimização ocorre porque

tal disco trabalha com a velocidade da respectiva memória RAM utilizada, que é consideravelmente maior que as respectivas velocidades em um disco rígido convencional. É importante ressaltar que a utilização desse tipo de disco zerou os casos de erro em que não é possível o simulador abrir o arquivo salvo em disco (Erro 2). Dessa forma, atribui-se a ocorrência desse erro à demora ocasional de resposta pelo disco rígido durante as evoluções.

Entretanto, a utilização de discos virtualizados na memória RAM possuem riscos intrínsecos ao seu funcionamento. Devido à estrutura de funcionamento desse tipo de memória, caso ocorra uma queda de luz momentânea durante sua utilização toda a informação armazenada nela é perdida.

Apesar de a plataforma possibilitar a simulação de topologias de modo encadeado, quedas de luz acarretaram constantes perdas de arquivos, arquivos estes de evoluções em curso e de evoluções passadas devido ao uso desse tipo de armazenamento. Conclui-se que apesar da melhora observada no tempo de evolução, melhora de aproximadamente 20%, deve-se utilizar uma rotina de backup desses arquivos para evitar perdas. Alternativamente um Nobreak pode efetivamente resolver esta questão de modo mais interessante.

Dessa forma, conclui-se que o trabalho desenvolvido contribuiu de forma satisfatória na evolução de topologias de circuitos eletrônicos, destacando-se a flexibilidade para a realização das mais diversas evoluções. Tal contribuição permite que evoluções possam ser realizadas de modo simplificado e claro para o projetista, possibilitando uma análise mais detalhada das variáveis no impacto de evoluções em trabalhos futuros de diversas aplicações.

Ressalta-se que o trabalho apresentado teve como foco principal desenvolver e apresentar uma plataforma customizável para a evolução de topologias de circuitos. Dessa forma, valores fixos de parâmetros são utilizados para facilitar uma análise do impacto de cada parâmetro no desempenho observado. Não foi o objetivo do trabalho determinar quais valores de parâmetros são mais adequados no processo de evolução de circuitos.

A escolha de regras e funções de pertinência para a avaliação fuzzy seguiu a lógica de escolha dos parâmetros da evolução mono-objetivo. Tais regras foram determinadas de modo heurístico a fim de demonstrar a incorporação e utilização pela plataforma desenvolvida. Tanto para evoluções mono-objetivo quanto para evoluções multiobjectivo, priorizou-se conceber uma plataforma de fácil utilização que possibilite a trabalhos futuros verificarem os melhores critérios a serem utilizados.

5.2.Trabalhos Futuros

Em relação aos trabalhos futuros possíveis, podem ser sugeridos estudos com o objetivo de identificar falhas de modo contínuo em circuitos. Através dessa identificação, seria possível realizar chamadas automáticas a plataforma desenvolvida e, consequentemente, a evolução com reparo automático do circuito a partir dessas falhas.

Outra ramificação possível é a utilização das topologias evoluídas em circuitos reconfiguráveis. Dessa forma, os resultados evoluídos pela plataforma seriam automaticamente carregados em tais circuitos e, consequentemente, estariam disponíveis para utilização de modo automático.

Com base no desempenho de discos virtualizados no tempo de evolução, 20% melhor que discos rígidos convencionais, outra possibilidade para aumentar ainda mais o desempenho seria a utilização de unidades de processamento gráfico (GPU – *Graphics Processing Unit*) para otimizar o processamento dessas evoluções. Tais unidades, disponíveis em placas de vídeo, podem diminuir o tempo necessário para realizar as evoluções estudadas nesse trabalho e nas evoluções de circuitos mais complexos.

A principal contribuição da plataforma desenvolvida é a possibilidade de alteração simplificada de cada parâmetro utilizado. Tal capacidade possibilita estudos futuros detalhados sobre o impacto de cada parâmetro na evolução de circuitos. O impacto de parâmetros também pode ser avaliado de acordo com o tipo, complexidade e entrada utilizados por cada circuito. Dessa forma, a plataforma em questão proporciona uma ampla ramificação de estudos futuros de forma descomplicada ao usuário.

REFERÊNCIAS

AMARAL, J. F. Síntese de Sistemas Fuzzy por Computação Evolucionária. Tese de Doutorado, Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, 2003.

AMARAL, J. L. M.; Sistemas Imunológicos Artificiais Aplicados À Detecção De Falhas, Tese de Doutorado, PUC/Rio, 2006.

AMARAL, J. L. M.; AMARAL, J. F. M.; TANSCHEIT, R.; Sistema Imunológico Artificial Para Detecção De Falhas Em Circuitos Analógicos Com Geração Automática De Detectores, XVI Congresso Brasileiro de Automática, 2006, Salvador.

BACELAR, Y. C.: Evolução De Circuitos Eletrônicos Analógicos Com Avaliação De Múltiplos Objetivos Baseada em Sistema Fuzzy. Dissertação de Mestrado, UERJ, 2017.

CARVALHO, T. M.: Síntese evolutiva de circuitos eletrônicos usando Plataforma Extrínseca. Projeto Final de Curso, UERJ, 2018.

COELHO, P.; AMARAL; J. F. M., GUIMARÃES; K.; BENTES, M.: Layout of Routers in Mesh Networks with Evolutionary Techniques. In Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019), pages 438-445. 2019 ISBN: 978-989-758-372-8. https://doi.org/10.5220/0007739204380445.

COELLO COELLO, C. A. Multi-objective evolutionary algorithms in real-world applications: some recent results and current challenges. Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences, Vol. 36 of the series Computational Methods in Applied Sciences pp 3-18, 2013.

COELLO COELLO, C. A. A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems, Volume 1, Issue 3, p. 269–308, 1999.

ENGELHARDT, M.; eCircuit Center; 2004; <www.ecircuitcenter.com/OpModels/V_Limit/opmodel3 V Limit>. Acesso em: 24 ago. 2019.

EREMIA, M.; LIU, CHEN-CHING; EDRIS, ABDEL-ATY. Artificial Intelligence and Computational Intelligence: A Challenge for Power System Engineers, 1072p, Wiley-IEEE Press, 2016.

FONSECA, A. C. X. S., Modelo Evolucionário baseado em Agregador Fuzzy para Avaliação de Múltiplos Objetivos, Dissertação de Mestrado, UERJ, 2017.

GACTO, M. J.; ALCALÁ, R.; HERRERA, F. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures, Information Sciences, Special Issue on Interpretable Fuzzy Systems, Vol. 181, Issue 20, p.4340–4360, 2011.

GOLDBERG, D. E. Genetic algorithms in search, optimization, and machine learning. Reading, Addison-Wesley Professional, 1989.

GREENWOOD, G. W.; TYRREL, A. M.; "Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems". IEEE Press Series on Computational Intelligence. David B. Fogel Series Editor. 2007. ISBN: 978-0-471-71977-9.

HADDOW P. C.; TYRRELL, A. M.: Evolvable Hardware Challenges: Past, Present and the Path to a Promising Future. In: Inspired by Nature, Emergence, Complexity and Computation 28. Springer International Publishing. 2018 ISBN 978-3-319-67996-9. https://doi.org/10.1007/978-3-319-67997-6.

HOLLAND, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, EUA, 1975.

JIANG, S.; YANG, S. Evolutionary Dynamic Multiobjective Optimization: Benchmarks and Algorithm Comparisons, IEEE Transactions on Cybernetics, Volume: PP, Issue: 99, pp. 1-14, 2016.

JONATHAN, M.; PACHECO, M. A. C. Técnicas de otimização de problemas com múltiplos objetivos - Um estudo sobre o método de minimização de energia e suas variantes. Departamento de Engenharia Elétrica da Pontifícia Universidade Católica do Rio de Janeiro, Revista Rica, 2010

LABATI, R. D.; GENOVESE, A.; MUÑOZ, E.; PIURI, V.; SCOTTI, F.; SFORZA, G. Computational intelligence for industrial and environmental applications, IEEE 8th International Conference on Intelligent Systems (IS), 2016.

LACERDA, E. G. M.; CARVALHO, A. C. P. L. Introdução aos algoritmos genéticos. In: Galvão, C. O., Valença, M. J. S. (orgs.) Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais. Porto Alegre: Associação Brasileira de Recursos Hídricos. p. 99-150. 1999.

LUCA, M.; LUCA, R.; BEJINARIU, SILVIU-IOAN; CIOBANU, A.; PADURARU, O.; ZBANCIOC, M.; BARBU, T. An Overview of Several Researches on Fuzzy Logic in Intelligent Systems. 2015 International Symposium on Signals, Circuits and Systems (ISSCS), 2015.

MATLAB – The Language of Technical Computing – MathWorks. Disponível em: <http://www.mathworks.com>. Acesso em: 20 dez. 2019.

REORDA, M. S; STERPONE, L.; ULLAH, A.: An Error-Detection and Self-Repairing Method for Dynamically and Partially Reconfigurable Systems. IEEE Transactions On Computers, Volume 66, No. 6, June 2017.

RIBEIRO, F. S., Plataforma de desenvolvimento de circuitos eletrônicos adaptativos, Dissertação de Mestrado, UERJ, 2012.

SANTINI, C. C., Desenvolvimento De Uma Plataforma Reconfigurável Analógica Para A Evolução Intrínseca De Circuitos, Dissertação de Mestrado, PUC/Rio, 2001.

SANTOS, F. M. DA C.; DA SILVA, I. N.; SUETAKE, M. Sobre a aplicação de sistemas inteligentes para diagnóstico de falhas em máquinas de indução - Uma visão geral, Sba Controle & Automação, v.23, n.5, p.553-569, 2012.

SHUKLA, P. K.; TRIPATHI, S. P. A Review on the Interpretability-Accuracy Trade-Off in Evolutionary Multi-Objective Fuzzy Systems (EMOFS), Information, 3, p. 256-277, 2012.

SINOHARA, H. T., "Reparos e Ajustes Automáticos de Circuitos Eletrônicos Através de Eletrônica Evolucionária". Rio de Janeiro, 2001, 84 p. Dissertação de Mestrado (Departamento de Engenharia Elétrica - PUC-Rio).

TANSCHEIT, R.; Sistemas fuzzy, In: Inteligência computacional: aplicada à administração, economia e engenharia em Matlab, pp. 229–264, São Paulo, Thomson Learning, 2007.

ZEBULUM, R. S.; Síntese de Circuitos Eletrônicos por Computação Evolutiva. Tese de Doutorado. Departamento de Engenharia Elétrica da Pontifícia Universidade Católica do Rio de Janeiro, 1999.

ZEBULUM, R.; PACHECO, M. A.; VELLASCO, M.; "Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms". Boca Raton, Florida: CRC PRess, 2001. ISBN 0-8493-0865-8.