ESTADO DA ARTE DA COMPUTAÇÃO EVOLUTIVA APLICADA À ELETRÓNICA

Marina Valença Alencar

Departamento de Engenharia Electrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas e Planeamento Industrial

2015

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Marina Valença Alencar, Nº 1141312, 1141312@isep.ipp.pt

Orientação científica: Cecília Maria do Rio Fernandes Moreira Reis, cmr@isep.ipp.pt

Roberto Ribeiro Neli, neli@utfpr.edu.br

Departamento de Engenharia Electrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas e Planeamento Industrial

2015

Dedico este trabalho aos meus pais, Richard e Lúcia.

Agradecimentos

Agradeço primeiramente a Deus, que me guarda e protege sempre, e permitiu que eu pudesse vivenciar novas experiências.

Aos meus pais, Richard e Lúcia, e meu irmão Leonardo, pelo apoio, incentivo, amor incondicional e por me darem forças para enfrentar essa jornada. Sem eles nada teria sentido.

A toda minha família que sempre esteve ao meu lado, apoiando e incentivando o meu caminhar. Ao meu namorado, Paulo Henrique, por me dar força e enfrentar ao meu lado esse um ano de muita saudade.

Aos meus amigos, os que cresceram comigo e aos que a UTFPR me proporcionou, em especial, Thayse, Simone, Luciana, Fernanda, Jéssica, Taís, Dener, Eduardo, Mateus, João Antônio, Andrey, Suzana, Thalita, Wendel, Heitor, Rafael, pela amizade, confiança e pelos momentos que passamos juntos. Aos amigos que Portugal me trouxe, a malta toda do mestrado de SPI, aos brasileiros que conheci aqui e aos meus queridos amigos espanhóis, terei todos em minhas lembranças. À minha família de Portugal, Héber, Mario e Mateus, por estarem sempre ao meu lado.

Aos professores da UTFPR-CM que me apoiaram durante a graduação e a vinda para Porto, e aos do ISEP que nos receberam de braços abertos. Agradeço em especial aos meus orientadores, Cecília Reis e Roberto Neli, pela paciência, apoio e dedicação à mim concedida, e sobretudo ao tempo que sempre me disponibilizaram para o acompanhamento deste trabalho.

Resumo

A Computação Evolutiva enquadra-se na área da Inteligência Artificial e é um ramo das ciências da computação que tem vindo a ser aplicado na resolução de problemas em diversas áreas da Engenharia. Este trabalho apresenta o estado da arte da Computação Evolutiva, assim como algumas das suas aplicações no ramo da eletrónica, denominada Eletrónica Evolutiva (ou *Hardware* Evolutivo), enfatizando a síntese de circuitos digitais combinatórios.

Em primeiro lugar apresenta-se a Inteligência Artificial, passando à Computação Evolutiva, nas suas principais vertentes: os Algoritmos Evolutivos baseados no processo da evolução das espécies de Charles Darwin e a Inteligência dos Enxames baseada no comportamento coletivo de alguns animais.

No que diz respeito aos Algoritmos Evolutivos, descrevem-se as estratégias evolutivas, a programação genética, a programação evolutiva e com maior ênfase, os Algoritmos Genéticos. Em relação à Inteligência dos Enxames, descreve-se a otimização por colônia de formigas e a otimização por enxame de partículas. Em simultâneo realizou-se também um estudo da Eletrónica Evolutiva, explicando sucintamente algumas das áreas de aplicação, entre elas: a robótica, as FPGA, o roteamento de placas de circuito impresso, a síntese de circuitos digitais e analógicos, as telecomunicações e os controladores.

A título de concretizar o estudo efetuado, apresenta-se um caso de estudo da aplicação dos algoritmos genéticos na síntese de circuitos digitais combinatórios, com base na análise e comparação de três referências de autores distintos.

Com este estudo foi possível comparar, não só os resultados obtidos por cada um dos autores, mas também a forma como os algoritmos genéticos foram implementados, nomeadamente no que diz respeito aos parâmetros, operadores genéticos utilizados, função de avaliação, implementação em *hardware* e tipo de codificação do circuito.

Palavras-Chave

Computação Evolutiva, Algoritmo Genético, Eletrónica Evolutiva, síntese de circuitos digitais.

Abstract

Evolutionary Computation is part of the area of Artificial Intelligence and is a branch of computer science that has been applied to solve problems in several areas of engineering. This work presents the state of the art of Evolutionary Computation, as well as some of its applications in the electronics field, called Evolutionary Electronics (or Evolutionary Hardware), emphasizing the synthesis of combinatorial digital circuits.

Firstly we present the Artificial Intelligence and then the Evolutionary Computation in its main aspects: the evolutionary algorithms based on the process of evolution of Charles Darwin and the swarm intelligence based on the collective behavior of some animals.

Regarding the evolutionary algorithms, we describe the evolutionary strategies, the genetic programming, the evolutionary programming and with greater emphasis, the Genetic Algorithms. Regarding the Swarm Intelligence, we describe the ant colony optimization and the particle swarm optimization. Simultaneously it was also carried out a study of Evolutionary Electronics, explaining succinctly some of the application areas, including: robotics, FPGA, the routing of printed circuit boards, the synthesis of digital and analog circuits, telecommunications and controllers.

In order to materialize this study, we present a case study on the application of genetic algorithms in the synthesis of combinatorial digital circuits, based on the analysis and comparison of three different authors.

Through this study it was possible to compare, not only the results obtained by each of the authors, but also how genetic algorithms have been implemented, particularly in what concerns to parameters, genetic operators, the fitness function, hardware implementation and the type circuit coding.

Keywords

Evolutionary Computation, Genetic Algorithms, Evolutionary Electronics, synthesis of digital circuits.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

CE – Computação Evolutiva

AE – Algoritmos evolutivos

IE – Inteligência dos Enxames

IA – Inteligência Artificial

AG – Algoritmos Genéticos

PE – Programação Evolutiva

PG – Programação Genética

ACO – Otimização por Colônias de Formigas

PSO – Otimização por Enxame de Partículas

t_c – Taxa de cruzamento

t_m – Taxa de mutação

Tc – Tamanho do cromossoma

EE – Eletrónica Evolutiva

HE – *Hardware* Evolutivo

RE – Robótica Evolutiva

PLD – *Programmable Logic Devices*

FPGA – *Field Programmable Gate Array*

SRAM – *Static Random Access Memory*

PROM – *Programmable Read Only Memory*

EPROM – *Erasable Programmable Read Only Memory*

EEPROM – *Electrical Erasable Programmable Read Only Memory*

MOS – *Metal Oxide Semiconductor*

PCI – Placas de Circuitos Impresso

TH – *Through-Hole Technology*

SMT – *Surface Mount Technology*

SMD – *Surface Mount Device*

PLA – *Programmable Logic Array*

CMOS – *Complementary Metal Oxide Semiconductor*

SDH – *Synchronous Digital Hierarchy*

IP – *Internet Protocol*

RWA – *Routing and Wavelength Assignment*

FAP – *Frequency Assignment Problem*

PID – Proporcional, Integral e Derivativo

Cout – *Carry out*

Cin – *Carry in*

1. INTRODUÇÃO

Esta tese pretende descrever a execução do trabalho realizado, no âmbito da unidade curricular de Tese/Dissertação, do 2º ano do Mestrado em Engenharia Eletrotécnica e de Computadores. Trata-se de um "estado da arte" referente à Computação Evolutiva, como, também a aplicação desse conceito em algumas áreas da Eletrónica. Seguindo um caso de estudo dos Algoritmos Genéticos aplicado na síntese de circuitos digitais, através da comparação de três referências de autores.

1.1. CONTEXTUALIZAÇÃO

Este trabalho surgiu da proposta da Professora Doutora Cecília Reis, juntamente com o interesse de realizar um trabalho na área da Computação Evolutiva (CE), como também da chamada Eletrónica Evolutiva, que basicamente é a interseção da CE com a eletrónica.

1.2. OBJETIVOS

Este trabalho tem como objetivo principal o "estado da arte" da Computação Evolutiva, suas vertentes e características, assim como a aplicação desse conceito à algumas áreas da eletrónica, dentre elas: robótica, FPGA, roteamento de placas de circuito impresso, síntese

de circuitos digitais e analógicos, telecomunicações e controladores. Realizar um caso de estudo detalhado da aplicação dos algoritmos genéticos na síntese de circuitos digitais (especificamente em circuitos combinatórios) do trabalho de três autores distintos, analisando e comparando-os.

1.3. CALENDARIZAÇÃO

Sendo o "estado da arte" da Computação Evolutiva, como também suas aplicações em algumas áreas da eletrónica e o caso de estudo da aplicação dos Algoritmos Genéticos na síntese de circuitos digitais combinatórios, os focos principais deste trabalho, as etapas para elaboração do trabalho seguem na Tabela 1.

Tabela 1 – Calendarização referente as etapas do trabalho

	ETAPAS	MESES						
		Mar.	Abr.	Mai.	Jun.	Jul.	Ago.	Set.
1	Elaboração da proposta de trabalho	■						
2	Levantamento bibliográfico sobre a Computação Evolutiva e suas aplicações	■	■	■				
3	Estudo da Computação Evolutiva e seus principais aspectos e algoritmos emergentes detalhados no capítulo 2		■	■	■			
4	Estudo da Computação Evolutiva na síntese/desenvolvimento de alguns circuitos eletrónicos, base do capítulo 3				■	■		
5	Caso de estudo da aplicação dos Algoritmos Genéticos na síntese de circuitos digitais					■	■	
6	Conclusões referentes ao trabalho em si e ao caso de estudo						■	
7	Ajustes finais na redação do trabalho para posterior apresentação						■	■

1.4. ORGANIZAÇÃO DO RELATÓRIO

Neste primeiro capítulo abordou-se a introdução ao trabalho, sua contextualização e os objetivos a serem cumpridos.

O segundo capítulo relata o “estado da arte” da CE, uma apresentação introdutória com o histórico da mesma e uma abordagem mais detalhada das suas duas principais vertentes: Algoritmos Evolutivos e Inteligência dos Enxames.

O terceiro capítulo aborda a aplicação da Computação Evolutiva em algumas áreas da eletrónica, entre elas: robótica, FPGA, roteamento de placas de circuito impresso, síntese de circuitos digitais e analógicos, telecomunicações e controladores. Colocando ênfase na síntese de circuitos digitais, pois será a área abordada no caso de estudo do capítulo 4.

Em seguida, no quarto capítulo faz-se um caso de estudo dos algoritmos genéticos aplicados à síntese de circuitos digitais combinatórios, através de três trabalhos referentes a abordagem de autores distintos, analisando e comparando-os.

Por fim, no quinto capítulo são reunidas as principais conclusões referentes ao trabalho em si e ao caso de estudo proposto.

2. COMPUTAÇÃO EVOLUTIVA

O capítulo 2 faz uma introdução à Computação Evolutiva e aponta detalhadamente duas das principais vertentes, os Algoritmos Evolutivos (AE) e a Inteligência dos Enxames (IE), assim como suas subdivisões.

2.1. INTRODUÇÃO

São várias as definições encontradas para Inteligência Artificial (IA), basicamente é a ciência que tenta compreender a inteligência num todo e simular sistemas com comportamentos parecidos com a inteligência humana. Uma de suas áreas é a Computação Evolutiva que será analisada neste capítulo.

A CE é uma área de estudo que trabalha com algoritmos guiados pelos princípios da teoria da evolução natural de Darwin, com o objetivo de encontrar a solução apropriada do problema independente de sua aplicação [1].

Seguindo a teoria de Darwin, publicada em 1859, que diz que todos os indivíduos são diferentes e devido a essas diferenças uns são mais aptos a determinados ambientes, por

isso possuem maior chance de sobreviver e gerar descendentes, que herdarão essas características [2].

Robert Charles Darwin (Figura 2) foi o cientista inglês que revolucionou a biologia no fim do século XIX, com a obra "*The Origin of Species*" (Figura 1), na qual demonstrou que os organismos tendem a produzir descendentes ligeiramente diferentes dos pais, e que a seleção natural favorece aqueles que se adaptam melhor ao meio ambiente, assim, determinados indivíduos têm características que os tornam mais capazes para sobreviver e reproduzir [3].

A teoria de Darwin pode ser resumida da seguinte forma:

1) Os filhos tendem a ser em maior número que os pais;

2) O número de indivíduos de uma espécie de uma geração para outra permanece constante;

3) Dos itens acima, conclui-se que haverá competição pela sobrevivência;

4) Dentro de uma mesma espécie, os indivíduos apresentam pequenas diferenças, muitas delas presentes nos respectivos pais;

5) O princípio da seleção natural indica que os indivíduos cujas variações se adaptarem melhor ao ambiente terão mais chances de sobreviver e se reproduzir [4].

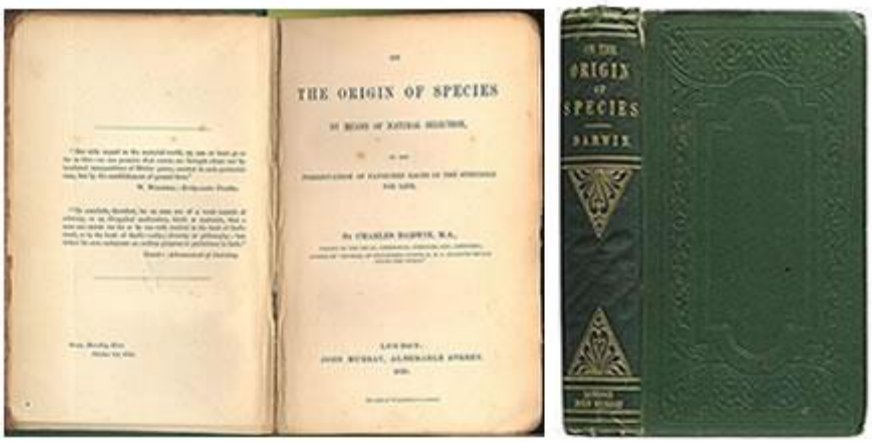


Figura 1 - Capa do livro "The Origin of Species"

[5]

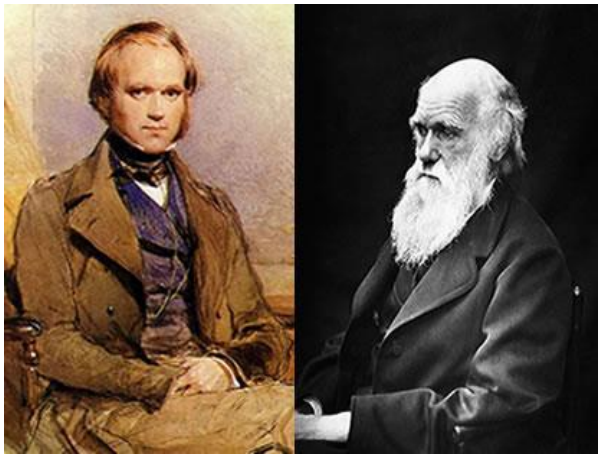


Figura 2 - Robert Charles Darwin

[5]

A vantagem mais significativa da CE está na possibilidade de resolver problemas dando-lhes a solução mais apropriada e não necessariamente a ótima, para isso são utilizadas principalmente duas vertentes de algoritmos, como mostrado na Figura 3: Algoritmos Evolutivos e Inteligência dos Enxames. Os AE são baseados na evolução por meio da seleção natural, recombinação de material genético (cruzamento) e mutações, são divididos em Algoritmos Genéticos, Programação Evolutiva, Estratégias Evolutivas, Programação Genética. Em contrapartida tem-se a IE, que se baseia no comportamento coletivo de algumas espécies, dentre esses algoritmos estão a Otimização por Colônias de Formigas, por Enxames de Partículas, dentre outros.

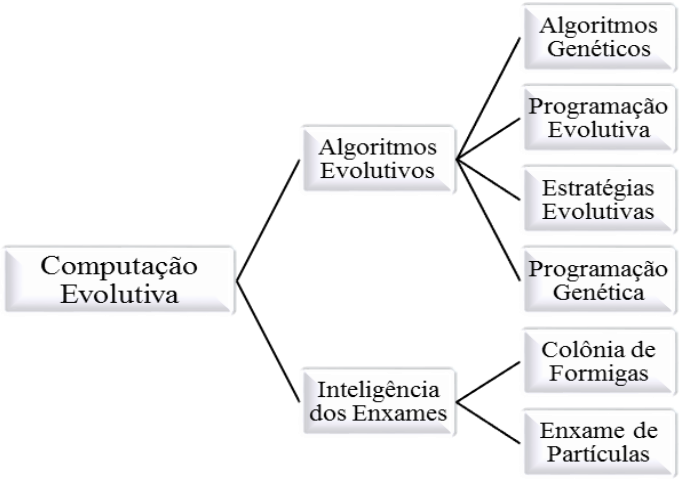


Figura 3 - Fluxograma da CE

2.2. ALGORITMOS EVOLUTIVOS

Em 1930, iniciaram-se as primeiras pesquisas na aplicação da evolução natural em algoritmos de exploração. Com o passar dos anos, em 1960, com a facilidade ao acesso a computadores, finalmente desenvolveram-se três principais abordagens dos AE: Algoritmos Genéticos (Holland, 1962), Programação Evolutiva (Fogel, 1962), Estratégias Evolutivas (Rechenberg e Schwefel, 1965). Depois desenvolveu-se a Programação Genética (Koza, 1992) [6].

Apesar das quatro abordagens terem sido desenvolvidas separagemmente, elas têm o mesmo princípio básico: é gerado, normalmente aleatoriamente, uma população inicial, cada indivíduo dessa população é considerado um candidato para solução do problema e o seu tamanho geralmente é constante, mas pode variar no decorrer do processo. Através de uma função de avaliação (*fitness*), que define o objetivo da otimização, é avaliado a qualidade (adaptação) dos indivíduos atribuindo-lhes um valor, assim pelo processo de seleção, os indivíduos mais aptos passam a ser uma solução inicial para o problema. Esse conjunto inicial pode passar pelo processo de mutação e *crossover* que são respectivamente, mudança para aparecer novos materiais genéticos e troca de material genético entre dois ou mais indivíduos, gerando novos descendentes para a próxima geração. Repetindo-se até que seja encontrada uma solução aceitável para o problema.

Nas subseções 2.2.1, 2.2.2, 2.2.3, 2.2.4 serão analisadas cada uma dessas abordagens sucintamente.

2.2.1. ALGORITMO GENÉTICO

Os Algoritmos Genéticos (AG), do inglês *Genetic Algorithms*, foram propostos por John Holland e seus alunos nos anos 60, com o objetivo de estudar os fenômenos naturais de adaptação e desenvolver modelos à serem implementados para diversos problemas de otimização [7].

Baseiam-se na seleção natural, na qual os seres mais aptos se destacam e têm maior probabilidade de sobrevivência, esta evolução, assim como na Biologia, se dá através dos operadores de Seleção, Mutação e Recombinação (também chamada de *crossover*). São aplicados em uma grande gama de problemas, sendo utilizados estratégias inteligentes de

pesquisa. O AG tem sido mais utilizado devido sua simplicidade e aplicações relacionadas a otimização e síntese de sistemas [8].

De acordo com Silva (2011), algoritmos genéticos podem ser definidos como procedimentos de pesquisa baseados na genética e seleção natural das espécies. Assim como acontece no meio ambiente, em um AG existe um grupo de soluções candidatas, conhecidas como indivíduos, que competem entre si para garantir a própria sobrevivência [9].

Um algoritmo genético para um problema particular deve ter os seguintes componentes [10]:

- Uma representação genética para soluções candidatas ou potenciais (processo de codificação);
- Uma maneira de criar uma população inicial de soluções candidatas ou potenciais;
- Uma função de avaliação, classificando as soluções em termos de sua adaptação ao ambiente, ou seja, sua capacidade de resolver o problema;
- Operadores genéticos;
- Valores para os diversos parâmetros usados pelo algoritmo genético (tamanho da população, probabilidades de aplicação dos operadores genéticos, etc.)

São muito empregados devido a:

- Versatilidade, pois a sua função é genérica, podendo ser aplicados a qualquer tipo de problema, sem a necessidade de mudar o programa principal;
- Robustez, apesar de não garantirem a solução ótima, garantem uma melhor solução para o problema;
- Simplicidade, pois são de fácil programação e compreensão;
- Eficiência, pois problemas de níveis complexos podem ser solucionados [4].

Apesar dos AG terem muitas vantagens em relação aos algoritmos clássicos, sua maior desvantagem está no tempo de processamento, principalmente no que diz respeito a

questão de avaliação dos indivíduos. Muitos investigadores tentam minimizar essa deficiência estudando algoritmos genéticos melhorados, alterando os operadores genéticos e procurando novos métodos de recombinação.

Como é um algoritmo baseado no processo de adaptação natural, a terminologia utilizada também segue a da teoria seleção natural e da genética. Então, um indivíduo corresponde a uma cadeia de caracteres (cromossomas), onde cada caractere (gene), encontra-se numa dada posição (*locus*) e com seu valor determinado (alelo). Um sinônimo de indivíduo é o genótipo e a sua estrutura decodificada é o fenótipo. A partir do fenótipo, o potencial de sobrevivência pode ser obtido através da avaliação da função aptidão. Nessa comparação, descreve-se o problema em forma de uma função matemática, em que os indivíduos mais aptos obterão valores mais altos de função, assim cada indivíduo é uma possível solução. Então, num grupo de indivíduos, verifica-se a potencialidade de cada um em relação ao grupo, tentando selecionar os mais aptos para o cruzamento. Depois de efetuado o cruzamento, cada gene de cada indivíduo estará sujeito a uma eventual mutação. Baseiam-se nos processos naturais de seleção, cruzamento e mutação, conhecidos como operadores genéticos [11].

Para inicializar o algoritmo, escolhe-se uma população inicial, que é normalmente gerada de forma aleatória. Através da função aptidão, avalia-se toda a população conforme a qualidade de cada indivíduo. Em seguida, através da seleção, escolhe-se os indivíduos dados como mais aptos anteriormente para a criação de uma nova geração (um novo conjunto de soluções possíveis). Esses indivíduos selecionados sofrem as duas operações genéticas que misturam suas características, o cruzamento e a mutação. Com isso, esses passos são repetidos até que seja encontrada uma solução aceitável ou o algoritmo não possa melhorar uma solução já encontrada. Na Figura 4 encontra-se um fluxograma da estrutura básica do AG [12].

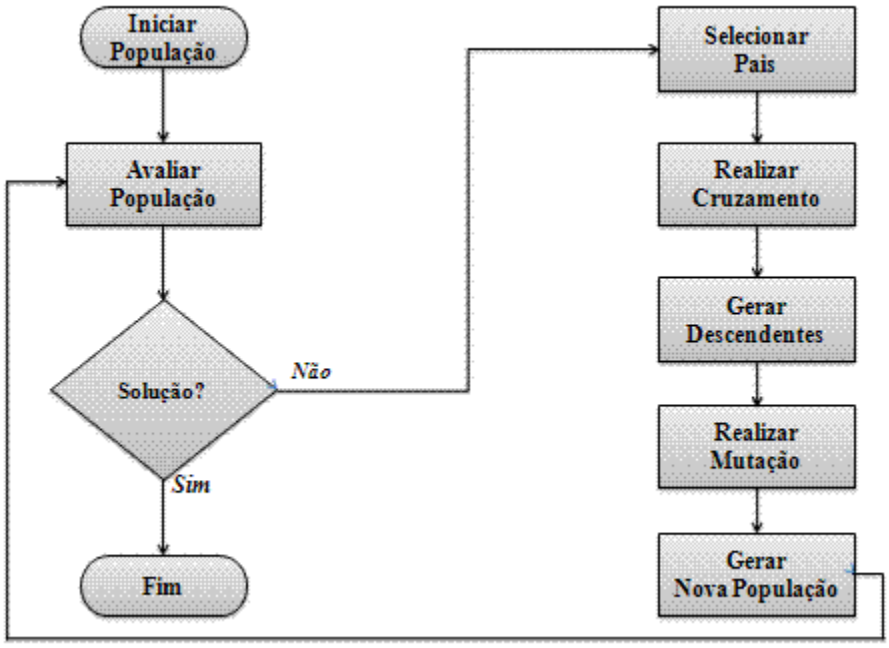


Figura 4 - Fluxograma do AG

[12]

A seguir, está apresentado um pseudocódigo que representa um algoritmo genético genérico [13]:

Inicializa *a população*
Avalia *indivíduos na população*
Repita
  Selecione *indivíduos para reprodução*
    Aplique *operadores de recombinação e mutação*
    Avalie *indivíduos na população*
  Selecione *indivíduos para sobreviver*
Até *critério de paragem satisfeito*
Fim

A seguir é apresentada mais detalhadamente a função aptidão (*fitness*), os operadores genéticos empregados nos AG e os parâmetros genéticos que são utilizados para que ocorra a diversificação da população, através de sucessivas gerações, mantendo as características genéticas da geração anterior.

A) FUNÇÃO APTIDÃO (*fitness*): Nos AG os indivíduos são avaliados de acordo com a função objetivo, que define o problema em estudo, fornecendo uma medida de como os indivíduos se comportam no domínio do problema. Esta função é definida pelo utilizador para modelar o sistema, é importante que seja representada precisamente, pois é através dela que se mede a proximidade de um indivíduo à solução desejada ou quão boa é esta solução.

A função aptidão é a parte da programação que exige o maior custo computacional, uma vez que ela avalia todos os indivíduos de cada geração, consumindo enorme tempo neste processo. Haupt, em 1998, propôs alguns cuidados especiais para se diminuir este custo computacional como, por exemplo: não avaliar mais de uma vez o mesmo indivíduo, evitar gerar cromossomas idênticos na população inicial, verificar se os pais são idênticos aos filhos, manter a população com todos os cromossomas distintos entre si e criar uma memória para os algoritmos genéticos, verificando se um determinado indivíduo já não foi gerado anteriormente [13].

B) SELEÇÃO: O objetivo da seleção é escolher os melhores indivíduos de acordo com o melhor *fitness*, para que originem descendentes ainda mais aptos ao problema. Apesar disso, não são escolhidos apenas os melhores, a fim de evitar a convergência no máximo local (valor que parece ser o melhor, mas não é efetivamente a melhor solução para o problema) [14].

Por isso, utiliza-se métodos para selecionar os indivíduos. Existe diversos deles, os mais utilizados são o Método da Roleta e o Método por Torneio.

- Método da Roleta, do inglês *Roulette Wheel*, foi proposto primeiramente por Holland, a cada indivíduo é atribuído uma probabilidade de ser selecionado, proporcional ao valor de aptidão do indivíduo com o total da aptidão acumulada, assim os indivíduos com maiores aptidões possuem maiores chances de serem sorteados. Neste método os indivíduos já sorteados, voltam a aparecer na lista dos possíveis indivíduos a serem sorteados. A roleta é girada de acordo com o tamanho da população. Como mostra a Figura 5 [2]:

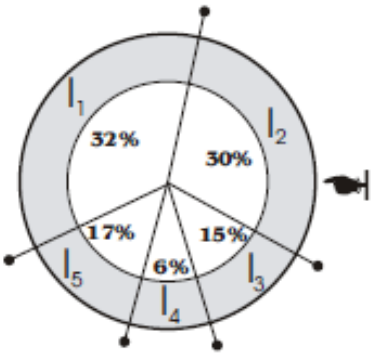


Figura 5 - Método da Roleta

[2]

Método de seleção proporcional à aptidão pode originar alguns problemas, como, por exemplo, ocasionar o surgimento de um grande número de cópias de um bom cromossoma, cuja aptidão seja elevada, diminuindo, consequentemente, a variabilidade da população, ocasionando problemas de convergência prematura. Este modelo também é fortemente dependente da escala da função aptidão, ou seja, quando maior a escala, menor será a diferença entre a probabilidade de escolha entre os melhores indivíduos e os piores indivíduos [13].

- Método de seleção por Torneio, consiste na escolha aleatória de um número fixo *N* de indivíduos da população atual. Dentre esses indivíduos, apenas o que possui a maior aptidão é copiado para a população seguinte. Repete-se este processo até completar a nova população. A seleção pode ser com ou sem reposição dos indivíduos já sorteados. Sua complexidade varia proporcionalmente ao tamanho da população, pois é independente de uma ordenação prévia dos elementos e do cálculo das probabilidades de seleção [2].

É comum que ocorra a cópia do melhor indivíduo da população atual para a nova, este processo chama-se elitismo, e tem como objetivo que esse indivíduo difunda suas características para os demais da população, privilegiando a melhor solução possível. Nesse indivíduo selecionado pelo elitismo procura-se não aplicar os operadores genéticos de mutação e cruzamento para não adulterar a solução representada por aquele indivíduo.

C) CRUZAMENTO: O operador de cruzamento (*crossover*), permite fazer a troca de material genético entre dois ou mais indivíduos, permitindo propagar as características dos

progenitores considerados mais aptos, criando novos indivíduos. A Figura 6 apresenta um exemplo do *crossover* com representação binária, onde o corte pode ser feito em qualquer ponto do cromossoma.

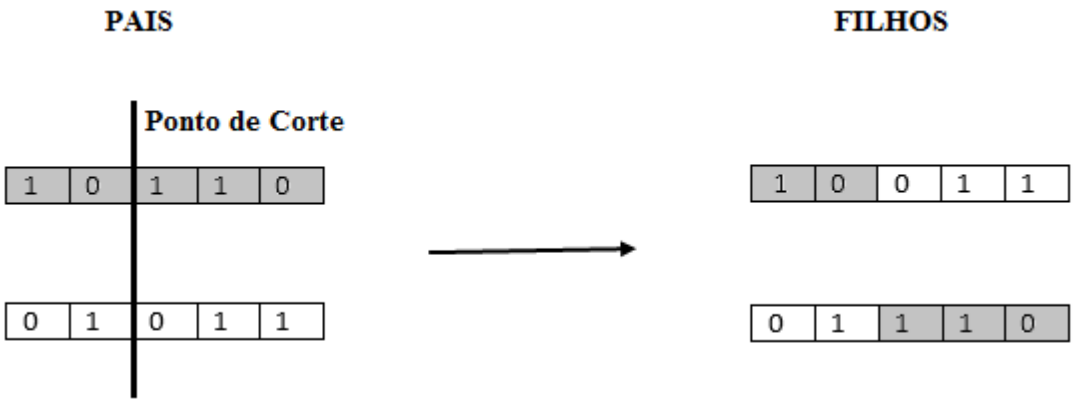


Figura 6 – *Crossover*

D) MUTAÇÃO: O operador de mutação permite a diversificação genética na população, pois varia a informação de um indivíduo sozinho por vez. Na representação binária, troca-se o *bit* 0 por 1 ou 1 por 0, representando um indivíduo completamente diferente, como ilustra a Figura 7.

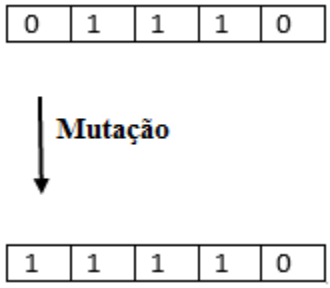


Figura 7 – Mutação

E) PARÂMETROS GENÉTICOS: são três os parâmetros genéticos que afetam diretamente no desempenho do AG, a taxa de cruzamento (t_c), taxa de mutação (t_m) e o tamanho da população. Escolhas inadequadas desses parâmetros podem aumentar o tempo de convergência, convergir prematuramente, estagnação da pesquisa, maior necessidade de recursos computacionais ou não convergir para uma solução viável.

A taxa de cruzamento determina a probabilidade de um cruzamento ocorrer. Quanto maior for essa taxa, mais rapidamente novas estruturas serão introduzidas na população. Mas se essa for muito alta, a maior parte da população será substituída, podendo ocorrer perda de estruturas de alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

A taxa de mutação determina a probabilidade de uma mutação ocorrer. Uma baixa taxa previne que uma dada solução fique estagnada em um valor, causando uma convergência prematura. Com uma taxa muito alta, a pesquisa se torna essencialmente aleatória.

O tamanho da população determina o número de cromossomas na população, afetando o desempenho global e a eficiência dos AG. Em uma população pequena, o desempenho pode cair, pois a população fornece uma pequena cobertura do espaço de pesquisa do problema. Uma grande população geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. Entretanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais ou um período maior de trabalho do algoritmo [15].

A Figura 8 mostra um resumo das características dos AG:

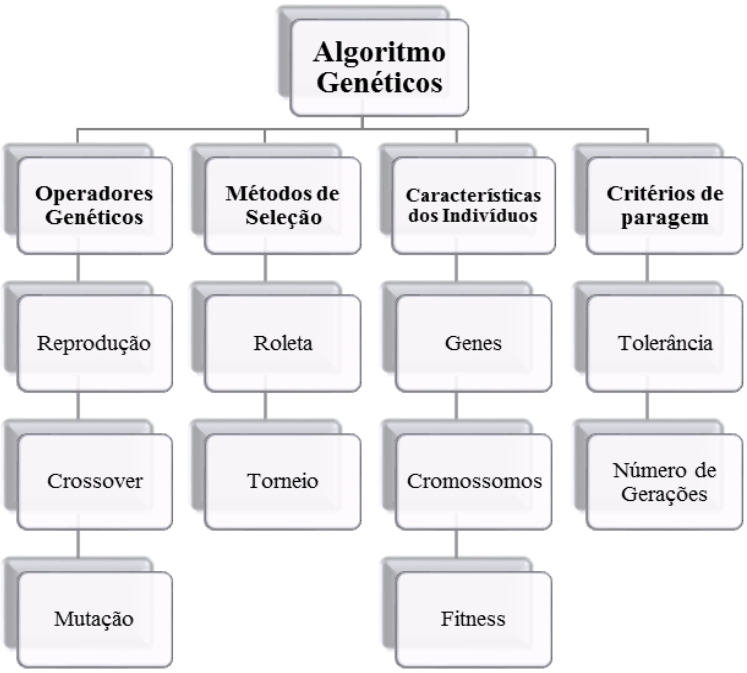


Figura 8 - Características do Algoritmos Genéticos

2.2.2. PROGRAMAÇÃO EVOLUTIVA

A Programação Evolutiva (PE) foi idealizada em 1962 por Lawrence J. Fogel, com um modelo de otimização análoga ao dos AG, enquanto neste o relacionamento entre os pais e seus descendentes é emulado. No entanto, enfatizam o relacionamento entre os progenitores e seus descendentes ao invés de tentar emular operadores genéticos específicos observados na natureza [16].

Como não realiza mutações, a representação da PE se torna mais flexível do que nos AG.

2.2.3. ESTRATÉGIAS EVOLUTIVAS

O modelo das Estratégias Evolutivas foi desenvolvido por Rechenberg e Schwefel em 1965. Foram concebidas para tratarem problemas técnicos de otimização e quase que exclusivamente empregadas na engenharia civil como alternativa aos métodos convencionais. Operam com cromossomas na forma de vetores de números reais e originalmente na proporção (1+1), isto é, cada progenitor gera um herdeiro por geração, normalmente por mutações distribuídas. Caso este descendente seja melhor que seu progenitor ele lhe toma o lugar. Atualmente estas estratégias foram estendidas para as proporções (m+1) e (m+n), além de terem tido estratégias de recombinações introduzidas no seu processo evolutivo [16].

2.2.4. PROGRAMAÇÃO GENÉTICA

A Programação Genética (PG) foi estudada em 1992 por John Koza, introduzida para solucionar problemas de aprendizado de máquina, buscando a construção automática de programas de computadores [17].

Tem uma abordagem semelhante aos Algoritmos Genéticos, considerada por muitos uma extensão destes devido à semelhança das duas abordagens, a principal diferença entre ambas é que nos AG a representação das soluções é abstrata e altamente estruturada, enquanto a PG apresenta como soluções programas de computador em uma linguagem de programação específica [18].

PG e AGs representam um campo novo de pesquisa dentro da Ciência da Computação. Neste campo muitos problemas continuam em aberto na tentativa de serem encontradas

novas soluções e ferramentas. Apesar disso, este paradigma tem-se mostrado bastante poderoso e muitos trabalhos exploram o uso de AGs e PG para solucionar problemas em diferentes áreas do conhecimento, desde tratamento de dados e biologia molecular, até ao projeto de circuitos elétricos e algoritmos de controlo [19].

2.3. INTELIGÊNCIA DOS ENXAMES

O termo Inteligência dos Enxames (IE), do inglês *Swarm Intelligence*, diz respeito a algoritmos de otimização baseados no comportamento coletivo de determinadas espécies naturais para solucionar problemas corriqueiros, em sistemas descentralizados e auto-organizados, como apresenta a Figura 9. Alguns exemplos dessa organização dos grupos é a Otimização por Colônias de Formigas e por Enxame de Partículas que engloba o comportamento dos animais, como, cardume de peixes, manada de animais e bando de pássaros.

Figura 9 - Enxame de formigas colaborando para criar uma ponte viva.
[20]

Pesquisadores têm muitas razões para achar o estudo de inteligência de enxames atrativo, pois oferece um caminho alternativo para o desenvolvimento de sistemas inteligentes por possuir autonomia, emergência e controle distribuído [21].

As propriedades principais de um sistema de inteligência de enxame são [22]:

- **Proximidade:** os agentes devem ser capazes de interagir;
- **Qualidade:** os agentes devem ser capazes de avaliar seus comportamentos;
- **Diversidade:** permite ao sistema reagir a situações inesperadas;
- **Estabilidade:** nem todas as variações ambientais devem afetar o comportamento de um agente;
- **Adaptabilidade:** capacidade de adequação a variações ambientais.

Nas subseções 2.3.1, 2.3.2 serão analisados particularmente alguns dos exemplos de otimização.

2.3.1. OTIMIZAÇÃO POR COLÔNIAS DE FORMIGAS

A Otimização por Colônias de Formigas (ACO), do inglês *Ant Colony Optimization*, foi inventada por Marco Dorigo em 1992. É um algoritmo baseado no comportamento coletivo das formigas ao saírem de suas colônias para encontrar comida através do cominho mais curto, como ilustra a Figura 10. Normalmente, a formiga anda de forma aleatória até encontrar o alimento, para então retornarem a colônia deixando o rastro de uma substância química natural delas que permite o reconhecimento entre elas, o feromônio. Assim, quando outras formigas encontrarem esse rastro, tendem a percorrer por ele e não mais aleatoriamente até o alimento, retornando até a colônia e enfatizando o rastro. Portanto, o caminho com maior concentração de feromônio, é o melhor caminho a ser seguido.

Segundo Payá-Zaforteza (2007), a analogia do comportamento das formigas com a otimização se realiza do seguinte modo [24]:

- A procura de alimento é equivalente à exploração das soluções factíveis em um problema de otimização combinatória;
- A quantidade de alimento é similar ao valor da função objetivo;
- O rastro de feromônio é a memória adaptativa do método.

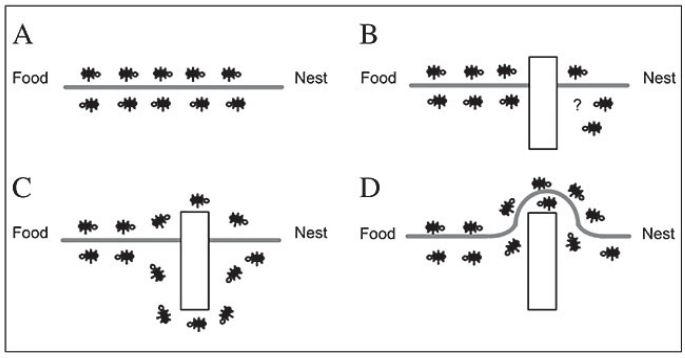


Figura 10 - Comportamento das formigas

[23]

2.3.2. OTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS

O modelo de Otimização por Enxames de Partículas (PSO), do inglês *Particle Swarm Optimization*, foi desenvolvido pelo psicólogo social James Kennedy e o engenheiro eletricista Russel Eberhart em 1995. Tem esse nome, pois é baseado no comportamento coletivo de alguns animais (peixes e alguns insetos), na forma como eles se movimentam e na procura por alimento, como ilustra a Figura 11.

Esse algoritmo de otimização comparado com os outros da Computação Evolutiva, é de fácil implementação, contém poucos parâmetros para ajuste, além de não fazer uso dos operadores de *crossover* e mutação. O algoritmo tem sido aplicado com sucesso em diversas áreas, tais como: otimização de funções, treinamento de redes neurais artificiais, controlo de sistemas nebulosos [25].

De acordo com SILVA 2011, originalmente foi criado para tratar problemas de otimização com variáveis reais não exigindo nenhum tipo de codificação das soluções. O PSO se baseia na informação da trajetória das partículas (indivíduos) e os pontos do espaço de pesquisa visitados por elas para informar a qualidade da solução (qualidade da função objetivo). Para tanto, usa-se uma estrutura de memória para preservar os melhores locais visitados. A indicação da movimentação de cada partícula a cada nova iteração depende de duas informações: a melhor posição de todo o enxame e a melhor posição da própria partícula [9].

Figura 11 - Aves voando alinhadas à procura de alimento

[20]

As partículas possuem dois operadores associados a elas: o vetor posição e o vetor velocidade. O vetor posição grava a posição da partícula no espaço de pesquisa e o vetor velocidade direciona as mudanças de posição das partículas durante a execução do algoritmo. Além da informação desses dois operadores, cada partícula grava duas posições: a posição *global best* (*gbest*), que é a melhor posição conhecida pelo enxame, e a posição *personal best* (*pbest*), que é a melhor posição conhecida pela partícula. Essas posições funcionam como um histórico de melhores resultados a ser utilizado no processo decisório de reposicionamento, ou seja, a partícula deve procurar se movimentar na direção das melhores regiões visitadas por ela e pela partícula com melhor resultado momentâneo do enxame.

Na inicialização, são gerados aleatoriamente os vetores, que representam as posições das partículas no espaço de pesquisa. Em seguida a função *fitness* é utilizada para calcular a aptidão de cada partícula. Ao ter a informação do valor de aptidão de cada partícula, o algoritmo verifica qual informação vai ficar gravada nas posições *pbest* e *gbest*. Na primeira iteração, na posição *pbest* de cada partícula fica gravada justamente a sua posição inicial. Nas demais iterações, a posição *pbest* somente será atualizada se a aptidão da partícula na iteração for melhor. Por sua vez, a posição *gbest* na primeira iteração grava a posição da partícula que obteve melhor valor de aptidão; nas demais iterações, a posição *gbest* somente é atualizada quando alguma partícula obtém aptidão melhor. Após verificar as informações das posições *pbest* e *gbest* é avaliado o critério de paragem, caso este não tenha sido alcançado o algoritmo continua atualizando o vetor velocidade de cada partícula, como é mostrado no fluxograma da Figura 12 [26].

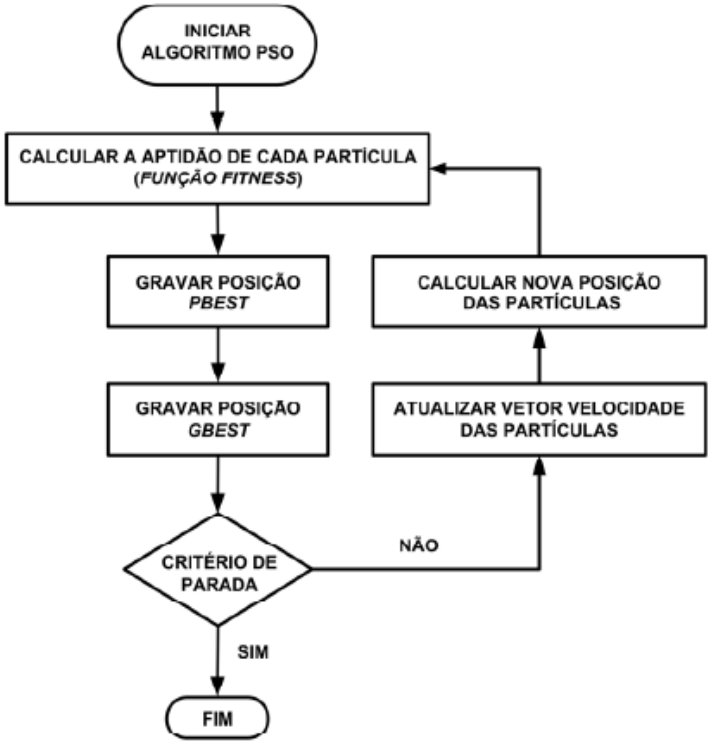


Figura 12 - Fluxograma do PSO

[26]

A implementação do algoritmo é simples, e concentra-se na avaliação, comparação e imitação. Assim como nos algoritmos genéticos, a avaliação é um processo no qual é atribuído um número real representativo a cada partícula, de acordo com a proximidade relativa de sua posição com a do alvo, sua função cognitiva. Sendo essa a função objetivo do problema, que dependendo de sua natureza deseja-se minimizar ou maximizar. A comparação, estabelece a influência da aptidão das partículas vizinhas no movimento futuro de uma partícula em direção a uma região mais próxima do alvo desejado, determinando sua posição social. A imitação é a ponderação entre as posições anteriores (cognitivas e sociais) de cada partícula, determinando seu movimento futuro, velocidade em módulo, direção e sentido, que representa a diferença entre duas posições no espaço vistas entre duas iterações consecutivas. Com a velocidade calculada, aplica-se à partícula, forçando-a a assumir uma nova posição no espaço de soluções. Assim, o algoritmo se repete, realizando novas iterações, até que atinja os critérios de paragem (são justamente iguais ao dos AG), número de iterações e tolerância em relação ao alvo [27].

Como é baseado na interação social, no PSO são estabelecidos relações entre as partículas, e como se influenciam umas às outras, de acordo com a topologia definida, as partículas se movimentam dentro do espaço de soluções, tendo em vista que tendem a se concentrar nas regiões onde as soluções estão mais próximas do alvo. Na Figura 13 são apresentadas algumas topologias encontradas [27].

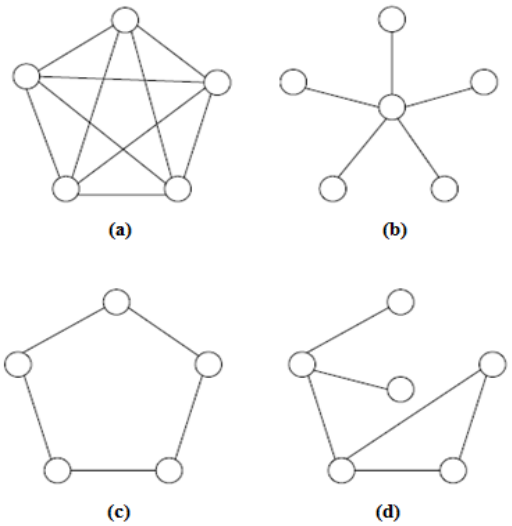


Figura 13 - Topologias: (a) estrela, (b) roda, (c) círculo, (d) randômica
[27]

As principais características do PSO estão resumidas na Figura 14 [27].

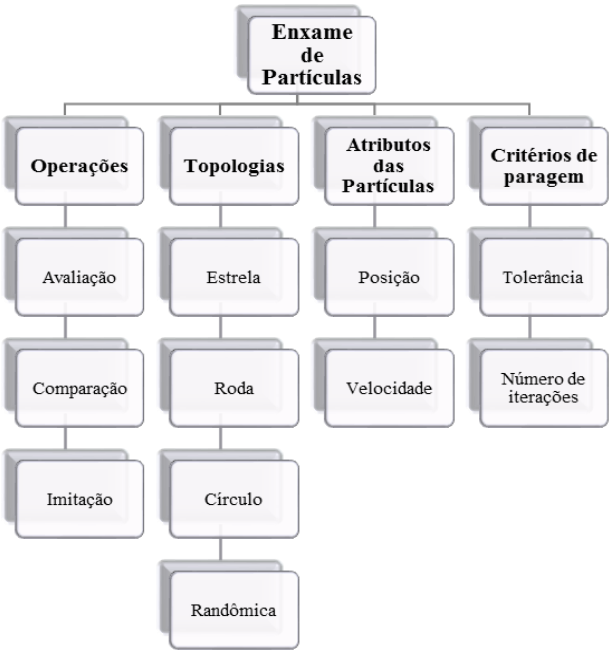


Figura 14 - Características do PSO

3. COMPUTAÇÃO EVOLUTIVA APLICADA À ELETRÓNICA

No Capítulo 3 será abordado o estudo da Eletrónica Evolutiva (EE), que compreende a aplicação da Computação Evolutiva na eletrónica, assim como suas diferentes áreas de aplicação.

3.1. INTRODUÇÃO

Dentre as várias aplicações da CE, se encontra a chamada Eletrónica Evolutiva, na qual compreende na interseção dos sistemas eletrónicos com a CE. Em muitos casos o nome *Hardware* Evolutivo (HE) é utilizado como sinônimo da EE, no entanto existem autores que distinguem as duas nomenclaturas, chamando de HE um caso específico da EE, quando o algoritmo é aplicado em plataformas reconfiguráveis. Neste trabalho serão utilizadas ambas as nomenclaturas como sinônimos. Na Tabela 2 segue uma lista com alguns trabalhos relevantes das aplicações da EE, apesar de existir outros não menos

importantes que estes. A intenção neste trabalho é apenas identificar aquelas que começaram com as tendências inovadoras na área [28].

Tabela 2 – Trabalhos relevantes das aplicações da EE

DATA	AUTORES	APLICAÇÃO
1991	Louis e Rawlins	Evolução das funções digitais básicas
1993	H. de Garis	Introdução ao conceitos de *Hardware* Evolutivo
1995	Higuchi *et al.*	Evolução dos circuitos digitais de reconhecimento de padrões
	Hemmi *et al.*	Uso da linguagem de descrição de hardware para evoluir circuitos
	Grimbleby *et al.*	Síntese de rede analógica automática usando AE
	Mange *et al.*	Chips reconfiguráveis com propriedades de autorreparação e autorreprodução
1996	Thompson	Primeira evolução intrínseca usando FPGA
	Koza *et al.*	Evolução de filtro passa-baixa e amplificadores de transistores bipolares
	Keymeulen *et al.*	Evolução dos circuitos digitais para sistema de navegação robótica
1997	Miller *et al.*	Evolução de novos circuitos digitais aritméticos
1998	Flockton e Sheenan	Evolução intrínseca para circuitos analógicos
	Zebulum *et al.*	Evolução de um circuito digital para controle de CPU
	Koza *et al.*	Evolução do circuito analógico para aplicações de controle
	Thompson *et al.*	Primeiros resultados sobre a evolução intrínseca de um circuito robustos
1999	Miler *et al.*	Evolução de filtros digitais
	Stoica *et al.*	Evolução do circuito CMOS analógico em um chip reconfigurável
	Zebulum *et al.*	Evolução multi-objetivos para filtros ativos
	Lohn *et al.*	Evolução multi-objetivos para circuitos analógicos

Apesar de ser uma área que está a se desenvolver são capazes de automatizar o desenvolvimento de circuitos digitais, analógicos e programáveis, mudando o comportamento autonomamente de acordo com a interação. A maior vantagem do uso desses algoritmos, se comparado aos métodos tradicionais (manuais), está em otimizar e encontrar os melhores circuitos.

A EE pode ser classificada pela natureza do projeto (eletrónica analógica ou digital), tipo de projeto (otimização ou síntese de circuitos) e meio evolutivo (extrínseca ou intrínseca), como sintetiza a Tabela 3. Nas aplicações em meio extrínseco, a evolução é realizada via *software* em outro ambiente e ao término do processo de evolução, somente o melhor cromossoma é repassado ao PLD[1] (*Programmable Logic Device*). Em meio intrínseco a evolução e a avaliação são realizados no dispositivo reprogramável, os cromossomas de cada geração são avaliados em *hardware*.

Tabela 3 – Classificação da Eletrónica Evolutiva

MEIO EVOLUTIVO	NATUREZA DO PROJETO	TIPO DE PROJETO
Extrínseca	Analógica	Otimização
Intrínseca	Digital	Síntese

Os operadores genéticos (evolutivos) são responsáveis por modificar a disposição dos componentes no circuito e alterar suas características, aumentando ou diminuindo o tamanho dos circuitos. A representação escolhida deve facilitar o mapeamento do genótipo e fenótipo do projeto.

[1] Como plataforma de programação, o HE se utiliza de circuitos lógicos programáveis, como FPGAs, CPLDs ou componentes de uso específico como ASIC's (*Application Specific Integrated Circuit*) e mais recentemente o FPTA (*Field Programmable Transistor Arrays*) e FPAA (*Field Programmable Analog Array*) usados em síntese de circuitos analógicos.

São diversas as áreas da eletrónica onde podem ser aplicados os algoritmos da CE. Em meio à essas áreas, nas subseções deste capítulo serão apresentadas algumas dessas aplicações, tais como:

- Robótica;
- FPGA (*Field Programmable Gate Array*);
- Roteamento de placas de circuito impresso;
- Síntese de circuitos digitais e analógicos;
- Telecomunicações;
- Controladores.

3.2. ROBÓTICA

De acordo com o dicionário Aurélio, o termo Robótica é definido como "*Conjunto dos estudos e das técnicas tendentes a conceber sistemas capazes de substituírem o homem em suas funções motoras, sensoriais e intelectuais*" [29]. Por várias décadas o homem vem pesquisando técnicas de aprimorar seus processos produtivos, com o desenvolvimento da tecnologia, e somente na segunda metade do século XX, foi possível automatizar esses processos, com o surgimento do conceito de "robôs" e "inteligência artificial". Com o estudo e desenvolvimento destas áreas, nos últimos anos, os robôs inteligentes já são utilizados em diversas aplicações de áreas diferentes, como mostra a Figura 15, tais como [30]:

- Substituição de humanos em tarefas cotidianas;
- Ambientes arriscados: capacidade de executar tarefas que seriam risco de vida aos humanos;
- Linha de montagem: substituição de peças, manipulação de materiais, soldagem e pintura;
- Transporte: cadeira de rodas automáticas, helicópteros autônomos, navegação em autoestradas;
- Medicina: realização de cirurgias precisas;
- Entretenimento: cães robôs, futebol de robôs, etc.

Figura 15 – Exemplo de aplicações no cotidiano
[30]

A robótica trata de máquinas reprogramáveis com funções variadas, que podem executar tarefas normalmente associadas a seres humanos. Também possui a capacidade de decidir determinadas ações que devem ser tomadas e planear a sua execução, de acordo com as alterações e restrições colocadas pela tarefa ou pelo meio de interação. É cada vez mais utilizada pois executa uma grande diversidade de tarefas de forma quase humana, se adapta a diferentes situações e é de fácil programação. Os robôs atuais são estruturas capazes de um controle preciso de movimento e com algumas técnicas de programação, permitem definir movimentos e repeti-los com elevada precisão [31].

A computação evolutiva é muito utilizada para sintetizar automaticamente controladores embebidos para robôs e equipes de robôs, a fim de treiná-los para desenvolver tarefas específicas. Está associada a otimizações realizadas na operação de um robô, como, encontrar a melhor trajetória ou a melhor maneira de se executar uma ação. Com ela define-se o objetivo do robô, sem necessitar definir como ele deve fazer para atingir determinado objetivo desejado, permanecendo sempre em constante evolução à procura das melhores soluções. É possível encontrar uma solução para um objetivo específico, de maneira robusta e com diversidade de soluções, permitindo a autoprogramação de sistemas complexos com um ou mais robôs.

Muitos pesquisadores analisam a aplicação das técnicas evolutivas em todo o processo do desenvolvimento do robô, e não apenas em partes do planeamento do mesmo. A etapa de evolução de um robô ou de um grupo deles pode ser simulada em um computador e depois os parâmetros da solução encontrada são colocados neles. No entanto, para aumentar a autonomia dos mesmos, a evolução pode ocorrer durante a sua execução [30].

Ao desenvolvimento de sistemas de controle adaptativo baseados nas técnicas de CE na síntese automática de controladores para robôs, a fim de treiná-los para desenvolver tarefas especificas, dá-se o nome de Robótica Evolutiva (RE), do inglês *Evolutionary Robotics.* Sugere que se determine o design de um robô ao longo das gerações, combinando indivíduos e favorecendo os com melhores características (*fitness*). Embora a evolução física do *hardware* ainda não seja algo totalmente consolidado, se restringindo à existência de uns poucos protótipos, a possibilidade de acelerar etapas do processo torna este tipo de evolução atrativo. Já em relação ao sistema de controle, todo o potencial dos algoritmos evolutivos, aliado à crescente capacidade de processamento hoje disponível, tem sido utilizado de forma integral para síntese automática de controladores [32].

A Figura 16 ilustra o processo de avaliação por um algoritmo genético na RE. Uma população inicial, que codifica o sistema de controle do robô, é aleatoriamente gerada e testada no ambiente. Cada robô é colocado no ambiente com o objetivo de realizar alguma tarefa e é avaliado com relação à aptidão (*fitness*) para resolver a tarefa. Os melhores indivíduos, ou seja, os robôs com melhor desempenho, são escolhidos para dar origem a uma nova população (próxima geração). Seus genótipos são mantidos ou modificadores pelos operadores genéticos, dando origem a uma nova geração de descendentes. O processo é repetido para um número *N* de gerações até que um indivíduo satisfaça algum critério de paragem pré-estabelecido [33].

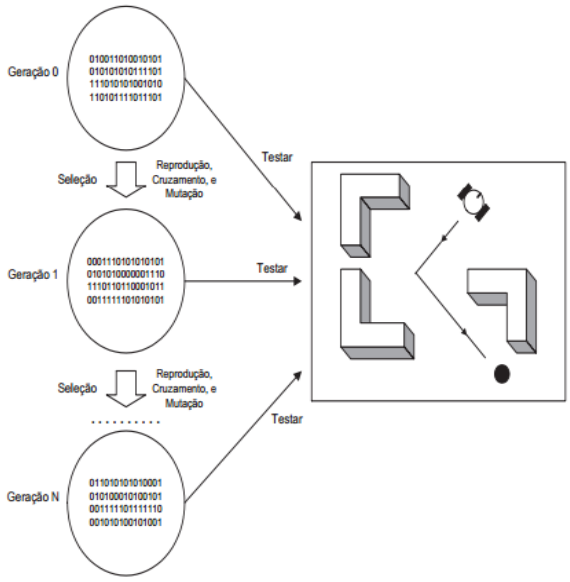


Figura 16 – Visão geral da Robótica Evolucionária
[33]

Problemas de navegação autônoma de robôs são, sem dúvida, muito desafiadores devido a dificuldades técnicas e conceituais. Sem qualquer intervenção humana, robôs autônomos devem executar tarefas em ambientes muitas vezes desconhecidos e cuja dinâmica é imprevisível, contendo apenas as informações provenientes de sensores e os graus de liberdade associados aos atuadores, que nem sempre são tão precisos. A complexidade do problema aumenta ao abordar vários robôs interagindo em um mesmo ambiente, onde um se torna obstáculo para o outro. A este fenômeno dá-se o nome de Robótica Coletiva. Pode-se comparar essa abordagem com os fundamentos da inteligência dos enxames, que baseia-se no comportamento coletivo de alguns animais, que analogamente nos robôs, permitiria a cooperação mútua nas tarefas [32].

3.3. FPGA (*FIELD PROGRAMMABLE GATE ARRAY*)

Muitos dos *hardwares* dos dispositivos eletrónicos que possuem implementação e estrutura fixa, são projetados para realizar sempre as mesmas funções. Ultimamente, estão sendo desenvolvidos dispositivos de uma nova geração, capazes de melhorar o processamento, denominados de dispositivos de lógica programável (PLD's). Os PLDs permitem a reconfiguração do *hardware* com uma reprogramação para atender uma tarefa específica, tornando-o muito útil, pois é mais flexível, com menor custo de desenvolvimento e facilidade de modificação [34].

Com a necessidade de implementar circuitos lógicos maiores, foi introduzido em 1983 pela empresa Xilinx Inc. o *Field Programmable Gate Array* (FPGA), um dispositivo lógico programável que suporta a implementação de circuitos lógicos grandes.

Consiste em um grande arranjo de células configuráveis contidos em um único circuito integrado, onde cada célula tem capacidade computacional de implementar as funções lógicas e realizar o roteamento para comunicação entre elas. Basicamente é constituída por:

- **Blocos lógicos:** formam uma matriz bidimensional no interior de cada bloco lógico, existem várias maneiras possíveis para implementação de funções lógicas, são programados para realizar as funções necessárias, e os canais de roteamento realizam a interconexão necessária entre os blocos;

- **Blocos de entrada e saída (I/O):** dispostos ou alocados ao redor do dispositivo, cada um desses blocos pode servir como entrada, saída ou acesso bidirecional a outros pinos de entrada e saída;
- **Chaves de interconexão:** são organizadas como canais de roteamento horizontal e vertical entre os blocos lógicos. Esses canais possuem chaves de interligação programáveis que permitem conectar os blocos lógicos em função das necessidades de cada projeto.

A Figura 17 apresenta a estrutura padrão de uma FPGA.

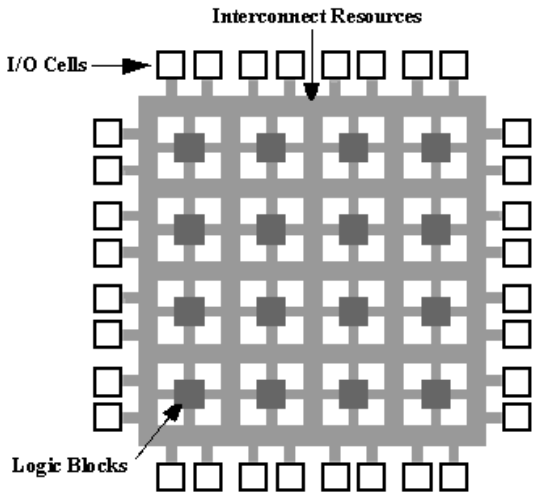


Figura 17 –Estrutura padrão FPGA

[36]

Existem três tipos de tecnologias de programação de um FPGA:

- **SRAM (*Static Random Access Memory*):** o controle dos transistores de passagem ou multiplexadores é feito por uma memória estática de acesso randômico SRAM. Devido à volatilidade dessas memórias, os FPGAs que se utilizam dessa tecnologia precisam de uma memória externa tipo PROM[2] (*Programmable Read Only Memory*), EPROM[3] (*Erasable Programmable Read Only Memory*) ou EEPROM[4] (Electrical *Erasable*

[2] Memória programável somente para leitura.
[3] Memória somente de leitura programável e apagável, é uma memória PROM que pode ser apagada se exposta à luz ultravioleta.
[4] Memória somente de leitura programável, e apagável eletricamente, é uma memória EPROM que pode ser apagada eletricamente sem o auxílio da luz ultravioleta.

Programmable Read Only Memory). Essa tecnologia ocupa muito espaço no circuito integrado, porém é rapidamente reprogramável;

- **Antifuse:** é um dispositivo de dois terminais, que no estado não programado apresenta uma alta impedância entre seus terminais (circuito aberto). A vantagem do seu uso está no tamanho reduzido, baixa capacitância quando não programado e baixa resistência quando programado;
- ***Gate* Flutuante:** baseado em transistores MOS (*Metal Oxide Semicondutor*), especialmente construídos com dois *gates* flutuantes semelhantes ao usados nas memórias EPROM e EEPROM. A maior vantagem dessa tecnologia é sua capacidade de programação e a retenção dos dados, os mesmos podem ser programados com o circuito integrado instalado na placa [36].

Através da CE, soluções de *hardware* evolutivo podem ser criadas de forma que circuitos eletrónicos não precisem de monitoramento de variáveis, cálculos de otimização ou intervenção humana para sua manutenção [10].

Geralmente, a evolução se dá a partir do arquivo binário, gerado depois da criação de um código com a linguagem de descrição de *hardware* do mapeamento do FPGA, pois para a evolução do arquivo binário só é necessário realizar a geração deste arquivo uma única vez, ou seja, as etapas de programação e projeto de um FPGA serão executadas somente uma vez, sendo a partir deste momento a programação do FPGA evoluída [37].

3.4. ROTEAMENTO DE PLACAS DE CIRCUITO IMPRESSO

Placas de circuito impresso (PCI), do inglês *Printed Circuit Board*, são o centro de quase todos os dispositivos eletrónicos, logo o seu *design* e fabricação são componentes extremamente importantes de muitos processos de produção industrial. Antes de uma PCI poder realizar sua tarefa, ela evolui através de três etapas principais. A primeira é o *design* lógico, que define os componentes a serem utilizados e as suas interligações. A segunda etapa é o *layout* físico da PCI onde as posições geométricas dos componentes e as suas ligações físicas são decididas. A etapa final é a produção industrial da PCI [38].

A PCI consiste num substrato inerte onde são impressas (depositadas) trilhas de condutores (cobre) sobre um ou ambos os lados, cuja função é conectar eletricamente os componentes fixados na placa para que assim executem as suas funções. O processo de fabricação tem evoluído ao longo dos tempos, sendo efetuado com um elevado grau de rapidez e precisão.

A placa de circuito impresso foi inventada pelo Doutor Paul Eisner, um cientista austríaco, após a Segunda Guerra Mundial. No início eram feitas de materiais cerâmicos e foram evoluindo tecnologicamente, passando por diversas modificações e adaptações. Hoje, são produzidas com multicamadas e, normalmente, feitas com um material laminado conhecido como FR-4.

Há duas tecnologias utilizadas no processo de montagem da PCI:

- ***Through-Hole Technology* (TH):** consiste basicamente na inserção dos componentes e soldagem dos componentes, como mostra a Figura 18. São fáceis para construir, testar e trabalhar, porém em projetos muito complexos, o *hardware* é fisicamente grande e poderá ser eletricamente ruidoso para aplicações em média e alta frequência. Os equipamentos que compõe a linha de montagem TH são ilustrados na Figura 19.

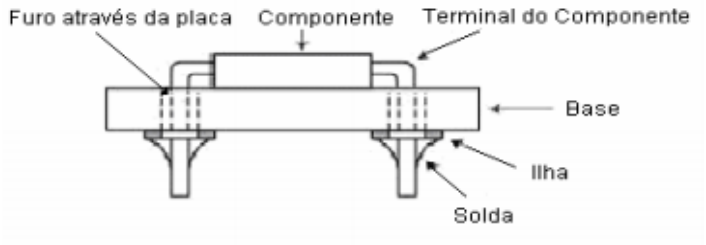


Figura 18 - Montagem de componente utilizando a tecnologia TH

[39]

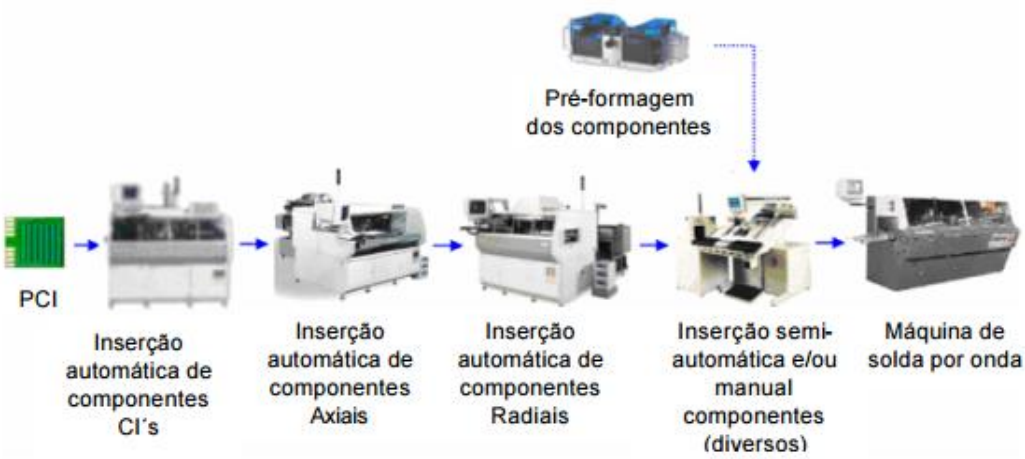


Figura 19 - Linha de Montagem TH

[39]

- ***Surface Mount Technology* (SMT):** é a tecnologia de montagem mais recente, onde os componentes são montados diretamente na superfície das placas de circuito impresso, não necessitando a perfuração da placa, como ilustra a Figura 20. Os dispositivos montados com essa tecnologia são chamados de dispositivos de montagem superficial (SMD - *Surface Mount Device*). Pode ocorrer utilizando o processo de soldagem por refusão (aplicar calor suficiente até que ocorra a separação da solda e do fluxo e posteriormente o derretimento da solda) ou através da soldagem pela máquina de solda por onda (como feito na tecnologia TH). Os equipamentos que compõe a linha de montagem SMT são mostrados na Figura 21.

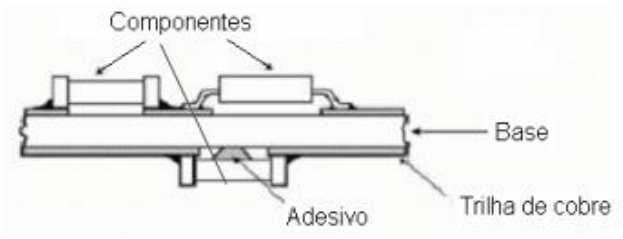


Figura 20 - Montagem de componente utilizando a tecnologia SMT

[39]

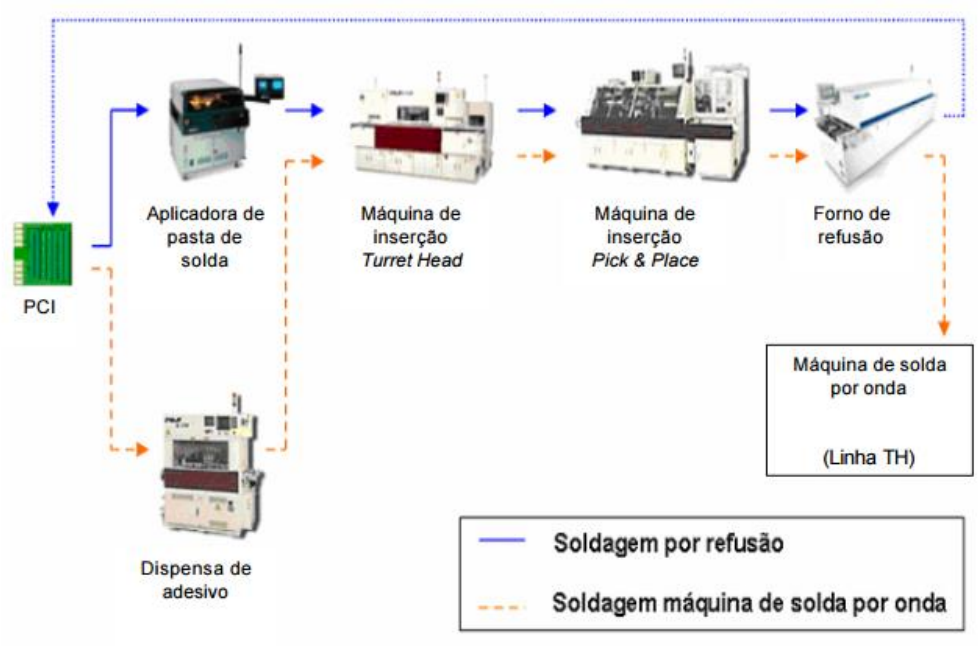


Figura 21 - Linha de Montagem SMT

[39]

Conforme a necessidade de cada projeto, poderá conter componentes montados com ambas as tecnologias, TH e SMT, ou somente cada uma delas, podendo ser montados em ambas as faces da placa ou em apenas uma [39].

A computação evolutiva é empregada nas PCI para otimizar o processo de posicionamento e montagem de componentes, de modo que minimize os custos e aumente a produtividade. Em geral, num projeto de PCI a disposição física dos componentes é definida pelo projetista, em função do tipo e tamanho dos componentes, da distribuição sobre a placa, dissipação de potência, proximidade de conectores, facilidade de acesso e manutenção dos componentes. Após a alocação dos componentes é necessário interligar os terminais destes através de trilhas de cobre. Esse caminho de trilha de um terminal até o outro não pode cruzar outra trilha da mesma camada, pois causaria um curto-circuito. Existem inúmeros softwares disponíveis para roteamento automático de PCI, sendo que a maioria deles é de um custo bastante elevado. O roteamento automático consiste em encontrar sequencialmente trilhas que não conflitam entre si, interligando os pontos necessários, até que todos os terminais sejam conectados. Os algoritmos de roteamento automático usualmente são derivados da teoria dos grafos, porém nem sempre são eficientes para PCI com muitos componentes e ligações.

A utilização da CE para o problema de roteamento automático de PCI é ainda uma área de pouca pesquisa, mas com resultados promissores. Além disto, a CE também vem sendo aplicada a nível de montagem dos componentes eletrónicos sobre a placa, pois minimizando o tempo de colocação e soldagem, consequentemente minimiza-se os custos de produção [40].

3.5. SÍNTESE DE CIRCUITOS

A síntese de circuitos abrange tanto os projetos analógicos como os digitais, sendo predominantes em sistemas digitais, pela existência atual de eficientes ferramentas de projeto. A otimização e a síntese do circuito é realizada a partir de um projeto inicial do circuito, fornecido como entrada funcional, mas não otimizada [41].

A disponibilidade de uma grande quantidade de simuladores para realizar experimentos extrínsecos e a disponibilidade dos dispositivos programáveis (FPGA), motivam a síntese de circuitos digitais e a otimização de funções lógicas. A síntese de um circuito inicia-se pela descrição de um problema, ou de sua especificação funcional, procurando uma topologia adequada para aplicação prática, escolha de tipos e valores de componentes, e se for o caso, a otimização do circuito (topologia mínima).

A síntese de circuitos mostra-se como uma tarefa mais complexa do que a de otimização, pois envolve aspectos físicos, técnicos e econômicos, tornando complexa a modelagem do problema [8].

3.5.1. CIRCUITOS DIGITAIS

Um sistema digital é a combinação de dispositivos para manipular uma informação lógica ou quantidades físicas que são representadas no formato digital, ou seja, as quantidades podem assumir apenas valores discretos. São na maioria das vezes dispositivos eletrónicos, mas podem ser também mecânicos, magnéticos ou pneumáticos.

O sistema de numeração convencional nos sistemas digitais é o binário, que opera com tensões que se encontram em faixas predeterminadas representadas pelos binários 0 e 1.

As portas lógicas são componentes básicas da eletrónica digital, com elas é possível a construção de qualquer circuito lógico. Geralmente são apresentadas pela equação lógica, símbolo ou tabela verdade. Podem possuir N entradas, mas apenas uma saída referente à sua função, cada terminal pode ter uma das condições binárias 0 (baixa ou falsa) ou 1 (alta ou verdadeira). Segue no Anexo A as sete portas lógicas básicas (com duas entradas) e suas devidas representações.

Também são chamados de circuitos lógicos, pois cada tipo de circuito digital obedece a um determinado conjunto de regras lógicas, como a forma com que o circuito responde a uma determinada entrada [42]. A Figura 22 ilustra a sequência do processo, no qual a partir da situação obtêm-se a tabela verdade e a partir desta, através de uma das técnicas de simplificação (Álgebra de Boole e Mapas de Karnaugh), a expressão simplificada e obtém-se o circuito final.

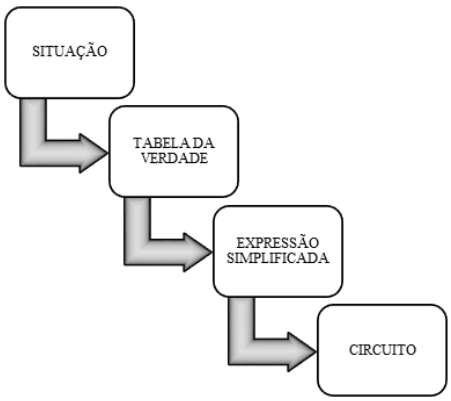


Figura 22 – Sequência do processo

Em meados de 1800, George Boole, um matemático inglês, apresentou a Álgebra de Boole, um sistema matemático de análise lógica, através de seus postulados, propriedades, identidades e teoremas fundamentais, que são utilizados para efetuar as devidas simplificações nas expressões e circuitos lógicos. No Anexo A está presente um quadro resumo com as especificações da álgebra booleana.

Para efetuar as simplificações, existem basicamente dois processos. O primeiro é através da Álgebra de Boole (como visto anteriormente), e o segundo é com a utilização dos mapas (diagramas) de Karnaugh. Estes, por sua vez, permitem a simplificação de maneira mais rápida dos casos extraídos de tabelas da verdade, um mapa onde se encontra possíveis

situações com seus respectivos resultados. Esses diagramas são estudados para problemas de até cinco variáveis geralmente**.**

O projeto de circuitos digitais envolve os campos de sistemas combinatórios e sistemas sequenciais.

Os circuitos combinatórios são aqueles em que a saída depende exclusivamente das combinações entre as variáveis de entrada, ou seja, são utilizados para solucionar problemas que necessitam de uma resposta perante determinadas situações (representadas pelas variáveis de entrada). Para a construção desses circuitos é necessário possuir as suas expressões características, obtidas através do processo já visto (tabela da verdade e simplificações). Compreendem os circuitos como: somadores, subtratores, codificadores, decodificadores, entre outros [43]. No Anexo A encontra-se detalhadamente o funcionamento de cada um desses circuitos. A Figura 23 mostra o esquema geral de um circuito combinatório, composto pelas variáveis de entrada, o circuito lógico em si e suas saídas [8].

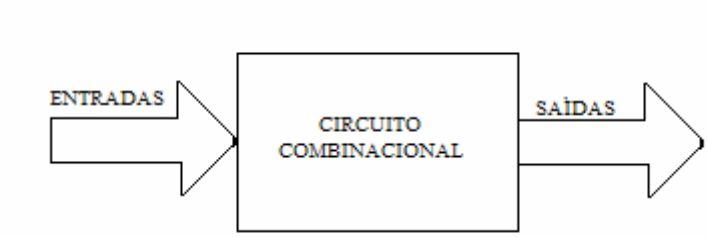


Figura 23 – Esquema geral de um circuito lógico combinatório

[8]

Os circuitos sequenciais, diferente dos combinatórios, não têm as saídas dependentes apenas das variáveis de entrada, mas também dos seus estados anteriores que permanecem armazenados, geralmente sob o comando de uma sequência de pulsos (*clock*). Esses circuitos compreendem os *flip-flops*, contadores e registradores, como apresenta a Figura 24.

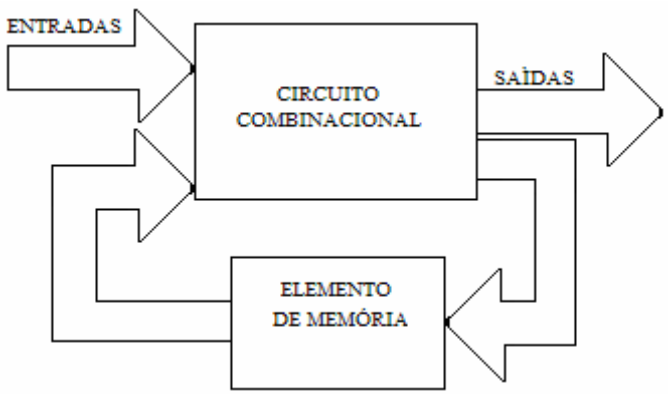


Figura 24 – Esquema geral de um circuito lógico sequencial
[8]

Os sistemas digitais foram os primeiros experimentos da EE, onde os algoritmos evolutivos têm sido utilizados para resolver problemas de otimização das funções lógicas [41]. Embora a maioria das aplicações das técnicas evolutivas nesses projetos tem sido na área de circuitos combinatórios [44].

No primeiro trabalho na síntese de circuitos digitais foram utilizados os algoritmos genéticos, proposto por Louis e Rawlins em 1991, com aplicação nas funções digitais básicas. John Koza em 1992, adaptou a programação genética na síntese de circuitos digitais combinatórios, em exemplos como multiplexadores e detectores de paridade. Na sequência destes trabalhos vieram outros também relevantes [8].

A abordagem convencional para projetar um circuito digital envolve uma grande quantidade de tarefas, enquanto com a CE ocorre sem a realização de um grande número de tarefas. As tarefas que a abordagem convencional inclui são: a declaração do problema, a determinação do número de variáveis de entrada e saída, a tabela verdade, a definição do mapeamento de entradas e saídas, a expressão booleana simplificada e o desenho de circuitos. Por outro lado, a síntese automática exige apenas a definição de variáveis de entrada e saída, o mapeamento de entradas e saídas e o desenho de circuitos [45].

Existem na literatura diferentes tipos de codificação, ou seja, a forma em que os circuitos são codificados em um cromossoma, entre elas: utilizando funções, portas lógicas e a representação com mapeamento de *bits* em um dispositivo programável.

Na representação com funções utiliza-se a soma de produtos das variáveis lógicas da função combinatória, onde os cromossomas são codificados em uma coleção de genes para

representar todos os possíveis produtos de variáveis lógicas, como mostra a Figura 25. Onde o ‘0’ é o complemento da variável, o ‘1’ é a presença da variável e o ‘2’ representa a ausência de uma determinada variável.

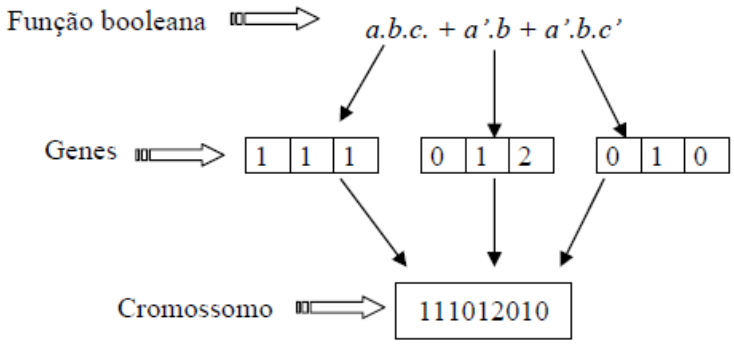


Figura 25 – Representação cromossômica de uma função booleana
[41]

Para a codificação do circuito por portas lógicas, na qual um cromossoma representa as entradas e as funções lógicas por números inteiros. Subdivide-se o cromossoma em níveis que determinam o circuito (quantidade de níveis e o número de portas para cada nível são determinados pelo projetista). Através da Figura 26 pode-se observar o modo como é realizada esta representação, ilustra através de um exemplo de circuito lógico em três níveis, sendo definidas cinco portas lógicas de duas entradas, codificando-as de 0 a 4, determinando oito entradas possíveis de E1 até E7 e uma saída.

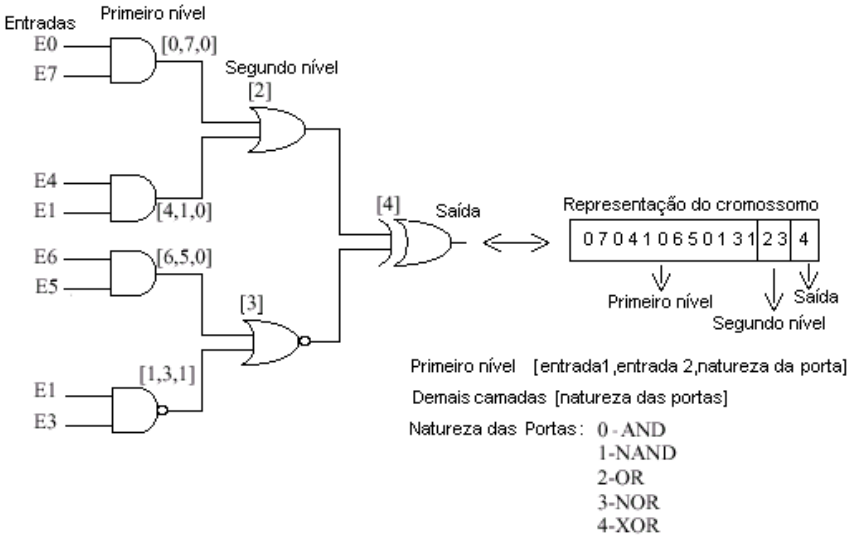


Figura 26 - Mapeamento entre circuitos e cromossomas
[41]

A codificação na forma de mapeamento de fusíveis se baseia na arquitetura de um dispositivo lógico programável, especificamente o *Programmable Logic Array* (PLA) (Anexo B). Formado internamente de uma matriz de funções AND e OR programáveis e conexões de fusíveis, sendo capaz de representar qualquer função combinatória, como mostra a Figura 27. Têm as entradas (A0, A1) e as saídas (Y0, Y1, Y2, Y3), as conexões ou fusíveis (f_i) compõem as linhas de fusíveis (p_0, p_1, p_2, p_3), na qual define zero ‘0’ como um fusível inativo e um ‘1’ para ativo ou conectado. O dimensionamento do mapa de fusíveis é realizado a partir das especificações do circuito, ou seja, do número de entradas *n* (quantidade de *buffers* inversores), do número de termos produto *k* (quantidade de portas AND) e do número de saídas *m* (quantidade de portas OR). A quantidade de conexões programáveis no plano de entrada é igual a $2.n.k$ e no plano de saída é igual a $k.m$. O total de número de genes no cromossoma (*Ng*) é calculado de acordo com a equação 3.1.

$$Ng = 2.n.k + k.m \tag{3.1}$$

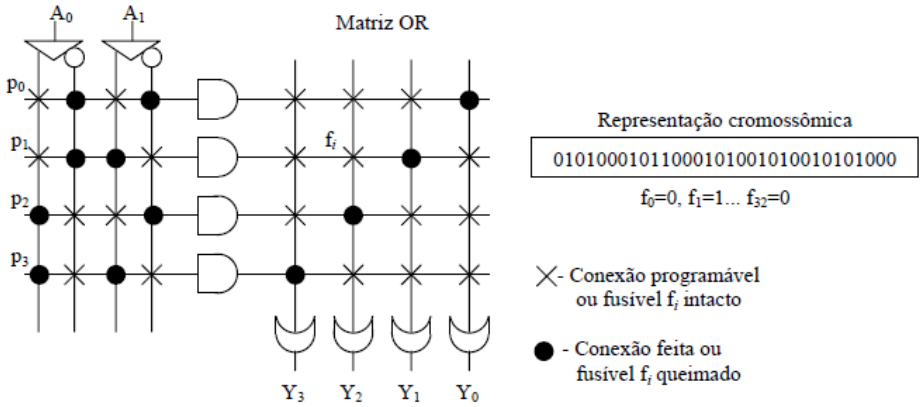


Figura 27 - Mapeamento de fusíveis e sua representação cromossômica
[41]

Esta representação é restrita à aplicações em circuitos combinatórios que são descritos por somas de produtos ou conjuntos completos e a avaliação é feita somente por sua tabela verdade. Encontram-se na literatura de síntese de circuitos outros modelos de representação, como, por exemplo, usando transistores [8].

3.5.2. CIRCUITOS ANALÓGICOS

Os circuitos analógicos estão relacionados com o sinal analógico que o circuito manipula nas entradas e saídas. Um sinal analógico é aquele contínuo no tempo, podendo assumir qualquer valor em um espaço contínuo de valores para um instante qualquer no tempo. Os amplificadores operacionais são um dos mais importantes blocos no projeto de um circuito analógico, suas principais características são: grande resistência de entrada e ganho diferencial de tensão, baixa resistência de saída e ganho de modo comum, resposta em frequência constante, insensibilidade à temperatura e variações da tensa de alimentação.

O projeto de circuitos analógicos é mais complexo do que projetos de natureza digital e depende mais da experiência e intuição do projetista [28].

Em uma das primeiras aplicações foi apresentada uma ferramenta de síntese e otimização de amplificadores operacionais CMOS[5] (*Complementary Metal Oxide Semiconductor*) utilizando algoritmos genéticos. A partir disto foram desenvolvidas várias metodologias para síntese de circuitos analógicos. Os principais circuitos sintetizados desde a década de 90 foram os amplificadores operacionais, filtros passivos, circuitos de controle, circuitos multiplicadores de tensão, sintonizadores de rádio frequência e outros [46].

No projeto de um circuito analógico, o objetivo é encontrar uma estrutura de um circuito e o valor de seus componentes que façam com que as especificações iniciais do circuito sejam atingidas, como por exemplo em relação ao ganho e atenuação, banda passante, resposta em frequência, deslocamento de fase, tempo de resposta, distorção harmônica, dentre outras. Pode-se também utilizar a CE em um circuito cuja a estrutura esteja fixa, para fazer uma pesquisa no espaço dos valores e tipos dos componentes que otimize o circuito. Esta abordagem é particularmente interessante para circuitos de grande complexidade, com a ajuda de softwares de simulação de circuitos eletrónicos do tipo SPICE ou semelhante [40].

---

[5] Semicondutor metal-óxido complementar, um tipo de tecnologia empregada na construção de circuitos integrados.

A representação em eletrónica analógica é feita em nível de componentes eletrónicos (condensador, resistência, indutor, transistor, diodo e outros), onde cada gene do cromossoma deve mapear um componente do circuito, de acordo com suas características/parâmetros, entre elas, o tipo, os nós de conexão e o valor (quando necessário), devem ser definidos pelo utilizador ao início da execução da evolução. A Figura 28 mostra o mapeamento de um gene em um resistência.

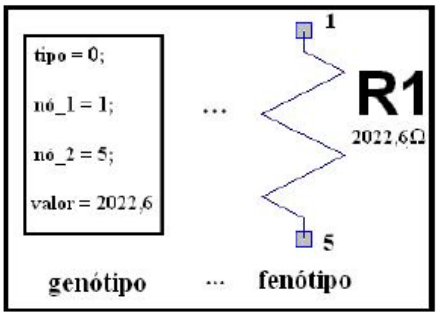


Figura 28 - Mapeamento genótipo-fenótipo do gene em resistência

[46]

O utilizador deve ter em mente o que deseja que o circuito analógico execute para estipular a função *fitness*, na qual, em aplicações analógicas envolve parâmetros como frequência, corrente, tensão, quantidade de componentes, custo dos componentes eletrónicos, entre outros [46].

A Figura 29 ilustra como é realizado o mapeamento de um circuito a nível de componentes, no qual cada gene é representado pelos nós (pontos) de conexão, valor do componente e tipo de componente.

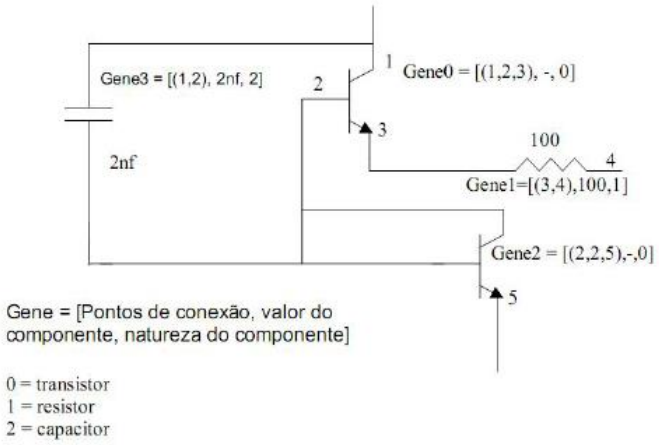


Figura 29 - Representação em nível de componentes

[46]

3.6. TELECOMUNICAÇÕES

A área das telecomunicações é de grande importância pelos serviços que disponibilizam. Algumas modificações foram necessárias na arquitetura das comunicações, pois houve o surgimento da Internet e de novos serviços. Como por exemplo, o aumento da necessidade de banda para o transporte de informação que vêm aumentando em grandes proporções, demanda uma tecnologia que possua todo o potencial necessário para prover a largura de banda necessária.

Genericamente, uma rede de transporte pode ser considerada como um conjunto de meios e equipamentos que transportam informações entre elementos de rede, os quais comutam ou roteiam a informação do cliente dentro da rede de transporte, levando os dados deste cliente ao destino apropriado de maneira confiável.

Com o desenvolvimento da tecnologia fotônica e da fibra ótica como meios de transmissão de capacidade de comutação a alta velocidade, no início dos anos 80, os sistemas começaram a dispor da fibra em substituição as linhas baseadas em cobre. Assim surgiram diversos padrões de transmissão como o *Synchronous Digital Hierarchy* (SDH - Hierarquia Digital Síncrona). Entretanto, nessas redes todas as operações de comutação, processamento e roteamento continuavam sendo feitas eletricamente, apenas os enlaces de transmissão passaram a pertencer ao domínio ótico.

No final da década de 90, com a demanda exponencial de capacidade de transmissão de dados por causa do avanço da Internet, surgiu a necessidade de incorporar também o controle baseado em *Internet Protocol* (IP), permitindo à rede de transporte ótica adaptar-se ao tráfego das suas redes clientes [47].

A sociedade está cada vez mais dependente de redes de telecomunicações eficientes e confiáveis, pois mais produtos e serviços são lançados no mercado utilizando essas redes. Com a crescente demanda gerada por essas aplicações e também pelo número cada vez maior de pessoas com acesso a essas tecnologias, os responsáveis por fornecer os serviços de telecomunicações, precisam investir na ampliação da infraestrutura, o que é problema importante de planeamento de redes [48].

Com esse crescimento de serviços de banda larga, telefonia fixa e móvel, os estudos de planeamento e recomposição das redes têm demandado esforços A complexidade das redes aumenta de acordo com as restrições impostas pela capacidade de investimentos e custos operacionais na obtenção de topologias a serem adotadas. Para resolver problemas de planeamento e recomposição de redes de telecomunicações é impossível não utilizar os recursos computacionais [47].

Com o aumento da demanda de serviços que requerem alta velocidade de comunicação, aumentou a instalação dos cabos de fibra ótica. Para este planeamento, uma alternativa válida é a aplicação da CE, que traz grande flexibilidade, principalmente pela facilidade de incorporar as restrições no problema e pela possibilidade de fornecer diversas soluções possíveis, na otimização do número de divisões em cada nó e a localização precisa de cada nó na rede física, de modo a minimizar a atenuação do sinal e a quantidade de fibras utilizadas. Também no roteamento dos dutos por onde os cabos devem passar, bem como os pontos de colocação dos divisores passivos, são fortemente restringidos pelos dutos/postes já existentes, o seu uso (outros cabeamentos pré-existentes), o seu acesso, o custo da construção de dutos específicos em função da localização geográfica dos clientes [40].

A CE tem chamado grande atenção também em aplicações de grande complexidade, como na otimização de alguns projetos de redes: roteamento de chamadas, RWA (*Routing and Wavelength Assignment*), gerência de redes. Esses algoritmos têm sucesso nos problemas de procura, em procedimentos de otimização e também uma boa aproximação para problemas específicos, mas em algumas aplicações acabam sendo inviáveis pelo tempo de execução computacional [47].

O AG vem sendo aplicado em problemas da alocação de frequências (*Frequency Assignment Problem* - FAP), uma aplicação particular nesta categoria é a alocação de canais para redes de comunicação de telefonia celular. A dimensionalidade pode atingir 6000 células e 12000 restrições, tornando-se um problema de difícil solução por métodos convencionais. Resultados mostram que esses algoritmos obtêm melhores resultados e em menos tempo para problemas reais quando comparado com algoritmos tradicionais. A utilização da CE torna-se cada vez mais atrativa à medida que a dimensão do problema aumenta, pois os métodos computacionais se tornam ineficientes [40].

3.7. CONTROLADORES

O projeto de sistemas de controle em geral é baseado em métodos algébricos clássicos que são eficientes para a maioria das aplicações práticas. Os algoritmos da CE podem ser aplicados em problemas de controle linear e não-linear e de sistemas dinâmicos que envolvam uma complexidade mais elevada [40].

Os controladores PID (Proporcional, Integral e Derivativo), e suas variações, são métodos convencionais de controle usados em muitos processos automatizados. São muito utilizados devido a sua estrutura simples, robustez, reduzido número de parâmetros a serem configurados, conhecimento intuitivo sobre o desempenho dos parâmetros e fácil implementação em sistemas computacionais discretos. Entretanto, a teoria de controle clássica, falha no tratamento de alguns processos complexos, devido principalmente a presença de não-linearidades e comportamentos variantes no tempo [49].

Os algoritmos genéticos apresentam-se como uma ferramenta útil para a determinação das constantes do controlador PID, uma vez que métodos tradicionais baseiam-se em tentativa e erro e alguns métodos como Ziegler-Nichols, que nada mais são do que tentativas iniciais para um posterior ajuste fino. Para este tipo de problema, cada indivíduo da população apresenta três genes cada um representando uma constante do controlador e apresentam a representação real (Kp, Ki, Kd)[6]. A função *fitness* é específica para cada aplicação e deve representar o comportamento dos cromossomas, como representar os parâmetros do controlador, devendo fornecer a informação do quanto é adequado o controlador quando sintonizado com os parâmetros escolhidos pelo algoritmo. A Figura 30 mostra como o AG é aplicado na malha para o ajuste do controlador PID [50].

[6] Parâmetros do controlador PID, são respectivamente os ganhos proporcional, integral e derivativo.

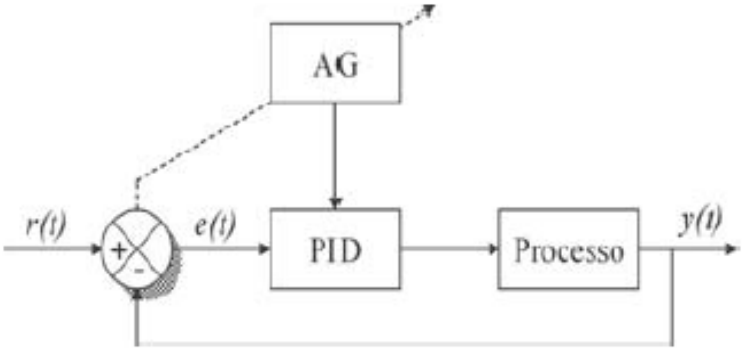


Figura 30 – Ajuste de um controlador PID através do AG
[50]

A CE também tem sido bastante aplicada no projeto de controladores com lógica *fuzzy* (ou lógica nebulosa). O controle *fuzzy* é uma técnica de controle de parâmetros que trata os valores dos parâmetros utilizando graus de pertinência, foi proposta por Zadeh em 1965 com o objetivo de inserir graus de incerteza na lógica tradicional booleana e permitir a inferência humana em conceitos e conhecimentos que não estão bem definidos. As partes básicas de um controlador *fuzzy* são a base de conhecimento, as regras fuzzy e o mecanismo de inferência. O mecanismo de inferência é utilizado para avaliar o estado atual do sistema e as regras são propostas de acordo com o estado identificado sendo utilizadas para controlar os valores dos parâmetros [51].

A aplicação desses algoritmos em sistemas de controle *fuzzy* pode ser dividido em duas categorias: otimização de funções de pertinência e otimização da base de regras.

- Otimização de funções de pertinência: definir a forma e os parâmetros do conjunto total de funções de pertinência de um controlador *fuzzy*;
- Otimização da base de regras**:** é mais complexo, pois a questão crítica é obter o conjunto de regras menor possível e o mais adequado possível para a função de controle, mas num projeto de controladores com um grande número de variáveis e de funções de pertinência, onde, as regras possíveis são calculadas de acordo com o número de variáveis elevado ao número de funções de pertinência. Ou seja, quanto maior o número de regras, mais tempo de processamento será consumido, podendo até mesmo inviabilizar o uso em tempo real do controlador [40].

4. CASO DE ESTUDO

No Capítulo 4 será abordado um caso de estudo do algoritmo genético aplicado na síntese de circuitos digitais combinatórios, com base em três diferentes referências de autores, respectivamente, [19], [8] e [41], analisando cada uma delas e comparando-as.

4.1. INTRODUÇÃO

O caso de estudo analisará três aplicações dos algoritmos genéticos na síntese de circuitos digitais combinatórios. O objetivo é comparar circuitos que realizam mesmas funções ou diferentes, levando em consideração o número de entradas, tipo de representação utilizada na codificação do problema, sua complexidade, os parâmetros utilizados nas simulações. Apesar das três referências obterem diferentes formas de implementação do AG, a partir da forma com que foram abordadas consegue-se realizar algumas comparações e retirar conclusões simples, mas de grande valia.

As seções 4.2, 4.3 e 4.4 realizam um breve levantamento sobre cada uma das referências, respectivamente, [19], [8] e [41]. Mostrando as definições do problema, dentre elas os operadores implementados, os parâmetros utilizados e os circuitos implementados por cada

um dos autores. Na seção 4.5 faz-se as comparações possíveis entre as referências, analisando-as de diferentes maneiras.

No Anexo C contém os resultados obtidos das referências estudadas que darão base para as comparações desse caso de estudo.

4.2. SÍNTESE DE SISTEMAS DIGITAIS POR COMPUTAÇÃO EVOLUTIVA

Esta seção faz um breve resumo do estudo da aplicação dos algoritmos genéticos na síntese de circuitos digitais combinatórios do trabalho de [19].

Nele foi realizado uma revisão bibliográfica da computação evolutiva, como os algoritmos evolutivos e os algoritmos da inteligência dos enxames. Realizou a síntese automática de circuitos lógicos combinatórios, baseada em três algoritmos, respectivamente, em algoritmos genéticos, algoritmos meméticos e algoritmos de otimização por enxame de partículas. Para o caso de estudo será utilizado apenas o capítulo 3, que refere-se à síntese de circuitos combinatórios com base nos algoritmos genéticos. Nas subseções seguintes consta a definição do problema em estudo e os circuitos implementados pelo mesmo.

4.2.1. DEFINIÇÃO DO PROBLEMA

Foram definidos quatro conjuntos de portas lógicas, de modo que gerassem circuitos com os componentes de cada um desses conjuntos, com o intuito de implementar circuitos funcionais com a menor complexidade possível.

A Tabela 4 apresenta os quatro conjuntos de portas lógicas utilizados. O *Gset2* é o conjunto mais simples, o *Gset3* e *4* são de média complexidade, e o *Gset6* é o mais complexo, onde *WIRE*, significa uma ligação direta, sem a utilização de portas lógicas.

Tabela 4 – Conjunto de portas lógicas

Conjunto	Portas Lógicas
Gset 2	{AND, XOR, WIRE}
Gset 3	{AND, OR, XOR, WIRE}
Gset 4	{AND, OR, XOR, NOT, WIRE}
Gset 6	{AND, OR, XOR, NOT, NAND, NOR, WIRE}

Os circuitos são codificados como uma matriz retangular de células lógicas. Cada célula apresenta três genes, duas entradas e o tipo de porta lógica, representados da seguinte maneira: *<entrada1><entrada2><tipo de porta>*. O cromossoma é constituido por conjuntos de três genes, de acordo com o tamanho da matriz. A Figura 31 ilustra um cromossoma que representa uma matriz 3x3, e a Figura 32 mostra a representação de um cromossoma referente a mesma.

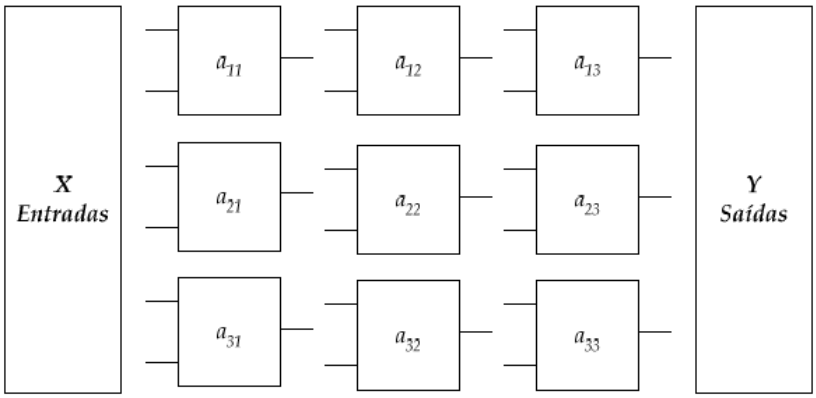


Figura 31 - Matriz 3x3 que representa um circuito
[19]

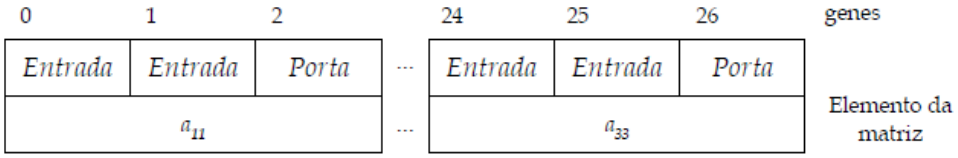


Figura 32 - Cromossoma referente a matriz 3x3
[19]

A população inicial de circuitos (vetores/indivíduos) foi gerada aleatoriamente. As gerações sucessivas de novos vetores são reproduzidos com base na função de aptidão. Os três operadores foram implementados:

- SELEÇÃO: realizada pelo método de torneio, onde são selecionados três indivíduos que vão competir entre si, vencendo o indivídu com melhor aptidão.

- CRUZAMENTO: os indivíduos vão sendo agrupados aos pares de uma forma aleatória e, em seguida, foi efetuado o cruzamento num ponto, que é apenas permitido entre células de forma a manter a integridade do cromossoma.

- MUTAÇÃO: altera as características totais de uma dada célula da matriz, modificando o tipo de porta lógica e as duas entradas.

Além disso, foi implementado o elitismo, mantendo as melhores soluções para a próxima geração.

Definiu-se o número de indivíduos para se proceder à criação da população inicial em 3000 indivíduos. Esta população mantém o mesmo tamanho ao longo das gerações. A taxa de cruzamento foi adaptada em 95% e a taxa de mutação em 5%. Esses parâmetros foram utilizados para todos os circuitos implementados.

A função de avaliação (*fitness*) foi dividida em duas partes (f_1 e f_2) que medem respectivamente, a funcionalidade e a simplicidade. Primeiramente, comparou-se o circuito gerado pelo AG com os valores esperados de acordo com a tabela verdade, correndo-a *bit* a *bit* (f_1). O circuito é dito funcional quando for atingido o f_{10}, após isso, o AG tenta gerar circuitos mais simples, ou seja, com menor número de portas lógicas. O f_2 avalia a simplicidade, e é incrementado de 1 por cada porta lógica do tipo *wire* existente no circuito, de acordo com as equações abaixo:

$$f_{10} = 2^{ni} \times no \tag{4.1}$$

$$f_1 = f_1 + 1 \; se \; \{bit\, i \, de \, Y\} = \{bit\, i \, de \, Y_R\}, i = 1,..., \; f_{10} \tag{4.2}$$

$$f_2 = f_2 + 1 \; se \; tipode \, porta = wire \tag{4.3}$$

$$F = \begin{cases} f_1, & F < f_{10} \\ f_1 + f_2, & F \geq f_{10} \end{cases} \tag{4.4}$$

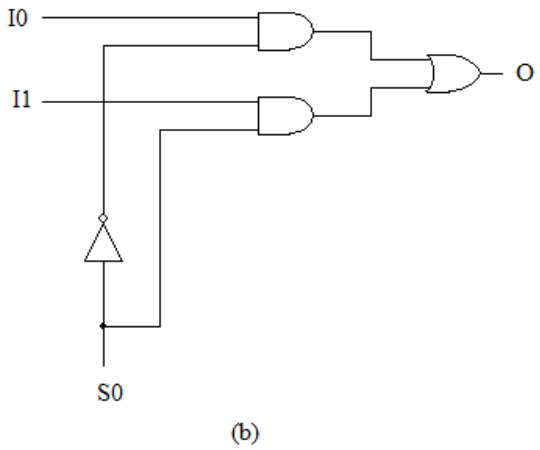
4.2.2. CIRCUITOS IMPLEMENTADOS

Nesta referência foram implementados quatro circuitos lógicos combinatórios diferentes, que são: o multiplexador 2 para 1, o somador completo de um *bit*, o circuito de teste de paridade de quatro *bits* e o multiplicador de dois *bits*.

- O multiplexador 2 para 1 apresenta uma tabela verdade de três entradas {S0, I1, I0}, onde S0 é a variável de seleção, e uma saída {O}, o que origina uma matriz de dimensões 3x3 e o comprimento do vetor que representa cada um dos circuitos (o tamanho do cromossoma) é 27. A Figura 33 mostra a tabela verdade e o circuito equivalente desse multiplexador, e a equação 4.5 é a expressão simplificada do circuito.

S0	O
0	I0
1	I1

(a)



(b)

Figura 33 – (a) Tabela verdade do multiplexador 2 para 1 (b) Circuito equivalente

$$O = \overline{S0}.I0 + S0.I1 \tag{4.5}$$

- O somador completo de um *bit*, apresenta uma tabela verdade de três entradas {A, B, Cin} e duas saídas {S, Cout} como mostra a Tabela 5, e as equações 4.6 e 4.7 que

apresentam as simplificações pelo mapa de Karnaugh das saídas, originando uma matriz de dimensões 3x3, portanto o tamanho do cromossoma é 27.

Tabela 5 – Tabela verdade do somador completo de um *bit*

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = A \oplus B \oplus Cin \tag{4.6}$$

$$Cout = A.B + A.Cin + B.Cin \tag{4.7}$$

- O circuito de teste de paridade de quatro *bits*, possui uma tabela verdade de quatro entradas {A3, A2, A1, A0} e uma saída {P} dando origem a uma matriz de dimensões 4x4, logo o comprimento do vetor é 48. Se o número de ‘1’s nas entradas for par ele coloca um ‘0’ na saída e se o número de ‘1’s for ímpar ele coloca um ‘1’ na saída, como mostra a **Tabela 6**.

Tabela 6 – Tabela verdade do circuito de teste de paridade

A3	A2	A1	A0	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

- O multiplicador de dois *bits* apresenta uma tabela verdade de quatro entradas {A1, A0, B1, B0} e quatro saídas {C3, C2, C1, C0}, como ilustra a **Tabela 7**. Possui a matriz correspondente de dimensão 4x4, logo o tamanho do cromossoma é 48.

Tabela 7 – Tabela verdade do multiplicador de 2 *bits*

A1	A0	B1	B0	C3	C2	C1	C0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Para todos os casos estudados, observou-se que o conjunto de portas lógicas *Gset2* e *Gset3* foram os que obtiveram melhores resultados, tanto pela média do número de gerações para obter a solução, quanto pela média da função de aptidão obtida.

4.3. UMA FERRAMENTA ALTERNATIVA PARA A SÍNTESE DE CIRCUITOS LÓGICOS USANDO A TÉCNICA DE CIRCUITO EVOLUTIVO

Esta seção faz uma síntese do estudo do trabalho de [8], da aplicação dos algoritmos genéticos na síntese e otimização de circuitos digitais (combinatórios e sequenciais), utilizando a teoria da CE e como plataforma os dispositivos reconfiguráveis.

Nele foi realizado um breve levantamento sobre os AG e seus operadores, uma revisão bibliográfica da aplicabilidade dessa técnica. Apresentou o conceito de circuito evolutivo e sua utilização na síntese de circuitos digitais. Estudou-se também a evolução dos circuitos digitais por portas lógicas com codificação binária e codificação por números inteiros e a representação por mapa de fusíveis, aplicadas em circuitos combinatórios básicos e circuitos sequenciais. Após a evolução do AG, a partir do melhor cromossoma encontrado durante o processo de evolução do AG e com base nos dados contidos nesse

cromossoma é gerado o código em VHDL que servirá para programar um dispositivo PLD[7].

Para o caso de estudo será utilzado apenas o capítulo 4 e os resultados experimentas do capítulo 5, que refere-se à sìntese de circuitos combinatórios, a partir de uma representação por mapas de fusíveis e portas lógicas com codificação binária.

4.3.1. DEFINIÇÃO DO PROBLEMA

Inicialmente, foram realizados estudos para a síntese de circuitos a partir de uma representação por mapas de fusíveis. Procurando não somente a síntese, mas também a otimização de circuitos mais complexos optou-se por usar portas lógicas com uma uma codificação binária. Com o objetivo de melhorar o desempenho do AG, usou-se também uma representação por portas lógicas com codificação por números inteiros.

Na representação de portas lógicas com codificação binária também foi utilizada a representação de portas lógicas na forma matricial .

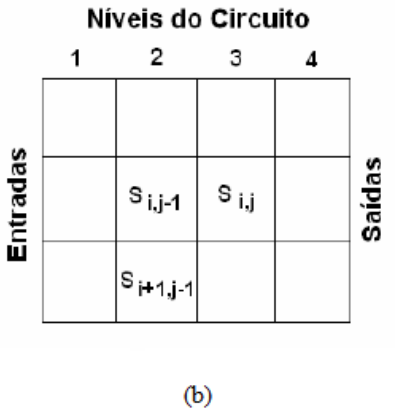
Conforme a Figura 34, em (a) é representado uma tabela com a codificação utilizada e a equivalência das portas. Na coluna 1 da tabela, mostra-se o número decimal associado. Na coluna 2, os três primeiros *bits* da esquerda para a direita definem a porta, e o quarto *bit* determina como serão realizadas as conexões da porta (restringe-se o modelo a portas de duas entradas). Na coluna 3 estão associados os tipos de portas que estão sendo utilizadas. Nesta representação os estudos foram restritos a uma matriz bidimensional e cada elemento da matriz é uma porta lógica limitada a duas entradas (NOT, AND, OR e XOR) ou fio de ligação (FIO). Em (b) é representado matricialmente o circuito, de forma que uma porta receba nas suas entradas qualquer porta da coluna anterior. Por exemplo, se uma porta S está na posição $S_{i,\,j}$, onde *j* indica o nível da porta (coluna) e *i* indica a posição na linha, então, uma entrada é proveniente da posição $S_{i,\,j-1}$ (mesma linha e coluna anterior) e a

[7] Os testes foram realizados em PLDs da família FLEX 10K (EPF10K70) que possui 118000 portas, 3744 elementos lógicos e o número máximo de 358 pinos I/O.

outra entrada ou vem de $S_{i+1,\,j\,-1}$ (coluna anterior e uma linha abaixo) ou de $S_{i-1,\,j-1}$ (coluna anterior e uma linha acima).

Decimal	Binário	Representação
0	000 0 e 000 1	FIO
1	100 0 e 100 1	NOT
2	001 0	AND Segunda entrada abaixo
3	010 0	OR Segunda entrada abaixo
4	011 0	XOR Segunda entrada abaixo
5	001 1	AND Segunda entrada acima
6	010 1	OR Segunda entrada acima
7	011 1	XOR Segunda entrada acima

(a)



(b)

Figura 34 – (a) Codificação das Portas Lógicas (b) Representação matricial do circuito [8]

Após definida a estrutura do cromossoma, a população inicial foi gerada aleatoriamente, e com base na função adaptação aplicou-se a seleção. Os três operadores genéticos foram implementados:

- SELEÇÃO: pelo método do torneio, selecionando aleatoriamente três indivíduos, e o melhor deles foi escolhido.

- CRUZAMENTO: recombinação de um simples ponto em um par de indivíduos, gerando novos descendentes.

- MUTAÇÃO: aplicada a do tipo indutiva. A estratégia de mutação implementada avalia todos os genes no cromossoma com a taxa de mutação t_m, mas para reduzir o esforço computacional deste processo, calculou-se a quantidade de genes a serem alterados pela mutação utilizando a equação 4.7, onde *muta* é a quantidade de genes que serão alterados na população, variando t_m entre 1% e 5%, Tc indica a quantidade de genes do cromossoma *i* e foi calculado de acordo com a equação 4.8, N é o número de indivíduos da população, *ne* o número de entradas e *ns* quantidade de saídas. Após definida a quantidade de genes, escolheu-se de forma aleatória quais sofreriam a mutação.

$$muta = t_m \cdot (Tc \cdot N) \tag{4.7}$$

$$Tc = 2x^2, \quad x = ne,\ se\ ne > ns \qquad ou \qquad x = ns,\ se\ ns > ne \tag{4.8}$$

Também foi implementado o elitismo, separando, com base no *fitness*, o melhor indivíduo para a geração seguinte.

A função de avaliação foi realizada em duas etapas. A primeira visa a obtenção de um circuito lógico correto através do cálculo da soma de acertos da sequência de saída de uma tabela verdade (ou expressão booleana), de acordo com a equação 4.9, onde a função *f(x)* indica o número de acertos na tabela verdade (x_i=0, falso; x_i=1, verdadeiro) e *n* é o número de linhas da tabela verdade.

$$f(x) = \sum_{i=0}^{n-1} x_i \tag{4.9}$$

Após obtido o circuito correto, buscou-se na segunda etapa a minimização de portas lógicas. Para isso trocou-se as portas lógicas por fios enquanto o desempenho especificado pela tabela verdade do circuito não fosse alterado. Conforme a equação 4.10 que reescreve a função adaptação, na qual aumentando o número de fios obtém-se a função *w(x)*, que é um valor inteiro equivalente à quantidade de fios adicionados.

$$f_a(x) = f(x) + w(x) \tag{4.10}$$

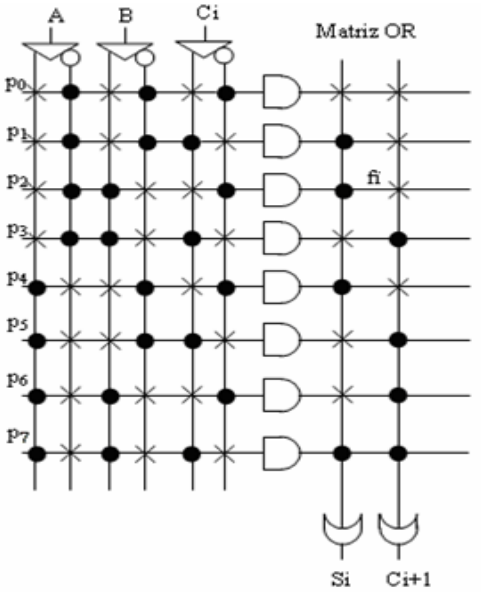
4.3.2. CIRCUITOS IMPLEMENTADOS

Para a representação por mapas de fusíveis, foram implementados dois circuitos combinatórios: um somador completo e um detector de números primos.

- O somador completo obtêm três entradas e duas saídas, de acordo com o mapeamento de fusíveis, possui 3 entradas (3 inversores), 7 portas AND e 2 saídas, totalizando 56 genes, obtido de acordo com a equação 3.1. Como são 8 linhas que definem a tabela verdade, os acertos devem totalizar 8 pontos. Para reduzir o esforço computacional na simulação, omitiu-se do cromossoma o caso de entradas zero da tabela verdade que resultasse em saída zero. A Figura 35 apresenta a tabela verdade e o mapeamento de fusíveis para o somador completo.

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a)



(b)

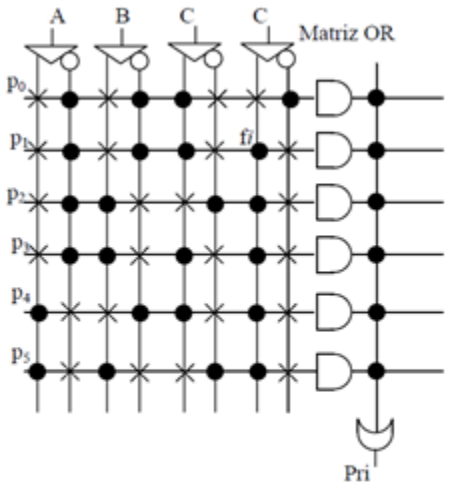
Figura 35 – (a) Tabela verdade (b) Mapeamento de fusíveis, do somador

Para as simulações considerou-se uma população de 250 indivíduos gerados aleatoriamente, sempre com 50 iterações. Foi realizado vários testes, foi encontrado soluções válidas com outros valores, contudo foi com a taxa de mutação em 10% e taxa de cruzamento em 80%, que resultou na maior incidência de cromossomas corretos. Utilizando taxas de cruzamento e mutação diferentes, respectivamente, 70% e 1%, na tentativa de avaliação do comportamento e convergência do algoritmo, não obteve sucesso, pois houve uma saturação da população por falta de diversidade.

- O detector de número primos de 4 *bits* (quatro entradas e uma saída), exibe uma saída ativa quando sua entrada binária corresponde a um número primo decimal, ou seja, números que são divisíveis por ele mesmo ou pela unidade, excluído o 1. De acordo com o mapeamento de fusíveis e a equação 3.1, possui 4 entradas (4 inversores), 6 portas AND e 1 saída, totalizando 54 genes. Como são 16 linhas que definem a tabela verdade, os acertos devem totalizar 16 pontos. A Figura 36 apresenta a tabela verdade e o mapeamento de fusíveis para o detector de número primos.

A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

(a)



(b)

Figura 36 - (a) Tabela verdade (b) Mapeamento de fusíveis, do detector de números primos

Com 250 indivíduos, as iterações foram mantidas em 50, com a taxa de cruzamento de 80% e a de mutação em 20%, obteve-se cromossomas corretos. Ao contrário, para taxa de recombinação em 70% e a taxa de mutação em 1%, obteve-se 15 pontos dos 16 necessários.

Para a representação por portas lógicas com codificação binária foram realizados a síntese de circuitos combinatórios de funções lógicas simples (de três entradas com uma saída e quatro entradas com uma saída). Com esta mesma representação foram implementados exemplos de circuitos sequenciais, uma máquina de estados com dois e cinco estados. Neste caso de estudo será apenas estudado a síntese dos circuitos combinatórios.

- CIRCUITO COMBINATÓRIO (TRÊS ENTRADAS E UMA SAÍDA): foi implementado um circuito combinatório com três entradas e uma saída, cuja tabela verdade é mostrada na Tabela 8. Os parâmetros do AG foram calibrados em uma população de 500 indivíduos, 200 iterações (totalizando cem mil análises), taxa de cruzamento em 80% e de mutação em 40% e tamanho do cromossoma igual a 72 genes.

Tabela 8 – Tabela verdade circuito 3 entradas e 1 saída

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- CIRCUITO COMBINATÓRIO (QUATRO ENTRADAS E UMA SAÍDA): foi implementado um circuito combinatório com quatro entradas e uma saída, cuja tabela verdade é mostrada na Tabela 9. Os parâmetros do AG foram calibrados em uma população de 500 indivíduos, taxa de cruzamento em 80% e de mutação em 40%.

Tabela 9 – Tabela verdade circuito 4 entradas e 1 saída

A	B	C	D	S
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Para a representação por portas lógicas com codificação inteira foram realizados a síntese para os mesmo circuitos combinatórios de funções lógicas simples, mais para um somador completo.

- CIRCUITO COMBINATÓRIO (TRÊS ENTRADAS E UMA SAÍDA): foi implementado um circuito combinatório com três entradas e uma saída, cuja tabela verdade é mostrada na Tabela 8. Os parâmetros do AG foram calibrados em uma população de 500 indivíduos, 500 iterações, taxa de cruzamento em 50% e de mutação em 5%. Para três variáveis na função, a função objetivo deve totalizar um número de acertos igual a oito, compõem uma matriz de ordem 3x3 que determina o tamanho do cromossoma em 9 genes.

- CIRCUITO COMBINATÓRIO (QUATRO ENTRADAS E UMA SAÍDA): foi implementado um circuito combinatório com quatro entradas e uma saída, cuja tabela verdade é mostrada na Tabela 9. Os parâmetros do AG foram calibrados em uma população de 1000 indivíduos, taxa de cruzamento em 50% e de mutação em 2%. O cromossoma tem 32 genes definindo uma matriz 4x8.

- SOMADOR COMPLETO: realiza a soma entre dois números binários (A e B) de um *bit* cada um, conforme a Tabela 10. O cromossoma foi mantido com 18 genes definindo uma matriz bidimensional de 3x6. População inicial de tamanho 1000, número de iterações em 500, taxa de cruzamento em 50% e taxa de mutação em 3%.

Tabela 10 – Tabela verdade do somador completo

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4.4. SÍNTESE DE CIRCUITOS DIGITAIS POR EVOLUÇÃO DE CIRCUITOS

Esta seção faz uma resumo do estudo da síntese de circuitos digitais por evolucão de circuitos, usando conceitos de *hardware* evolutivo, do trabalho de [41].

Foi discutido os conceitos básicos sobre *Hardware* evoulutivo, a codificação do problema e uma representação por mapas de fusíveis. Foi utilizado o AG para sintetizar dois circuitos digitais combinatórios, um decodificador e um multiplexador.

4.4.1. DEFINIÇÃO DO PROBLEMA

Após definida a estrutura do cromossoma, a população inicial foi gerada aleatoriamente, e com base na função adaptação aplicou-se a seleção. Os três operadores genéticos foram implementados.

- SELEÇÃO: método por torneio, no qual escolheu-se aleatoriamente três cromossomas e procurou-se pelo maior *fitness.*

- CRUZAMENTO: foi uniforme entre indivíduos escolhidos aleatoriamente.

- MUTAÇÃO: simples, foi realizada a troca de *bits* de um gene de um indivíduo aleatoriamente.

A função de avaliação foi obtida através da porcentagem do número de acertos da tabela verdade do circuito em estudo.

O tamanho da população, a taxa de mutação, a taxa de cruzamento e o critério de paragem foram estabelecidos em função do espaço de pesquisa do problema. Para os ambos os circuitos, considerou-se uma população de 200 indivíduos gerados aleatoriamente e controlou-se a convergência através das taxas de cruzamento e mutação.

4.4.2. CIRCUITOS IMPLEMENTADOS

Nesta referência foram implementados dois circuitos digitais combinatórios, baseados na representação por mapeamento de fusíveis: um decodificador 2x4 e um multiplexador 4_1.

- O decodificador 2x4 possui duas entradas e quatro saídas, no qual para cada entrada selecionada apenas uma saída é disponibilizada, como mostra a Figura 37. Pela representação com mapas de fusíveis para o decodificador 2x4, utilizando a equação 3.1, contabilizou-se 2 entradas (2 inversores), 4 portas ANDs e 4 saídas, totalizando 32 genes, mostrados na Figura 38.

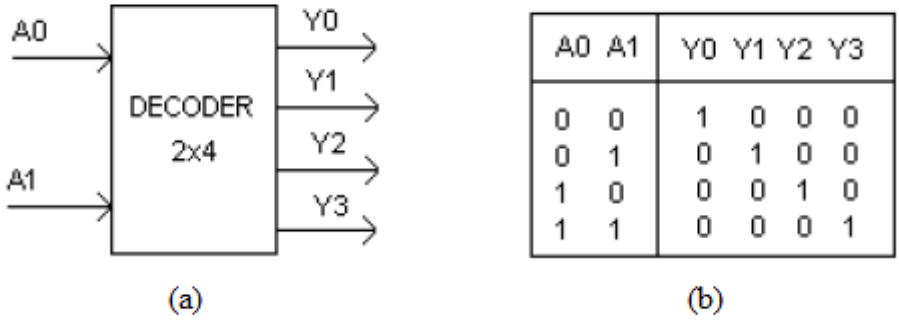


Figura 37 – (a) Símbolo lógico, (b) Tabela verdade, do decodificador 2x4
[41]

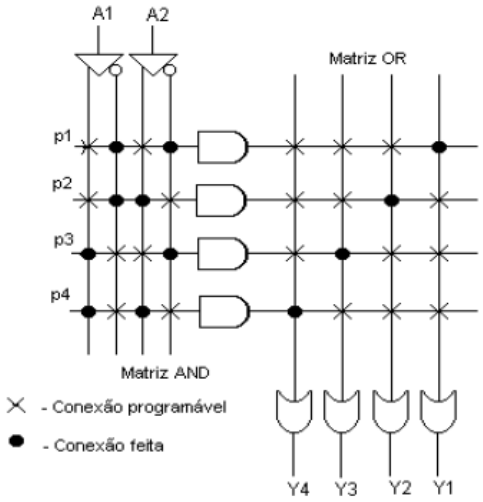


Figura 38 – Estrutura simplificada do PLA para o decodificador 2x4
[41]

Mantendo fixos o tamanho da população em 200 indivíduos e o número de gerações em 50, para uma taxa de cruzamento relativamente alta de 70% e taxa de mutação baixa de 1%, ocorreu uma saturação da população e o AG não chegou à configuração do circuito antes de 40 iterações. E para taxas de cruzamento e mutação elevadas, respectivamente, 65% e 50%, houve uma diversificação da população, sendo mais rápida a convergência e a obtenção de uma configuração de sucesso para o circuito.

- O multiplexador 4_1 possui quatro entradas, duas entradas de seleção e uma saída, como mostra a Figura 39. A Figura 40 apresenta a representação por mapas de fusíveis para o decodificador 2x4, utilizando a equação 3.1, contabilizou-se 2 entradas (2 inversores), 4 portas ANDs, 1 saída e 4 canais de seleção, totalizando 36 genes

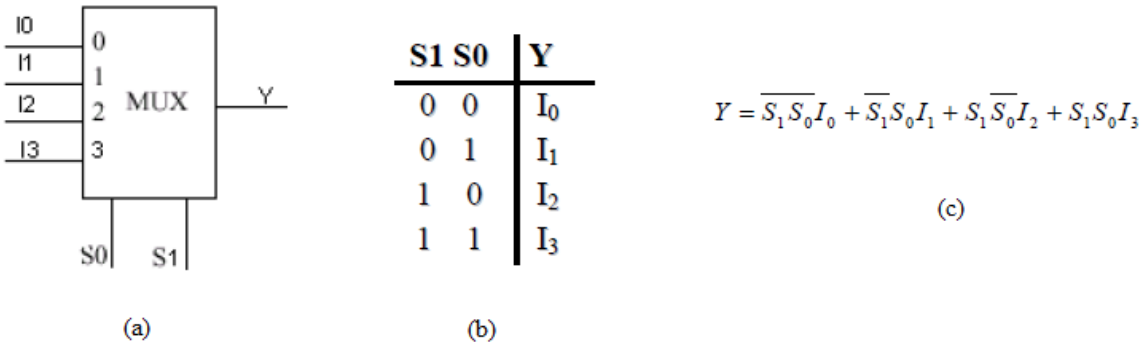


Figura 39 – (a) Símbolo gráfico (b) Tabela verdade (c) Expressão Booleana, para o multiplexador [41]

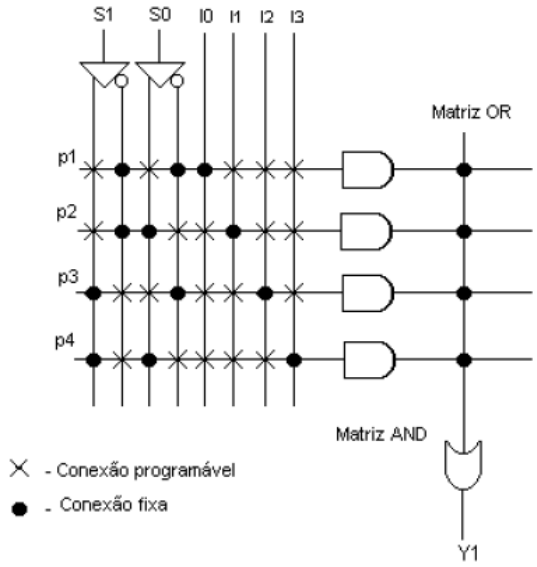


Figura 40 - Mapeamento de fusíveis do multiplexador 4_1 [41]

Como apresenta um espaço de pesquisa maior, as possibilidades de acertos com o AG são bem maiores, possibilitando obter soluções de boa qualidade na maioria das simulações realizadas. Os resultados foram retirados com a taxa de cruzamento e mutação, respectivamente, 65% e 6%.

4.5. COMPARAÇÕES E CONCLUSÕES DAS REFRÊNCIAS

Nesta seção foram realizadas algumas comparações entre as referências citadas acima, através da geração de tabelas e gráficos para melhor entendimento.

A primeira comparação realizada foi entre os quatro circuitos representados por mapeamento de fusíveis para posterior geração do código VHDL, de [8] e [41], entre eles: somador completo, detector de números primos de 4 *bits*, decodificador 2x4 e um multiplexador 4_1.

A Tabela 11 mostra um resumo dos quatro circuitos comparados. Através dela percebe-se que elevando a taxa de mutação (t_m), o AG resulta melhores soluções, visto que para pequenas taxas resultou-se na saturação da população por falta de diversidade da mesma.

Tabela 11 – Comparação dos parâmetros entre circuitos

SOMADOR (3 entradas/ 2 saídas): **1º)** t_c=80% e t_m=10% 50 iterações/ 250 indivíduos *É possível encontrar soluções válidas com outros valores, entretanto estes foram os que resultaram na maior incidência de cromossomas corretos. **2º)** t_c=70% e t_m=1% 50 iterações/ 250 indivíduos *Houve uma saturação da população por falta de diversidade.	**DET. NºPRIMOS (4 entradas/ 1 saídas):** **1º)** t_c=80% e t_m=20% 50 iterações/ 250 indivíduos *É possível encontrar soluções válidas com outros valores, entretanto estes foram os que resultaram na maior incidência de cromossomas corretos. **2º)** t_c=70% e t_m=1% 50 iterações/ 250 indivíduos *Obtendo 15 pontos dos 16 necessários
DECODIFICADOR 2X4 (2 entradas/ 4 saídas): **1º)** t_c=70% e t_m=1% 50 iterações/ 200 indivíduos *Saturação da população **2º)** t_c=65% e t_m=50% 50 iterações/ 200 indivíduos *Diversificação da população, sendo mais rápida a convergência e a obtenção de uma configuração de sucesso para o circuito.	**MULTIPLEXADOR 4_1 (4 entradas/ 2 entradas seleção/1 saída):** **1º)** t_c=65% e t_m=6% 25 a 30 iterações/ 200 indivíduos *Convergência rápida em função do número de gerações. As possibilidades de acertos são bem maiores, sendo possível obter soluções de boa qualidade na maioria das simulações realizadas.

Como visto na teoria dos algoritmos genéticos, utiliza-se a taxa de mutação reduzida para que a população não diversifique em demasia. Mas, oberserva-se que, para pequenas

populações, têm-se a necesssidade de aumentar essa taxa para não ocasionar o saturamento da população pela falta da diversidade.

A Figura 41, apresenta um gráfico correspondente ao número de iterações em função das taxas de acertos da tabela verdade (*fitness*), esses dados foram coletados dos resultados presentes no Anexo C, baseados nas simulações realizadas que geraram circuitos evoluídos. Percebe-se que a complexidade do problema aumenta (consequentemente o custo computacional também) de acordo com o aumento do número de entradas do circuito, pois as possibilidades da tabela verdade aumentam de acordo com 2^n (onde *n* é o número de entradas do circuito).

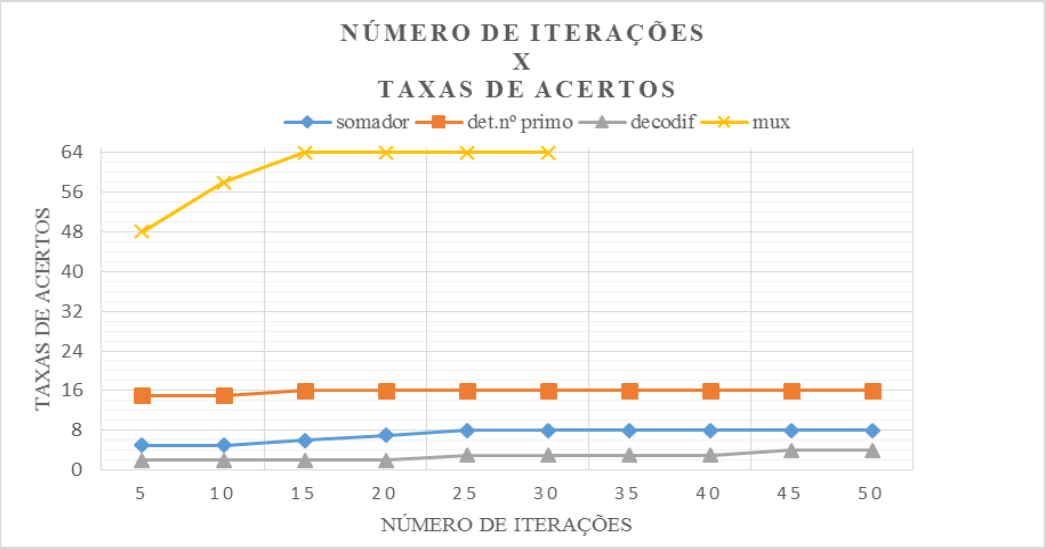


Figura 41 - Número de iterações x taxas de acertos

A Figura 42 mostra o número de iterações em função das taxas de acertos para as simulações nas quais ocorreram a saturação da população pela falta de diversidade. Oberva-se claramente isto, pois nenhuma delas atingiu o número de acertos necessários para satisfazer a função avaliação. Com poucas iterações pode-se observar que o número de acertos permanecem estagnados e não conseguem evoluir para um circuito correto.

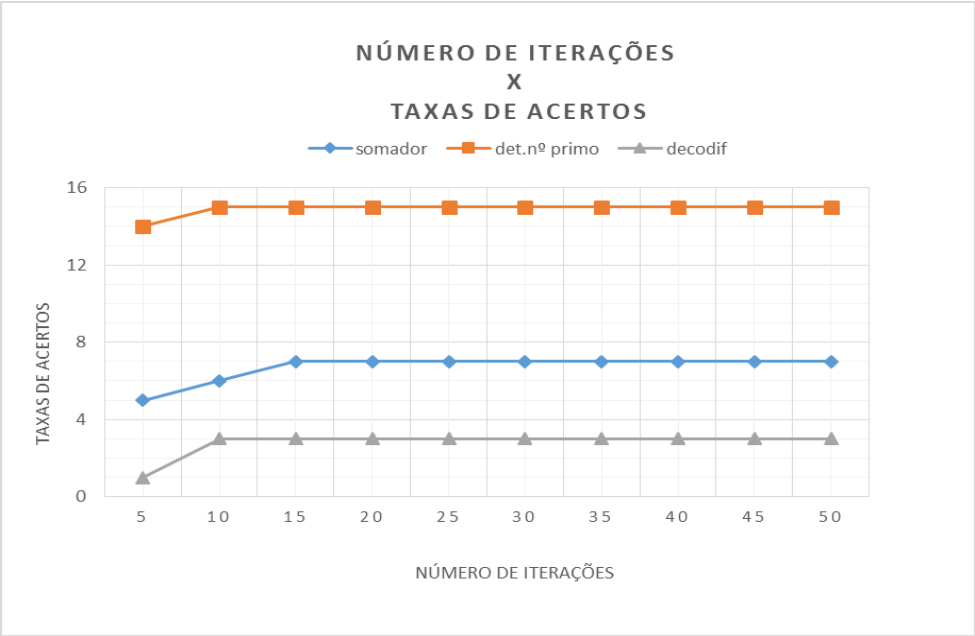


Figura 42 – Número de iterações x taxas de acertos

A segunda comparação é entre circuitos combinatórios de quatro entradas (o circuito combinatório qualquer e o detector de números primos de [8]). Sabendo que as simulações foram realizadas com os seguintes parâmetros: para o detector de números primos, população com 250 indivíduos, t_c=80%, t_m=20% e 50 iterações. E para o outro circuito combinatório, população com 500 indivíduos, t_c=80%, t_m=40% e 500 iterações. A partir da Figura 43, observou-se que, para o mesmo número de entradas, o circuito com o maior número de indivíduos e com a taxa de mutação elevada, fez com que a população se tornasse aleatória, necessitando de um maior número de iterações para convergir ao resultado, ou seja, alcançar o *fitness* do AG.

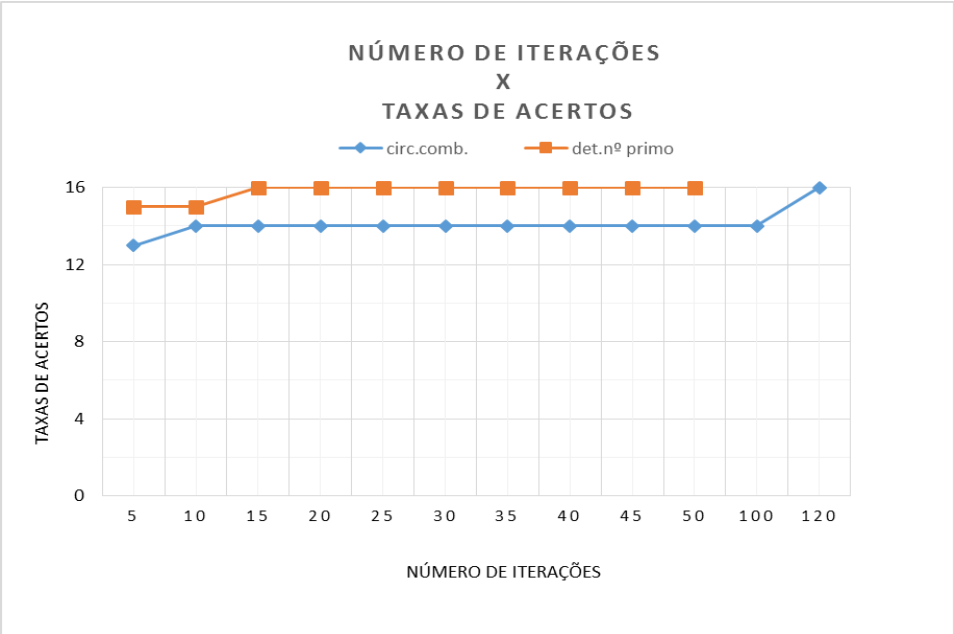


Figura 43 – Número de iterações x taxas de acertos

Para a terceira análise, fez-se a comparação entre os somadores completos com representação à nível de portas lógicas, de [8] e [19], em relação à síntese do números de portas lógicas que resultaram no circuito equivalente, comparando-os com o resultado obtido a partir do Mapa de Karnaugh.

De acordo com a técnica tradicional de síntese de circuitos, no somador de 1 *bit* (três entradas e duas saídas), o circuito final contém 3 AND's, 1 OR e 2 XOR's, totalizando 6 portas ao todo. No de [8], obeteve-se 4 XOR's e 1 AND, totalizando 5 portas lógicas. Em [19], também obeteve-se 5 portas lógicas ao todo, mas 3 XOR's e 2 AND's. A Tabela 12 mostra as equações obtidas em cada um dos casos.

Tabela 12 – Comparação entre diferentes técnicas

Mapa de Karnaugh	**Sobrinho, 2007 [8]**	**Reis, 2007 [19]**
$S = A \oplus B \oplus Cin$ $Cout = B.Cin + A.Cin + A.B$	$S = A \oplus B \oplus Cin$ $Cout = [(B \oplus Cin) + (A \oplus B)] \oplus (A \oplus B \oplus Cin)$	$S = A \oplus B \oplus Cin$ $Cout = (A.B) \oplus [(A \oplus B).Cin]$
	População: 1000 indivíduos 500 iterações t_c=50% t_m=3%	População: 3000 indivíduos 100 iterações t_c=95% t_m=5%

Por último analisa-se a maneira em que foram implementados os AG, em relação aos operadores e parâmetros genéticos e a função avaliação

Os três operadores genéticos foram implementados e a população inicial foi gerada aleatóriamente em ambas as referências, como ilustra a Tabela 13.

Tabela 13 – Comparação entre os operadores genéticos

	Reis, 2007 [19]	**Sobrinho, 2007 [8]**	**Mantovani, 2004 [41]**
SELEÇÃO	Método do Torneio	Método do Torneio	Método do Torneio
CRUZAMENTO	De um ponto em 2 indivíduos aleatórios (permitido apenas entre células para manter a integridade do cromossoma)	De um simples ponto em um par de indivíduos aleatórios	Uniforme entre indivíduos aleatórios.
MUTAÇÃO	Altera as características totais do tipo de porta e entradas.	Indutiva (utilizada na representação por números inteiros)	Realizada a troca de *bits* de um gene de um indivíduo aleatório

O método da seleção por Torneio, para ambos, foi empregado de forma que a cada 3 indivíduos, aquele com melhor *fitness* fosse o vencedor.

Em ambas foi implementado o elitismo, mantendo as melhores soluções para a próxima geração.

A função de avaliação foi outro ponto em comum encontrado, pois nas três referências o objetivo era percorrer a tabela verdade avaliando sua funcionalidade. Em [8] e [19], além da funcionalidade, uma outra função foi aplicada para tentar gerar circuitos mais simples, ou seja, com menor número de portas lógicas.

Em relação aos parâmetros genéticos, observou-se que aqueles que utilizam populações iniciais relativamente menores, elevou-se a taxa de mutação, para que a população não sature por falta de diversidade. E os com maiores populações iniciais, tendem a diminuir

essa taxa para não tornar a população aleatória a ponto de aumentar o tempo de convergência do algoritmo.

5. CONCLUSÕES

Neste trabalho foram apresentados os algoritmos baseados nos princípios da teoria da evolução natural das espécies de Charles Darwin, a chamada Computação Evolutiva, observando-se a sua grande aplicabilidade em problemas de naturezas e complexidades diferentes, tornando-a uma ferramenta de pesquisa e otimização promissora.

Como exemplo, foram estudadas algumas aplicações da CE na eletrónica (Eletrónica Evolutiva) mostrando-se assim a eficiência desses algoritmos tanto na síntese como na otimização de circuitos eletrónicos.

O caso de estudo incidiu na aplicação dos AG na síntese de circuitos digitais, mais especificamente em circuitos combinatórios, realizando-se uma comparação entre três referências de trabalhos distintas. A comparação focou-se, não só em relação aos resultados obtidos por cada um dos autores, mas também na forma como o AG foi implementado, como: parâmetros e operadores genéticos utilizados, função de avaliação, implementação em *hardware* e tipo de codificação do circuito.

Com base no estudo efetuado, verificou-se que não é necessário ter conhecimento específico do circuito a ser sintetizado, sendo apenas necessária a tabela verdade de funcionamento dos circuitos a serem implementados. No entanto, observou-se a grande

importância do ajuste adequado dos parâmetros e operadores genéticos, pois a convergência do algoritmo depende deles.

Verificou-se também que, para populações com um grande número de indivíduos, é mais adequada a utilização de uma taxa de mutação baixa, para que a população não se diversifique em demasia ou se torne aleatória ao ponto de não ocorrer convergência para um resultado correto. Já para populações com poucos indivíduos, uma taxa de mutação baixa, resultaria na saturação da população.

Em relação à taxa de cruzamento é importante que seja elevada para que a população consiga reproduzir-se e portanto evoluir. Levando em consideração o circuito a ser sintetizado, percebeu-se que a complexidade do problema aumenta de acordo com o aumento do número de entradas do circuito, pois a tabela verdade aumenta exponencialmente, e o processo de avaliação (fitness) percorre toda a tabela verdade, aumentando a dificuldade na convergência do AG.

Com o estudo efetuado, verificou-se a eficiência do AG em relação aos métodos tradicionais de simplificação de circuitos.

Espera-se que o trabalho desenvolvido e apresentado nesta Tese seja de interesse para a área académica, uma vez que faz a interseção de trabalhos de vários autores, comparando-os de modo a ser possível retirar algumas conclusões gerais.

Referências Documentais

[1] LINDEN, Ricardo. *Algoritmos genéticos: uma importante ferramenta da inteligência computacional*. 2 ed. Rio de Janeiro: Editora BRASPORT Livros e Multimídia Ltda. 2008.

[2] PILA, Adriano – *Computação Evolutiva para a construção de conhecimento com propriedades específicas*. Tese apresentada para obtenção do título de Doutor em Ciências de Computação e Matemática Computacional pela Universidade de São Paulo - USP, em Maio de 2007.

[3] MENDES, Iba. *A origem das espécies - Charles Darwin*. São Paulo: Poeteiro Editor Digital. 2014.

[4] CORREIA, Davi. *Algoritmos Genéticos e Elementos Finitos na Síntese de Dispositivos Fotônicos*. Dissertação apresentada para a obtenção do título de Mestre em Engenharia Elétrica e Computação à Faculdade de Engenharia Elétrica e de Computação da Universidade de Campinas - UNICAMP, em Março de 2002.

[5] TORRES, Lúcia B. *Caminhos de Darwin pela Natureza Tropical Brasileira*. Em: <http://www.portaldosfarmacos.ccs.ufrj.br/atualidades_charles_darwin.html>. Acesso em: Abril de 2015.

[6] LOPES, Anabela Maria Azevedo Oliveira. *Algoritmos Genéticos: Aplicação Na Síntese De Alguns Algoritmos De Controlo*. Tese/dissertação apresentada para o título de mestre em Engenharia Electrotécnica e de Computadores na Área de Especialização de Automação e Sistemas ao Instituto Superior de Engenharia do Porto - ISEP, em Julho 2009.

[7] CORREIA, Marisol. *Algoritmos Genéticos.* Em: <http://www.dosalgarves.com/revistas/N12/5rev12.pdf>. Acesso em: Abril de 2015.

[8] SOBRINHO, Edilton Furquim Goulart. *Uma Ferramenta Alternativa para Síntese de Circuitos Lógicos Usando a Técnica de Circuito Evolutivo, p.12.* Dissertação

apresentada para obtenção do título de Mestre em Engenharia Elétrica, à Faculdade de Engenharia da Universidade de Estadual de São Paulo - UNESP – Campus de Ilha Solteira, em Junho de 2007.

[9] SILVA, Michelli M. da. *Otimização de Estruturas Reticuladas Incluindo Não-Linearidade Geométrica.* Dissertação apresentada para obtenção do título de Mestre ao programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, em 2011.

[10] OLIVEIRA, Tiago C. *Algoritmo Genético Implementado em FPGA para Evolução de Hardware.* Monografia apresentada como requisito parcial à conclusão do curso de Engenharia da Computação do Centro Universitário Positivo, Dezembro de 2007.

[11] SOARES, Gustavo L. *Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações.* Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica à Universidade Federal de Minas Gerais, em junho 1997.

[12] RODRIGUES, Wesley O. P., REIS NETO, José F. dos, SOUZA, Celso C. de. *Algoritmos Genéticos como Ferramenta de Suporte à Decisão no Planeamento de Produção de um Lacticínio*. Trabalho publicado na Revista Capital Científico – Eletrónica (RCCe) – ISSN 2177-4153 – Volume 10 n. 2 – Julho/Dezembro 2012.

[13] BASTOS, Erich A. *Otimização de Seções Retangulares de Concreto Armado Submetidas à Flexo-Compressão Oblíqua Utilizando Algoritmos Genéticos*. Dissertação apresentada para a obtenção do título de Mestre em Ciências em Engenharia Civil à Universidade Federal do Rio de Janeiro, Outubro de 2004.

[14] IKEDA, Patrícia A. *Introdução aos Algoritmos Genéticos*. Trabalho apresentado para a cadeira de Introdução ao Escalonamento e Aplicações ao Instituto de Matemática e Estatística da Universidade de São Paulo, em 2009. Acesso em: http://www.ime.usp.br/~gold/cursos/2009/mac5758/PatriciaGenetico.pdf

[15] CATARINA, Adair S. BACH, Sirlei L. *Estudo do efeito dos parâmetros genéticos sobre a solução otimizada e sobre o tempo de convergência em algoritmos genéticos com codificações binária e real.* Artigo publicado na revista Acta Scientiarum. Technology pela Editora da Universidade Estadual de Maringá, v. 25, no. 2, p. 147-152, em 2003.

[16] CASTRO, Rodrigo E. *Otimização de Estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto*. Tese apresentada para a obtenção do título de Doutor em Ciências em Engenharia Civil à Universidade Federal do Rio de Janeiro, em Maio de 2001.

[17] CASTOLDI, Marcelo F. Algoritmo Híbrido para Projeto de Controladores de Amortecimento de Sistemas Elétricos de Potência Utilizando Algoritmos Genéticos e Gradiente Descendente. Tese apresentada para a obtenção do título de Doutor em Ciências, Programa de Engenharia Elétrica à Escola de Engenharia de São Carlos da Universidade de São Paulo, em 2011.

[18] FRANZEN, Evandro. *Estudo e Implementação da Programação Genética para Síntese de Fala.* Dissertação apresentada para a obtenção do título de Mestre em Ciências da Computação à Universidade Federal do Rio Grande do Sul, em Novembro de 2002.

[19] REIS, Cecília M. *Síntese de Sistemas Digitais por Computação Evolutiva.* Tese apresentada para a obtenção do título de Doutor em Ciências da Engenharia à Universidade de Trás-os-Montes e Alto Douro, em 2007.

[20] CARACIOLO, Marcel P. *Introdução à Inteligência de Enxame - Otimização por Enxame de Partículas (PSO).* Em: <http://aimotion.blogspot.pt/2009/04/introducao-inteligencia-de-enxame.html>. Acesso em: Abril de 2015.

[21] SANTOS, Daniela S. *Bee clustering: um Algoritmo para Agrupamento de Dados Inspirados em Inteligência de Enxames.* Dissertação apresentada para a obtenção do título de Mestre em Ciência da Computação à Universidade Federal do Rio Grande do Sul, em Abril de 2009.

[22] NODA, Filipe M. *Controlador Fuzzy Otimizado Com Algoritmo De Colônia De Abelhas Artificiais.* Projeto Final apresentado para obtenção do título de Engenheiro Eletricista à Universidade Federal do Paraná, em 2010.

[23] PENNACHIN, Cassio. *Otimização inspirada em formigas.* Em: <http://blog.vettalabs.com/index.html%3Fp=134>. Acesso em: Abril de 2015.

[24] MEDEIROS, Guilherme Fleith de; KRIPKA, Moacir. *Algumas Aplicações de Métodos Heurísticos na Otimização de Estruturas.* Revista CIATEC – UPF, vol.4 (1), p.p.19-32, em 2012.

[25] CARVALHO, Érica da C R. *Solução de problemas de otimização com restrições usando estratégias de penalização adaptativa e um algoritmo do tipo PSO*. Dissertação apresentada para à obtenção do título de Mestre ao programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, em 2011.

[26] LEITÃO, H. A. S., LOPES, W. T. A., BERNARDINO JUNIOR, F. M. *PSO Algorithm Applied to Codebook Design for Channel-Optimized Vector Quantization*. Publicado em IEEE Latin America Transactions, VOL. 13, NO. 4, em Abril de 2015.

[27] SZENDRODI, Claudio E. C. *Síntese de Filtros Analógicos Utilizando Técnicas Evolutivas e Sócio-Cognitivas*. Dissertação apresentada para a obtenção do título de Mestre em Ciências em Engenharia Elétrica à Universidade Federal do Rio de Janeiro, em Abril de 2005.

[28] ZEBULUM, Ricardo S. PACHECO, Marco A. C., VELLASCO, Marley M. B. R. Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. Editora CRC Press, do grupo Taylor & Francis, em 2002.

[29] Dicionário do Aurélio. Em: <http://www.dicionariodoaurelio.com/robotica>. Acesso em: Junho de 2015

[30] MEDEIROS, Talles H. de. GÓES, Luís F. W. CARVALHO, Milene B. MENEZES, Ilg. *Computação Bioinspirada aplicada à Robótica.* Capítulo 4

[31] PIRES, J. Norberto. *Robótica: das Máquinas Gregas à Moderna Robótica Industrial*. Departamento de Engenharia Mecânica da Universidade de Coimbra, publicado ao Jornal Público, caderno de computadores, em Julho de 2002.

[32] CAZANGI, Renato R. *Síntese de Controladores Autônomos em Robótica Móvel por meio de Computação Bio-inspirada.* Tese submetida para a obtenção do título de Doutor em Engenharia Elétrica na área de Engenharia de Computação à Universidade Estadual de Campinas, em Dezembro de 2008.

[33] PEREZ, Anderson L.F. *Extensão da Programação Genética Distribuída para Suportar a Evolução do Sistema de Controle em uma População de Robôs Móveis*. Tese submetida para a obtenção do título de Doutor em Engenharia Elétrica à Universidade Federal de Santa Catarina, em Fevereiro de 2010.

[34] SILVA, Bruno de A. *Metodologia para Criação de um Sistema de Hardware Evolutivo em FPGA Utilizando o Processador Nios II*. Monografia submetida para a obtenção do título de Bacharel em Ciências da Computação à Universidade Federal de Lavras, em 2008.

[35] OLIVEIRA, Caio A. de, AGUIAR, Jéssica A. de, FONTANINI, Mateus G. S. Dispositivos Lógicos Programáveis. Arquivo de pesquisa as Universidade Estadual Paulista.

[36] TEIXEIRA, Marco A. Técnicas de reconfigurabilidade dos FPGAs da família APEX 20K – Altera. Dissertação submetida para a obtenção do título de Mestre em Ciências Matemáticas e de Computação à Universidade de São Paulo, em Julho de 2002.

[37] PÓVOA, Rogério C. B. L. Síntese Evolucionária de Circuitos Digitais Empregando FPGA's.

[38] ABBOUD, Nadine. GRÖTSCHEL, Martin. KOCH, Thorsten. *Mathematical methods for physical layout of printed circuit boards: an overview.* Artigo apoiado pela DFG Research Center Matheon em Berlim, publicado online em 23 de Maio de 2007.

[39] DORO, Marcos M. *Sistemática para Implantação da Garantia da Qualidade em Empresas Montadoras de Placas de Circuito Impresso.* Dissertação submetida para a obtenção do título de Mestre em Metrologia à Universidade Federal de Santa Catarina, em Julho de 2004.

[40] LOPES, Heitor S. *Algoritmos Genéticos em Projetos de Engenharia: Aplicações e Perspectivas Futuras*. Artigo apresentado no 4º Simpósio Brasileiro de Automação Inteligente.

[41] MANTOVANI, Suely C. A., OLIVEIRA, José R. de. *Síntese de Circuitos Digitais por Evolução de Circuitos.* Trabalho submetido ao XXXVI Simpósio Brasileiro de Pesquisa Operacional.

[42] TOCCI, Ronald J. *Sistemas Digitais – Princípios e Aplicações*. 10ª edição, editora Pearson.

[43] IDOETA, Ivan V., CAPUANO, Francisco G. *Elementos de Eletrónica Digital*. 36ª edição, editora Érica.

[44] ARAÚJO, Sérgio G. de. *Síntese de Sistemas Digitais utilizando Técnicas Evolutivas*. Tese submetida para obtenção do título de Doutor em Ciências em Engenharia Elétrica, à Universidade Federal do Rio de Janeiro, em Julho de 2004.

[45] S. M. Ashik Eftakhar, SK. Mahbub Habib, M. M. A. Hashem. *Evolutionary Design of Digital Circuits Using Genetic Programming.* Artigo publicado ao Departamento de Ciências da Computação e Engenharia na Khulna University of Engineering and Technology.

[46] PEIXOTO, LUIZ H. R. *Síntese de Circuitos Eletrónicos Analógicos/Digitais por Computação Evolutiva em Plataforma Paralela*. Monografia apresentada para a obtenção do título de Bacharel em Ciência da Computação à Universidade Federal de Lavras, em 2011.

[47] LÓPEZ-PASTOR, Eduardo T. *Algoritmo de RWA com Considerações de Sobrevivência Baseado em Heurística – Algoritmo Genético para Redes IP/WDM.* Tese apresentada para a obtenção do título de Doutor em Engenharia Elétrica à Universidade de Brasília, em 2007.

[48] FIGUEIREDO, Rodrigo M., SANTOS, José V. C. dos. *Um Comparativo entre Métodos Computacionais para Planeamento de Redes de Telecomunicações.* Programa de Pós-Graduação em Computação Aplicada, UNISINOS – São Leopoldo (RS), artigo publicado na Revista Brasileira de Computação Aplicada (ISSN 2176-6649), Passo Fundo, v. 5, n. 1, p. 14-25, em abril de 2013.

[49] CAMPOS, Tatiane J. de. *Hardware Evolutivo Aplicado à Geração Automática de Controladores para Servo-Mecanismos*. Tese apresentada para obtenção do título de Doutor em Engenharia Elétrica, à Faculdade de Engenharia Elétrica e de Computação da Universidade de Estadual de Campinas, em Julho de 2007

[50] SACCHI, Filipe. *Sintonia de um sistema PID via Algoritmos Genéticos aplicado ao controle de um manipulador robótico em forma de paralelogramo*. Trabalho

apresentado pelo departamento de Engenharia Elétrica da Pontifícia Universidade Católica do Rio de Janeiro.

[51] ANDRÉ, Leanderson, PARPINELLI, Rafael S. *Tutorial Sobre o Uso de Técnicas para Controle Online de Parâmetros em Algoritmos de Inteligência de Enxame e Computação Evolutiva*. Trabalho feito no Programa de Pós-Graduação em Computação Aplicada na Universidade do Estado de Santa Catarina, publicado na Revista de Informática Teórica e Aplicada, volume 21, número 2, em 2014.

[52] SILVA, Luiz M. C.da. *Análise de Circuitos Digitais – Multiplexadores*. Aula ministrada na Universidade Tecnológica Federal do Paraná campus Cornélio Procópio. Em: <http://www.cp.utfpr.edu.br/chiesse/Sistemas_Digitais/Multiplexadores.pdf>. Acesso em: Julho de 2015.

[53] BROWN, Stephen; ROSE, Jonathan. *Architecture of FPGAs and CPLDs: A Tutorial*. Department of Electrical and Computer Engineering University of Toronto.

[54] OLIVEIRA, Caio A. de, AGUIAR, Jéssica A. de, FONTANINI, Mateus G. S. *Dispositivos Lógicos Programáveis*. Em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/logica-programavel.pdf>. Acesso em: Julho de 2015.

Anexo A. Circuitos Digitais

Neste anexo pretende-se fazer uma complementação da subseção 3.5.1, com o aprofundamento de alguns tópicos referentes a eletrónica digital que são de grande importância para a realização do caso de estudo, tornando-o mais compreensível possível.

A.1. Álgebra de Boole e portas lógicas básicas

A Figura 44 mostra um quadro resumo das especificações da álgebra booleana, que servem como ferramentas para a simplificação de circuitos digitais.

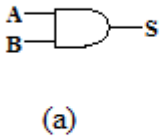
POSTULADOS		
Complementação	**Adição**	**Multiplicação**
$A = 0 \rightarrow \overline{A} = 1$ $A = 1 \rightarrow \overline{A} = 0$	$0 + 0 = 0$ $0 + 1 = 1$ $1 + 0 = 1$ $1 + 1 = 1$	$0 . 0 = 0$ $0 . 1 = 0$ $1 . 0 = 0$ $1 . 1 = 1$
IDENTIDADES		
Complementação	**Adição**	**Multiplicação**
$\overline{\overline{A}} = A$	$A + 0 = A$ $A + 1 = 1$ $A + A = A$ $A + \overline{A} = 1$	$A . 0 = 0$ $A . 1 = A$ $A . A = A$ $A . \overline{A} = 0$
PROPRIEDADES		
Comutativa:	$A + B = B + A$ $A . B = B . A$	
Associativa:	$A + (B + C) = (A + B) + C = A + B + C$ $A . (B . C) = (A . B) . C = A . B . C$	
Distributiva:	$A . (B + C) = A . B + A . C$	
TEOREMAS DE MORGAN		
$\overline{(A . B)} = \overline{A} + \overline{B}$ $\overline{(A + B)} = \overline{A} . \overline{B}$		
IDENTIDADES AUXILIARES		
$A + A . B = A$ $A + \overline{A} . B = A + B$ $(A + B) . (A + C) = A + B . C$		

Figura 44 - Quadro resumo da Álgebra de Boole
[43]

Abaixo encontram-se as sete portas lógicas básicas (com duas entradas) e suas devidas representações.

- *AND*: também conhecida como porta E, equivalente a uma multiplicação como mostra a equação A.1. Resulta em um valor lógico verdadeiro se, e somente se, todas as entradas tiverem um valor verdadeiro, como mostra a Figura 45.

$$S = A \,.\, B \tag{A.1}$$



(a)

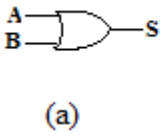
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

(b)

Figura 45 – (a) Simbologia da porta *AND* (b) Tabela verdade *AND*

- *OR*: também conhecida como porta OU, equivalente a uma soma como mostra a equação A.2. Resulta em um valor lógico verdadeiro se uma das entradas ou todas tiverem um valor verdadeiro, como mostra a Figura 46.

$$S = A + B \tag{A.2}$$



(a)

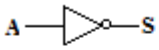
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

(b)

Figura 46 - (a) Simbologia da porta *OR* (b) Tabela verdade OR

- NOT: também chamada de porta inversora, sua saída é o estado lógico complementar da sua entrada, como mostra a equação A.3 e a Figura 47. Possui apenas uma entrada e a sua saída negada.

$$S = \overline{A} \tag{A.3}$$



(a)

A	S
0	1
1	0

(b)

Figura 47 - (a) Simbologia da porta *NOT* (b) Tabela verdade *NOT*

- NAND: é uma porta AND seguida de uma porta NOT, ou seja, a saída é falsa somente se todas as entradas forem verdadeiras, como ilustra a equação A.4 e a Figura 48.

$$S = \overline{A.\,B} \tag{A.4}$$

(a)

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

(b)

Figura 48 - (a) Simbologia da porta *NAND* (b) Tabela verdade *NAND*

- NOR: é uma porta OR seguida de uma porta NOT, ou seja, a saída é falsa se uma das entradas ou todas forem verdadeiras, como ilustra a equação A.5 e a Figura 49.

$$S = \overline{A + B} \tag{A.5}$$

(a)

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

(b)

Figura 49 - (a) Simbologia da porta *NOR* (b) Tabela verdade *NOR*

- XOR: conhecida também como OU exclusivo, a saída é verdadeira se as entradas são diferentes, e a saída é falsa se as entradas forem iguais (ambas as entradas são falsas ou verdadeiras), como mostra a equação A.6 e a Figura 50.

$$S = \overline{A}.B + A.\overline{B} = A \oplus B \tag{A.6}$$

(a)

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

(b)

Figura 50 - (a) Simbologia da porta X*OR* (b) Tabela verdade *XOR*

- XNOR: é a combinação de uma porta XOR seguida de uma NOT, sua saída é verdadeira se as entradas são iguais, e falsa caso as entradas sejam diferentes, como ilustra a equação A.7 e a Figura 51.

$$S = \overline{A}.\overline{B} + A.B = \overline{A \oplus B} \qquad \text{(A.7)}$$

(a)

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

(b)

Figura 51 - (a) Simbologia da porta *XNOR* (b) Tabela verdade *XNOR*

A.2. Circuitos digitais combinatórios

Como visto anteriormente na subseção 3.5.1, o nível lógico da saída dos circuitos combinatórios depende exclusivamente da combinação dos níveis lógicos presentes nas entradas, não possuindo a característica de memória. Entre esses circuitos destancam-se os codificadores, decodificadores, multiplexadores, demultiplexadores, testes de paridade e os circuitos aritméticos (meio somador, somador completo, meio subtrator, subtrator completo, multiplicador), como mostra abaixo.

- CODIFICADORES/ DECODIFICADORES: são circuitos capazes de transformar um código conhecido para um desconhecido, onde um faz o processo contrário do outro. Transforma a informação, ou seja, transformar o valor de entrada no valor de saída. Podendo ativar uma combinação de saídas mediante a ativação de uma única entrada dentre o seu conjunto de entradas, ativar uma única saída dentre o seu conjunto de saídas mediante a ativação uma combinação de suas entradas ou ativar uma combinação de suas saídas mediante a ativação de uma combinação de suas entradas.

- MULTIPLEXADORES: também conhecido por MUX, tem a finalidade de selecionar, através de variáveis de seleção, uma de suas entradas, conectando-a eletronicamente à uma única saída. A Figura 52 representa um multiplexador de N canais, onde as variáveis de seleção são representados por A_1, A_2,..., A_n, as entradas por E_9, E_8,..., E_n, e a saída por S.

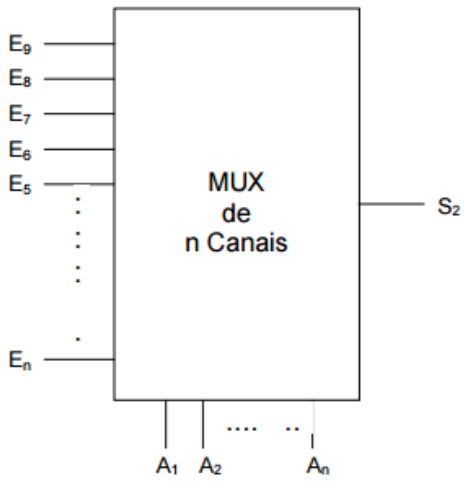


Figura 52 - Representação de um multiplexador de N canais
[52]

Nele, o número de entradas está relacionado com o número de variáveis de seleção, de acordo com a equação A.8, onde *n* é o número de entradas e *m* o número de variáveis de seleção [52].

$$n = 2^m \tag{A.8}$$

- DEMULTIPLEXADORES: também conhecido por DEMUX, tem a finalidade de selecionar, através das variáveis de seleção, qual de suas saídas deve receber a informação presente em sua única entrada. Realiza a operação inversa do multiplexador. A Figura 53 representa um demultiplexador de N canais, onde as variáveis de seleção são representados por A_1, A_2,..., A_{m-1}, a entrada por E e as saídas por S_0, S_1,..., S_{n-1}.

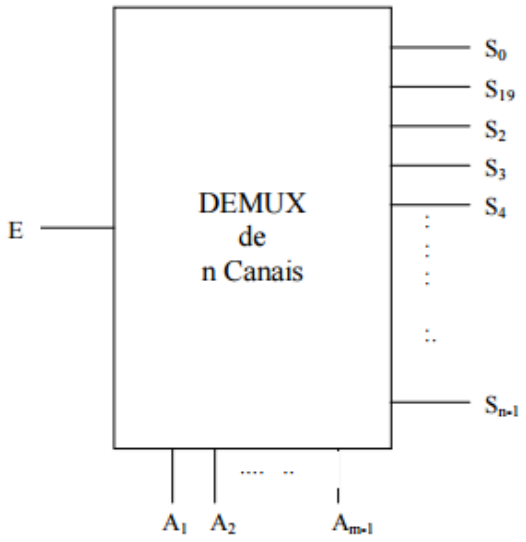


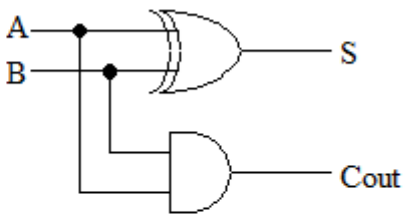
Figura 53 - Representação de um demultiplexador de N canais
[52]

O número de entradas está relacionado com o número de variáveis de seleção da mesma maneira que no multiplexador [52].

- MEIO SOMADOR: um circuito aritmético que possibilita a soma de números binários de 1 *bit*, com entradas A e B, e como saída, a soma dos algarismos (S) e o respectivo transporte de saída (*carry out* – Cout). A Figura 54 ilustra a tabela verdade, o circuito equivalente e as expressões características do meio somador.

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a)



(b)

$S = A \oplus B$

$Cout = A.B$

(c)

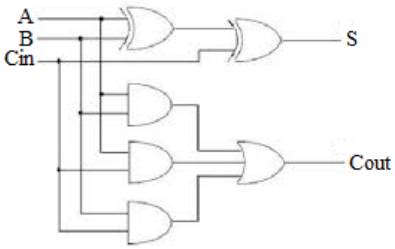
Figura 54 – (a) Tabela verdade (b) circuito equivalente (c) expressões características, do meio somador

É insuficente na soma de número de mais *bits*, pois não possibilita a introdução do transporte de entrada (*carry in* - Cin) proveniente da coluna anterior [43].

- SOMADOR COMPLETO: circuito aritmético que efetua a soma de dois números binários de mais *bits*, somando coluna a coluna, levando em consideração o Cin (equivalente ao Cout da coluna anterior). A Figura 55 ilustra a tabela verdade, o circuito equivalente e as expressões características do somador completo [43].

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a)



(b)

$S = A \oplus B \oplus Cin$

$Cout = B.Cin + A.Cin + A.B$

(c)

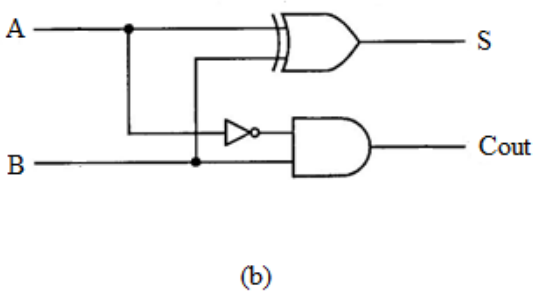
Figura 55 - (a) Tabela verdade (b) circuito equivalente (c) expressões características, do somador completo

- MEIO SUBTRATOR: um circuito aritmético que realiza a subtração de números binários de 1 *bit*, com entradas A e B, e como saída, a subtração dos algarismos (S) e o

respectivo transporte de saída (*carry out* – Cout). A Figura 56 ilustra a tabela verdade, o circuito equivalente e as expressões características do meio subtrator [43].

A	B	S	Cout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

(a)



(b)

$S = A \oplus B$

$Cout = \overline{A}.B$

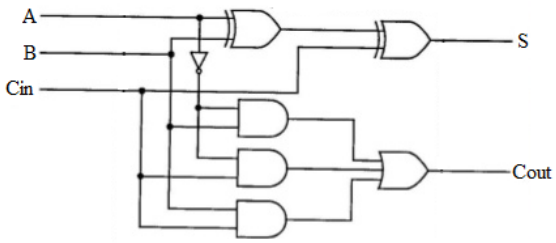
(c)

Figura 56 - (a) Tabela verdade (b) circuito equivalente (c) expressões características, do meio subtrator

- SUBTRATOR COMPLETO: circuito aritmético que realiza a subtração de dois números binários de mais *bits*, subtraindo coluna a coluna, levando em consideração o Cin (equivalente ao Cout da coluna anterior). A Figura 57 ilustra a tabela verdade, o circuito equivalente e as expressões características do subtrator completo [43].

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(a)



(b)

$S = A \oplus B \oplus Cin$

$Cout = \overline{A}.B + \overline{A}.Cin + B.Cin$

(c)

Figura 57 - (a) Tabela verdade (b) circuito equivalente (c) expressões características, do subtrator completo

- MULTIPLICADOR: é responsável pela multiplicação de números binários de N *bits*, a **Figura 58** apresenta a multiplicação de dois números com 2 *bits*. É basicamente igual a multiplicação com números decimais.

		A_1	A_0
x		B_1	B_0
		B_0A_1	B_0A_0
	B_1A_1	B_1A_0	+
P_3	P_2	P_1	P_0

Figura 58 – Multiplicação para números de 2 *bits*

Anexo B. Dispositivos de Lógica Programável

Os dispositivos de lógica programável (*Programmable Logic Devices* – PLD) referem a qualquer tipo de circuito integrado utilizado para a implementação de *hardware* digital, onde o chip pode ser configurado pelo usuário final para realizar projetos diferentes. A principal característica é a capacidade de programação das funções lógicas pelo usuário, facilitando, assim, as prováveis mudanças de projeto. Em comparação com outras tecnologias de circuitos integrados digitais, os dispositivos de lógica programável apresentam um ciclo de projeto menor e custos reduzidos. Os PLDs podem ser classificados em função de portas lógicas que comportam, como: **SPLDs** (*Simple Programmable Logic Devices*), dispositivos simples e de baixa capacidade, geralmente um PLA (*Programmable Logic Arrays*) ou PAL (*Programmable Array Logic*); **HCPLDs** (*High Complex Programmable Logic Devices*), dispositivos de alta capacidade que podem ser vistos como dispositivos que integram na sua estrutura centenas de macrocélulas programáveis, que são interligadas por conexões também programáveis, englobam os dispositivos CPLDs (*Complex Programmable Logic Devices*) e FPGAs (*Field Programmable Gate Arrays*) [53].

- PLA: é relativamente um pequeno PLD que contém dois níveis de portas lógicas, um plano de portas AND seguido por um planho de portas OR, ambos programáveis, como mostra a Figura 59. São adequados para as implementações de funções lógicas na forma de produtos de soma, e eles se apresentam muito versáteis, pois os termos AND e OR podem possuir muitas entradas.

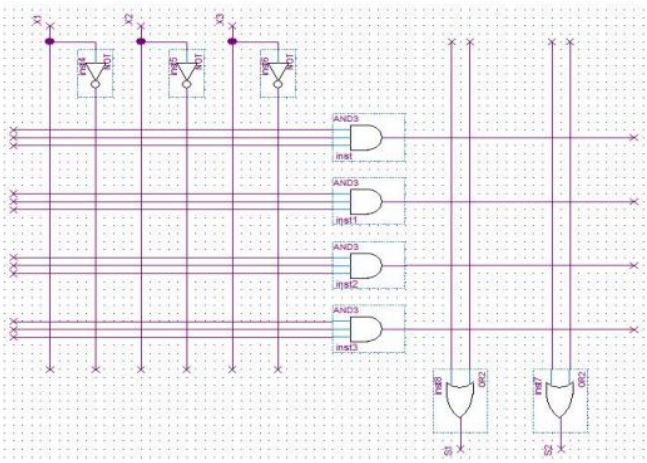


Figura 59 - Estrutura simplificado de um PLA
[54]

Quando os PLAs foram introduzidos no início de 1970, pela Philips, suas principais desvantagens eram a fabricação cara e baixa velocidade de desempenho. Ambos os desvantagens eram devido aos dois níveis de configurabilidade lógica, porque os planos lógicos programáveis eram difíceis de fabricar. Para superar essas debilidades, foram desenvolvidos os PAL.

- PAL: também é um PLD relativamente pequeno que tem um único plano AND programável, seguido por um plano OR fixo, como mostra a Figura 60.

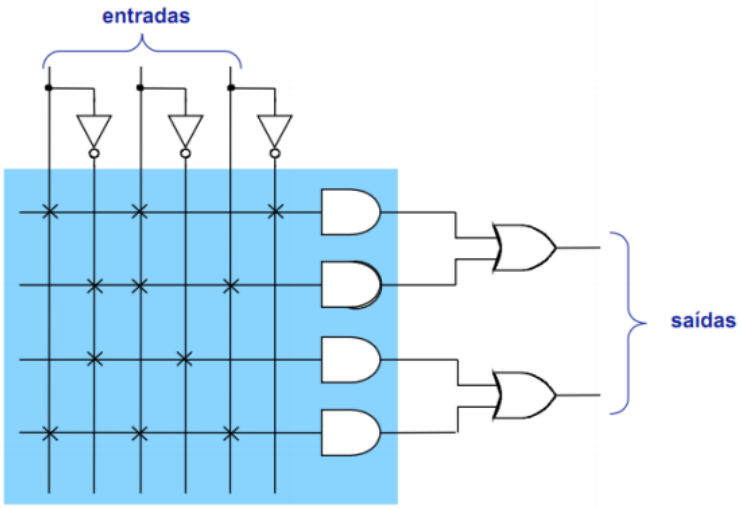


Figura 60 - Estrutura simplificado de um PAL
[54]

- CPLD: Os Dispositivos Lógicos Programáveis Complexos são dispositivos programáveis e reprogramáveis pelo usuário, com alto desempenho, baixo custo por função e alta capacidade de integração. Até mesmo para os CPLDs, apenas circuitos moderadamente grandes podem ser acomodados em um único circuito integrado. Para se implementar circuitos lógicos maiores, é conveniente utilizar-se de outro tipo de dispositivo HCPLD que possua capacidade lógica maior, os FPGAs.

- FPGA: é um HCPLD que suporta a implementação de circuitos lógicos relativamente grandes. Consiste em um grande arranjo de células lógicas ou blocos lógicos configuráveis contidos em um único circuito integrado. Cada célula contém capacidade computacional para implementar funções lógicas e realizar roteamento para comunicação entre elas. Os FPGAs não possuem planos OR ou AND, consistem em um grande arranjo de células configuráveis que podem ser utilizadas para a implementação de funções lógicas.

Conforme a complexidade dos projetos aumenta, as descrições em nível de esquemas lógicos tornam-se inviáveis, fazendo-se necessário descrever esses projetos em modos mais abstratos, como as linguagens de descrição de *hardware*, também conhecidas como HDLs (*Hardware Description Language*). Existem diversas linguagens de descrição de *hardware* disponíveis, sendo as mais comuns: ABEL (*Advanced Boolean Equation Language*), VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) e Verilog. A linguagem ABEL foi a primeira linguagem HDL a ser desenvolvida, criada para programar dispositivos SPLD, é uma linguagem mais simples que a linguagem VHDL. Já a VHDL e Verilog são capazes de programar sistemas de maior complexidade como, por exemplo, os dispositivos FPGA [54].

Anexo C. Resultados das referências estudadas

Este anexo contém os resultados encontrados nas três referências das seções 4.2, 4.3 e 4.4, que foram utilizidos para o caso de estudo da seção 4.5, observando e comparando os resultados obtidos.

A) Referente a seção 4.2, os resultados obtidos através das simulações foram [19]:

- Multiplexador 2 para 1: Como este circuito tem 3 entradas e 1 saída, resulta $f_{10} = 8$ e F ≥12. Através da Figura 61, observou-se que o melhor conjunto de portas lógicas é o *Gset2*, dado que chega à solução com a menor média do número de gerações do AG.

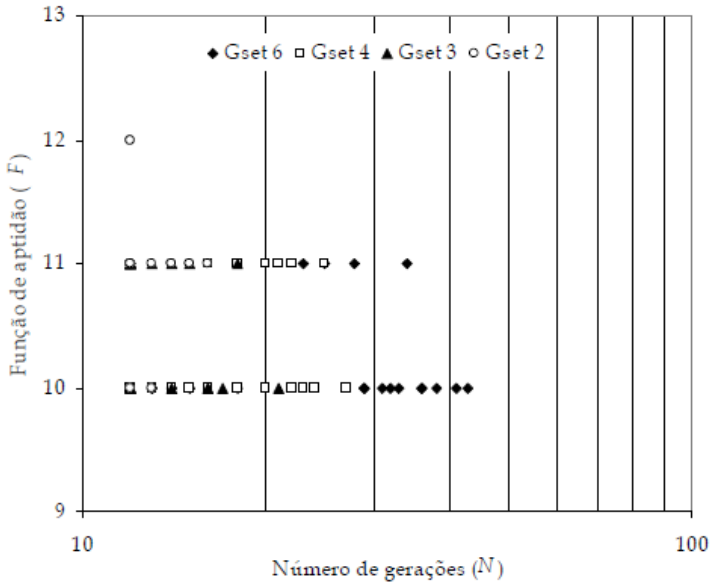


Figura 61 - Função de aptidão versus número de gerações

Os melhores circuitos resultantes apresentam uma função de aptidão final F = 12, como o circuito ilustrado na Figura 62.

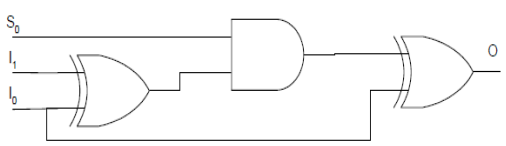


Figura 62 - Circuito Multiplexador 2 para 1 gerado pelo AG

- Somador completo: o circuito contém 3 entradas e 2 saídas, tem-se $f_{10} = 16$ e $F \geq 20$.

Através da Figura 63, observa-se que conjunto de portas lógicas que apresenta melhor desempenho é aquele que atinge a solução no menor número de gerações N e com a função de aptidão mais elevada, neste caso são os conjuntos de portas lógicas *Gset3* e *Gset2*.

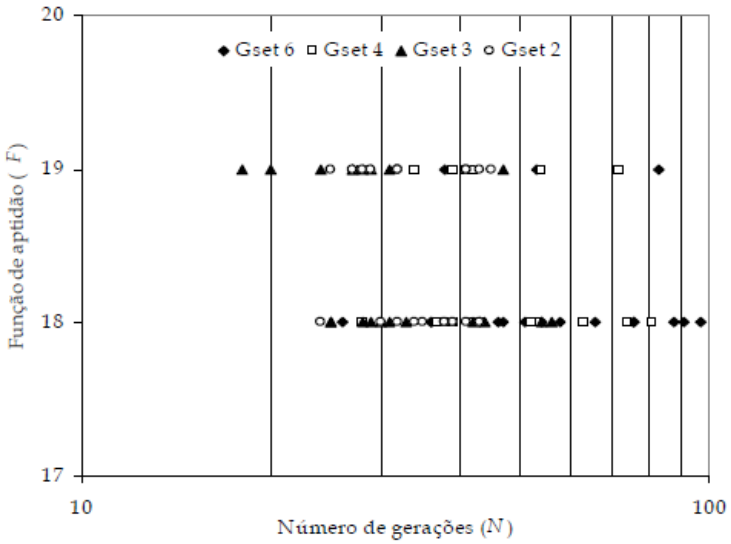


Figura 63 - Função de aptidão versus número de gerações

Os melhores circuitos obtidos têm uma função de aptidão final F=19, sendo o esquemático ilustrado na Figura 64 um exemplo de um destes circuitos gerado com o *Gset2*.

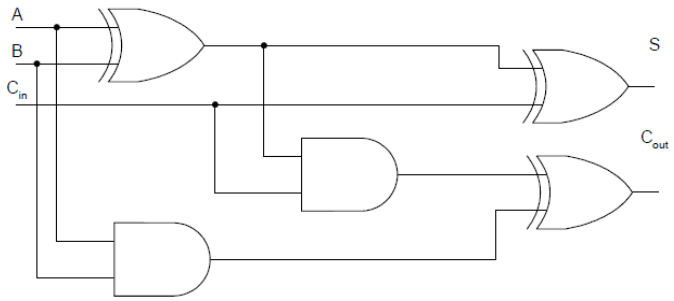


Figura 64 - Circuito somador gerado pelo AG

- Teste de paridade de 4 *bits*: como este circuito tem 4 entradas e 1 saída, resulta $f_{10}=16$ e $F \geq 24$. Através da Figura 65, observou-se que o melhor conjunto de portas lógicas é o *Gset 2*, dado que chega à solução com a menor média do número de gerações do AG. A Figura 66 apresenta o esquemático de um dos circuitos gerado com a mais alta função de aptidão (F=25).

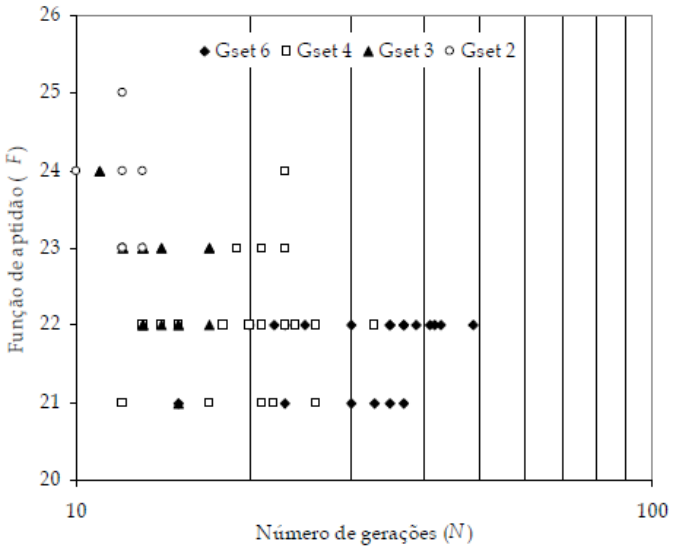


Figura 65 - Função de aptidão versus número de gerações

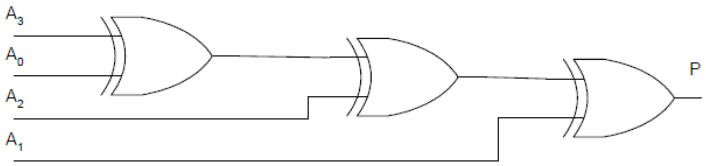


Figura 66 – Circuito teste de paridade gerado pelo AG

- Multiplicador de 2 *bits*: Como este circuito tem 4 entradas e 4 saídas, resulta f_{10}=64 e F ≥72. Através da Figura 67, mais uma vez conclui-se que o melhor conjunto de portas lógicas é o *Gset2*, dado que chega à solução com a menor média do número de gerações do AG. A Figura 68 apresenta o esquemático de um dos circuitos gerado com a mais alta função de aptidão (F=72).

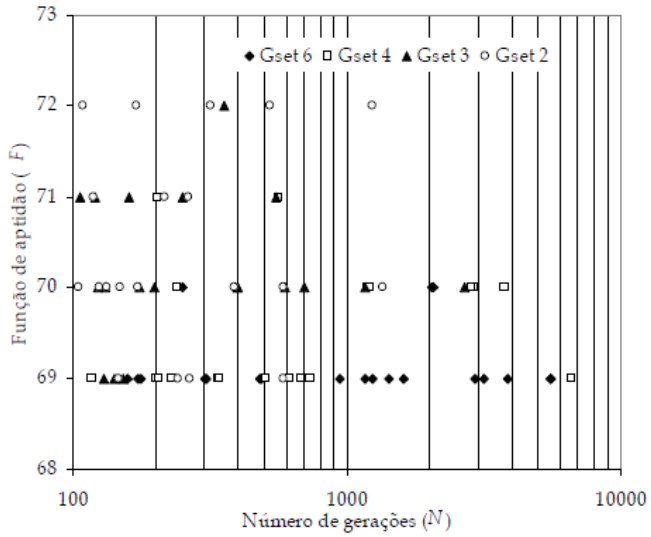


Figura 67 - Função de aptidão versus número de gerações

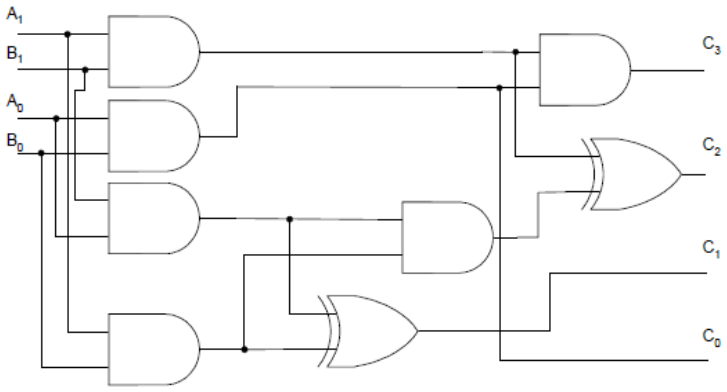


Figura 68 - Circuito multiplicador gerado pelo AG

B) Referente a seção 4.3, os resultados obtidos através das simulações foram [8], para a representação por mapeamento de fusíveis:

- Somador: com 3 entradas, a Figura 69 mostra o número de iterações em função da taxa de acertos para uma população de 250 indivíduos e 50 iterações, em (a) para t_c=80% e t_m=10%, foi encontrado um circuito evoluído mostrado na Figura 70. Em (b) para t_c=70% e t_m=1%, ocorre uma saturação da população por falta de diversidade.

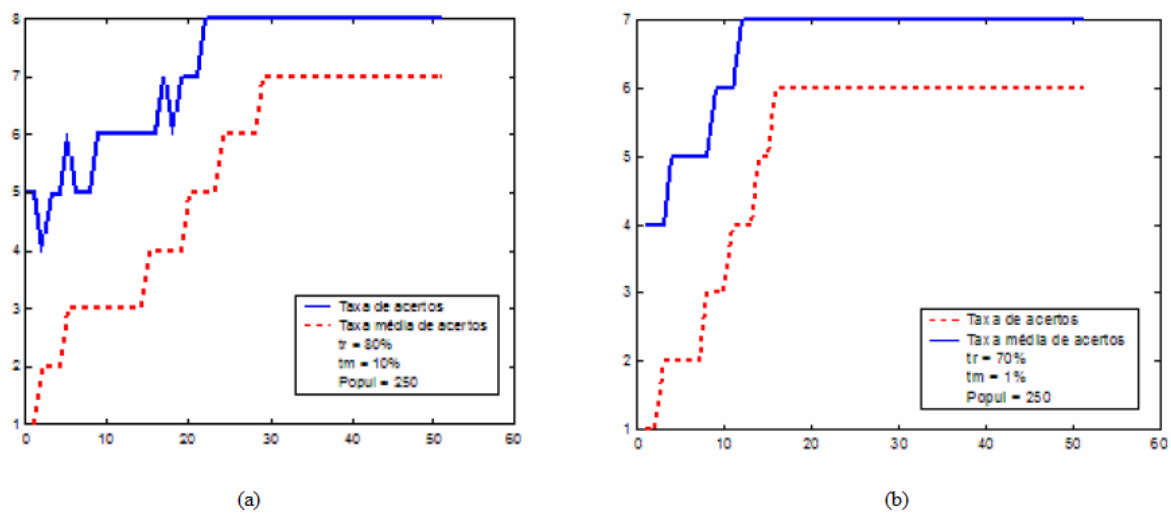


Figura 69 - Número de iterações x taxa de acertos para o somador

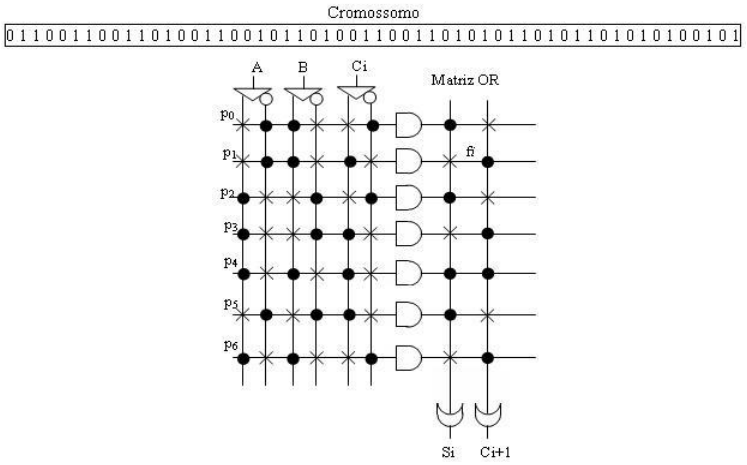


Figura 70 - Cromossoma e circuito encontrado pelo AG

- Detector de números primos: com 4 entradas, a Figura 71 mostra o número de iterações em função da taxa de acertos para uma população de 250 indivíduos e 50 iterações, em (a) para t_c=80% e t_m=10%, foi encontrado um circuito evoluído mostrado na Figura 72. Em (b) para t_c=70% e t_m=1%, obteve-se 15 pontos dos 16 necessários.

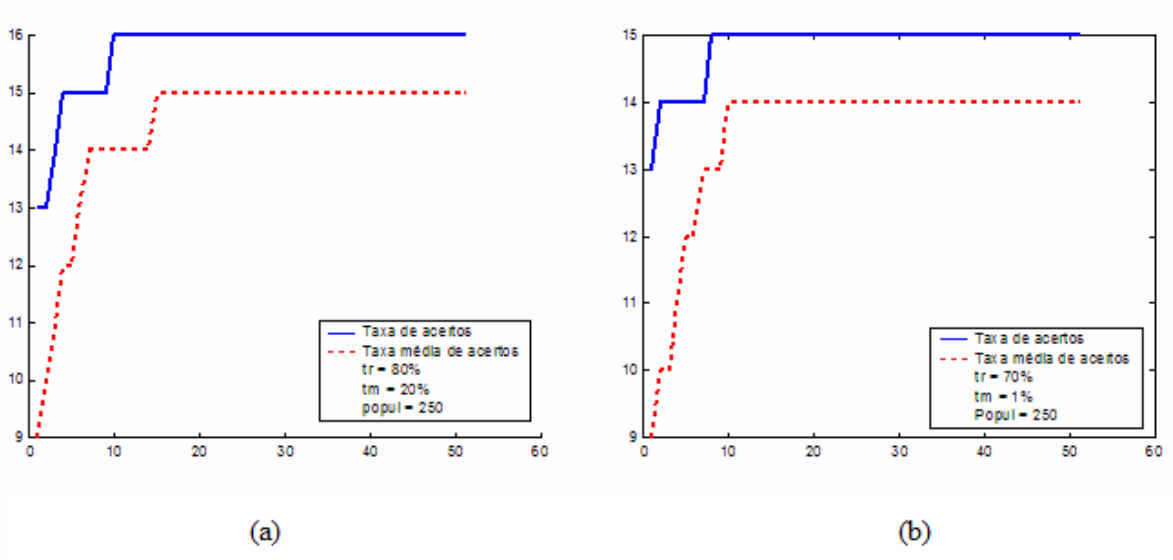


Figura 71 - Número de iterações x taxa de acertos para o detector de números primos

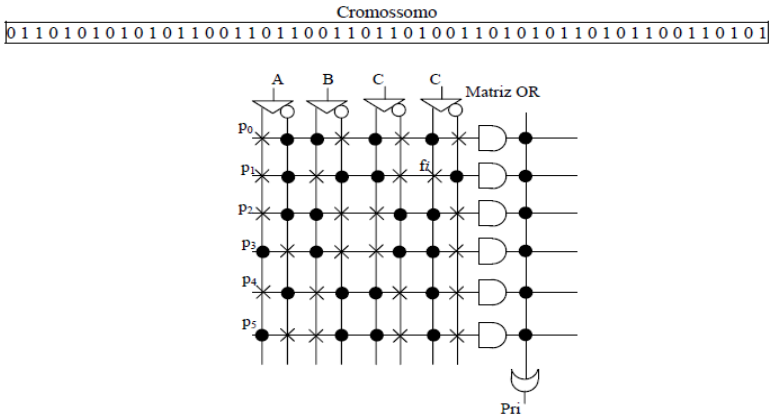


Figura 72 - Cromossoma e circuito encontrado pelo AG

Para a representação por portas lógicas com codificação binária:

- Circuito combinatório (3 entradas): a Figura 73 mostra o número de iterações em função da taxa de acertos (*fitness*) para uma população de 500 indivíduos, 200 iterações, com t_c=80% e t_m=40%, foi encontrado um circuito evoluído mostrado na Figura 74.

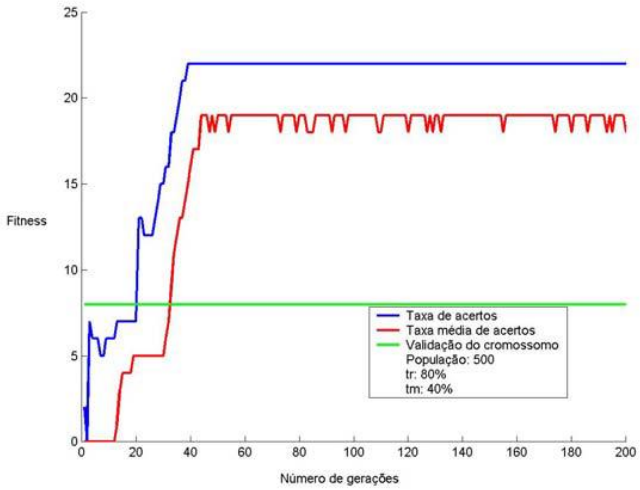


Figura 73 - Número de iterações x taxa de acertos para o circuito combinatório de 3 entradas

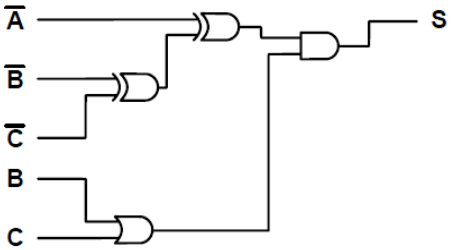


Figura 74 - Circuito resultante do AG

- Circuito combinatório (4 entradas): a Figura 75 mostra o número de iterações em função da taxa de acertos (*fitness*) para uma população de 500 indivíduos, 500 iterações, com t_c=80% e t_m=40%, foi encontrado um circuito evoluído mostrado na Figura 76.

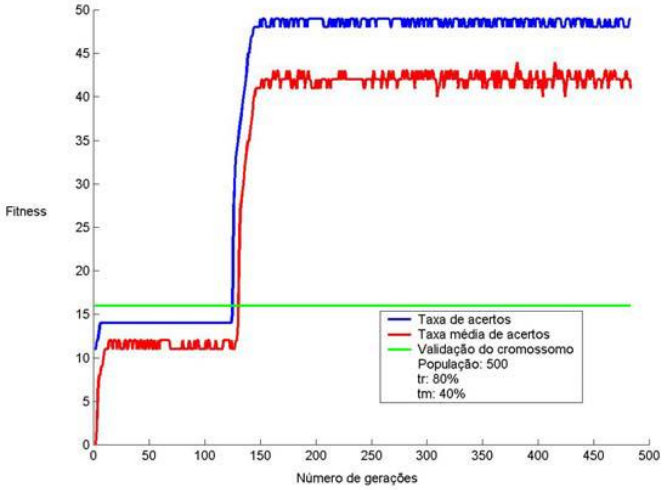


Figura 75 - Número de iterações x taxa de acertos para o circuito combinatório de 4 entradas

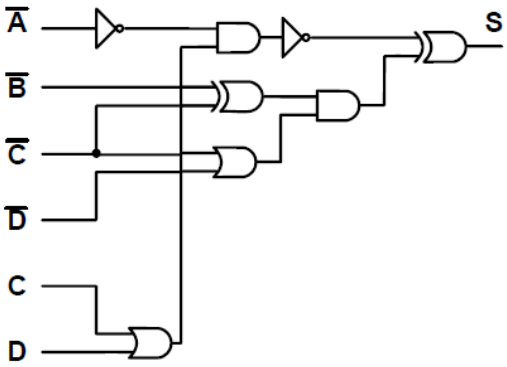


Figura 76 – Circuito resultante do AG

Para representação por portas lógicas com codificação inteira:

- Circuito combinatório (3 entradas): para uma população de 500 indivíduos, 500 iterações, com t_c=50% e t_m=5%. Na simplificação pelo mapa de Karnaugh obtém-se a equação C.1, totalizando 5 portas lógicas (2 AND's, 2 XOR's e 1 OR). Com a aplicação do AG, o circuito é sintetizado para a equação C.2, na iteração 260 aproximadamente, totalizando 4 portas lógicas (2 XOR's, 1 AND e 1 OR).

$$S = C.(A \oplus B) + B.(A \oplus C) \tag{C.1}$$

$$S = A.(B + C) \oplus (B \oplus C) \tag{C.2}$$

- Circuito combinatório (4 entradas): para uma população de 1000 indivíduos, 1000 iterações, com t_c=50% e t_m=2%. Na simplificação pelo mapa de Karnaugh obtém-se a equação C.3, totalizando 11 portas lógicas (4 AND's, 4 NOT's, 2 XOR's e 1 OR). Com a aplicação do AG, o circuito é sintetizado para a equação C.4, na iteração 990 aproximadamente, totalizando 9 portas lógicas (4 AND's, 2 XOR's, 2 NOT's e 1 OR).

$$S = [(\overline{A}.C) \oplus (\overline{D}.\overline{B})] + [(\overline{C}.D).(C \oplus \overline{B})] \tag{C.3}$$

$$S = \overline{\{A.[A.(C + D)] \oplus (B \oplus C).(\overline{\overline{C}.\overline{D}})\}} \tag{C.4}$$

- Somador de 1 *bit*: para uma população de 1000 indivíduos, 500 iterações, com t_c=50% e t_m=3%. O circuito é sintetizado para a Figura 77, na iteração 100 aproximadamente, totalizando 5 portas lógicas (4 XOR's e 1 OR).

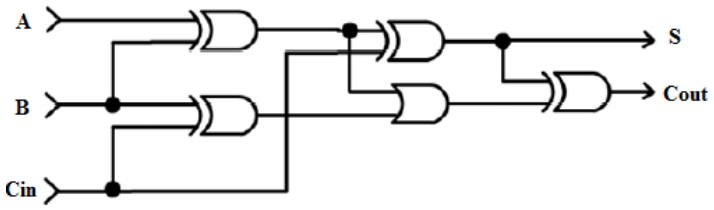


Figura 77 – Circuito Somador minimizado

C) Referente a seção 4.4, os resultados obtidos através das simulações foram [41]:

- Decodificador: com 2 entradas, a Figura 78 mostra o número de iterações em função da taxa de acertos para uma população de 200 indivíduos e 50 iterações, em (a) para t_c=70% e t_m=1%, ocorre uma saturação da população por falta de diversidade. Em (b) para t_c=65% e t_m=50%, foi encontrado um circuito evoluído mostrada na Figura 79.

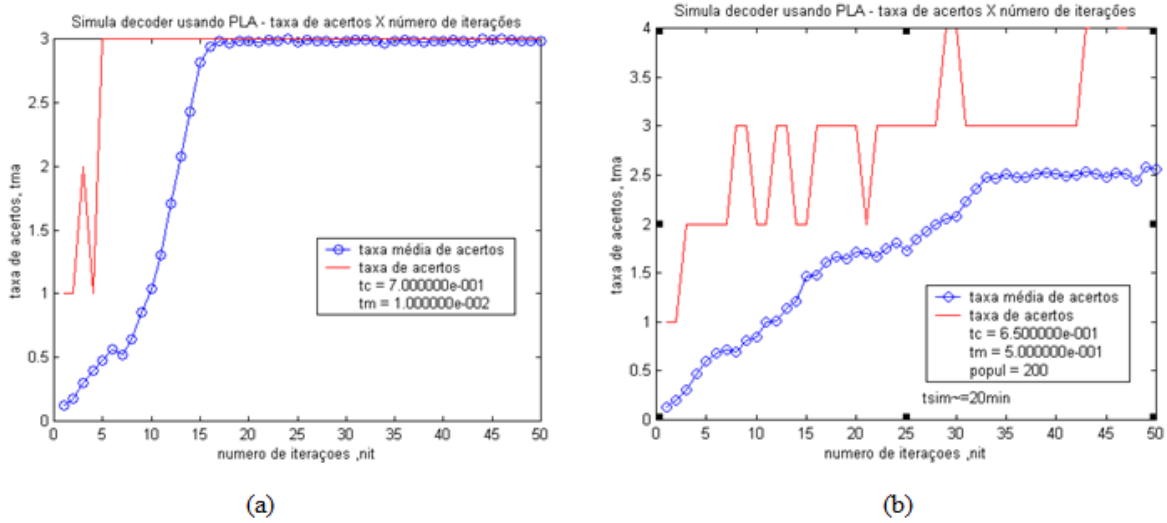


Figura 78 - Número de iterações x taxa de acertos para o decodificador

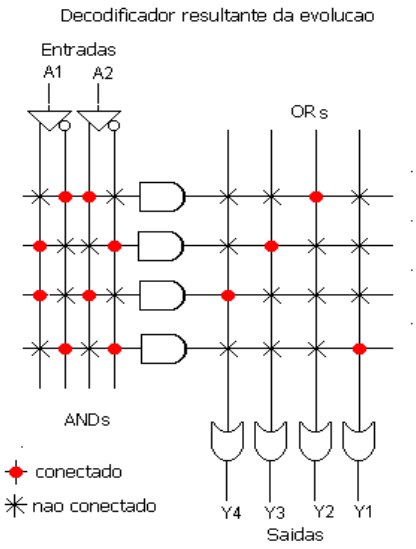


Figura 79 – Decodificador resultante

- Multiplexador: com 4 canais de entrada e 2 entradas de seleção, à taxas de t_c=65% e t_m=6%, em uma população de 200 indivíduos e 30 iterações, a Figura 80 mostra o número de iterações em função da taxa de acertos. O circuito que representa uma das melhores soluções obtido em 10 execuções é dado pelo cromossoma:

Fusível = 01011000110010100101100010110100000 11 - Fitness = 64

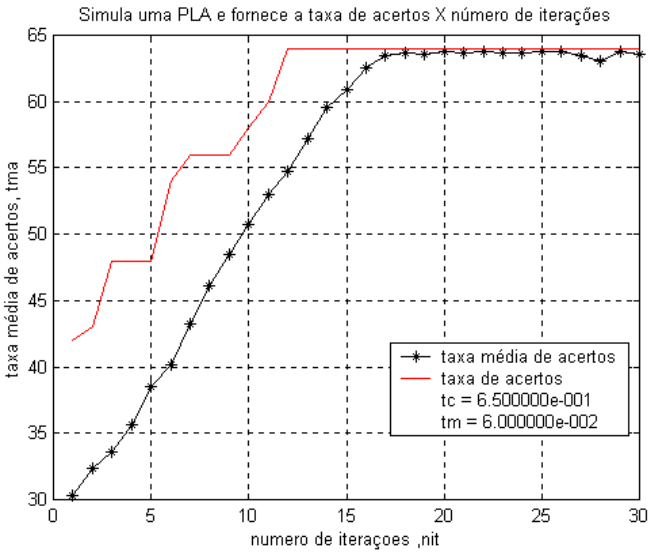


Figura 80 - Número de acertos x taxa de acertos para o multiplexador