UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Síntese de Filtros Passivos com Algoritmo Genético Orientado a Objetos

Orlando Verducci Junior

Itajubá, Março de 2015

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Orlando Verducci Junior

Síntese de Filtros Passivos com Algoritmo Genético Orientado a Objetos

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciências em Ciência e Tecnologia da Computação

Área de concentração: Sistemas de Computação

Orientador: Prof. Dr. Paulo César Crepaldi

Coorientador: Prof. Dr. Leonardo Breseghello Zoccal

Março de 2015

Itajubá - MG

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Orlando Verducci Junior

Síntese de Filtros Passivos com Algoritmo Genético Orientado a Objetos

Dissertação aprovada por banca examinadora em 20 de março de 2015, conferindo ao autor o título de ***Mestre em Ciências em Ciência e Tecnologia da Computação.***

Banca Examinadora:

Prof. Dr. Paulo César Crepaldi (Orientador)

Prof. Dr. Leonardo B. Zoccal (Coorientador)

Prof. Dr. Robson Luiz Moreno

Prof. Dr. José Feliciano Adami

Itajubá 2015

DEDICATÓRIA

Aos meus pais,
Orlando Verducci (in memoriam)
e Therezinha Soares Verducci.

AGRADECIMENTOS

À minha esposa, Andréa, pelo carinho, apoio e paciência dedicados a mim durante todo o período em que estive empenhado e concentrado neste trabalho.

Aos colegas do LNA (Laboratório Nacional de Astrofísica), Francisco Rodrigues, Bruno Vaz Castilho de Souza e Antonio César de Oliveira, que, muito mais do que chefes, foram referências para mim, que me possibilitaram crescer pessoal e profissionalmente nos últimos anos.

Aos Professores Luiz Lenarth Gabriel Vermaas e André Bernardi, que me apoiaram e me orientaram no início de tudo, quando esta conquista ainda era apenas um sonho distante.

A todos os colegas alunos do Programa de Mestrado em Ciência e Tecnologia da Computação da UNIFEI que estiveram comigo e compartilharam seus conhecimentos durante a árdua, porém gratificante, rotina de estudos, provas, trabalhos e seminários.

Ao Prof. Otávio Augusto Salgado Carpinteiro pelas aulas cativantes e esclarecedoras que me proporcionou e por ter sido sempre tão receptivo, oferecendo ajuda, conselho e orientação, que tanto contribuíram para as escolhas que fiz durante o meu Mestrado.

Ao Prof. Tales Cleber Pimenta, que foi o maior responsável pela apresentação na conferência IEEE ICM2014 do artigo gerado a partir deste trabalho.

Ao Prof. Leonardo Breseghello Zoccal, coorientador neste projeto, pelos seus ensinamentos que tive o privilégio de receber nos últimos anos e pela preciosa colaboração na revisão desta Dissertação, sempre atento à coesão do texto e à clareza do argumento.

Ao meu orientador, Prof. Paulo César Crepaldi, que idealizou e coordenou a realização deste projeto, sempre acreditando em mim e disponibilizando boa parte do seu valioso tempo para me ensinar, dirimir minhas dúvidas, alertar sobre meus equívocos e aprimorar a pesquisa realizada.

RESUMO

Este projeto de pesquisa descreve o desenvolvimento de um algoritmo evolucionário, mais especificamente um algoritmo genético, usado para sintetizar de forma automática circuitos elétricos analógicos. O contexto do trabalho é o desenvolvimento de filtros passivos RLC de até três componentes e com escolhas da frequência de corte e do tipo de filtro: passa-baixa, passa-alta, passa-banda ou rejeita-banda. A avaliação de cada solução gerada foi realizada calculando as tensões no circuito por Análise das Tensões Nodais para as diversas topologias possíveis, sem utilizar simuladores ou circuitos eletrônicos programáveis, tais como FPGA e FPAA. O algoritmo genético foi totalmente desenvolvido com linguagem orientada a objetos, Java™, para permitir flexibilidade ao desenvolvedor que queira modificar os métodos envolvidos nas etapas de execução do algoritmo e, assim, comparar resultados. O algoritmo foi construído a partir de um diagrama de classes que indica as relações entre população, indivíduo (circuito candidato), cromossomo (representação genética do circuito), método de seleção, cruzamento, mutação, avaliação do indivíduo (qualidade do circuito), entre outras classes pertinentes.

ABSTRACT

This research describes the development of an evolutionary algorithm, particularly a genetic algorithm, used in order to synthesize analog circuits automatically. The context of this work is the development of passive RLC filters with up to three components, by choosing cutoff frequency and type of filters (low pass, high pass, band pass or notch). The evaluation of each solution was performed by calculating the circuit voltages by nodal analysis for the various possible topologies without using simulators or programmable hardware, such as FPGA and FPAA. The genetic algorithm was entirely developed on object-oriented language, Java™, to allow flexibility for the users who wish to modify some methods used during the execution and, thus, compare results. The algorithm has been implemented from a class diagram that shows the relationships between population, individual (candidate circuit), chromosome (genetic representation of the circuit), selection method, crossover, mutation, evaluation of the individual (quality of the circuit), and other classes.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

AE – Algoritmo Evolucionário

AG – Algoritmo Genético

AGOO – Algoritmo Genético Orientado a Objetos

AN – Análise Nodal (método por)

C – Capacitância

DNA – *Deoxyribonucleic acid* (Ácido desoxirribonucleico)

EE – Estratégia Evolucionária

f_c – Frequência de corte para o filtro passivo

KCL – *Kirchhoff Current Law* (Lei de Kirchhoff para a Corrente)

L – Indutância

PA – (Filtro) Passa-Alta

PB – (Filtro) Passa-Baixa

PE – Programação Evolucionária

PF – (Filtro) Passa-Faixa

PG – Programação Genética

R – Resistência

RF – (Filtro) Rejeita-Faixa

RLC – Circuito elétrico passivo contendo resistores, indutores e capacitores

RNA – *Ribonucleic acid* (Ácido ribonucleico)

UML – *Unified Modeling Language* (Linguagem de Modelamento Unificada)

VLR – *Variable Length Representation* (Representação de Tamanho Variável)

SUMÁRIO

1. INTRODUÇÃO

1.1 A Teoria da Evolução

As descobertas arqueológicas dos últimos duzentos anos mostram que os seres vivos existentes hoje não são iguais aos que habitavam a Terra há 10 milhões de anos, e esses, por sua vez, são distintos dos que aqui viveram há 100 milhões de anos. Ao longo da existência de nosso planeta, as espécies que aqui habitam foram sofrendo modificações, muito lentas, porém constantes. É possível inferir, baseado nessas evidências de eras anteriores, que as espécies continuarão mudando nos próximos séculos.

Há muitos fatores que explicam o fenômeno da evolução das espécies, entre os quais estão as mudanças no ambiente em que os seres vivos se encontram e os mecanismos biológicos da reprodução (HALDANE, 1932). Este processo de variação, ou adaptação, foi investigado por Charles Darwin em sua obra mais conhecida, “A Origem das Espécies”, publicada em 1859, em cujo texto a teoria da evolução foi proposta para explicar a diversidade das espécies em várias partes do planeta.

Segundo a Teoria da Evolução de Darwin, hoje aceita por quase toda a comunidade científica da área, mas que, à época, gerou muita polêmica e críticas, os indivíduos de um ecossistema competem por recursos limitados essenciais à sua sobrevivência e os menos competitivos têm menos chances de se desenvolver, procriar e, portanto, transmitir suas características às gerações seguintes; o que faz com que as espécies pouco adaptadas ao meio em que vivem sejam condenadas à extinção ao longo de muitas gerações, enquanto que os mais competitivos tendem a se proliferar. Havendo mudanças no ambiente, porém, algumas espécies podem tirar proveito da nova situação e se tornarem mais competitivas, o que as levarão a viver mais, a ter mais descendentes e serem mais populosas. Esse processo de privilegiar as espécies mais aptas para a sobrevivência em um determinado meio ambiente é conhecido por seleção natural**,** que garante a evolução natural dos seres vivos ao longo de muitas gerações e a diversidade de espécies no planeta (DARWIN, 1859). É importante destacar que esta evolução natural não é um processo determinístico, que busca melhorar as características do indivíduo, uma vez que não há garantias de que os descendentes de pais bem adaptados também o sejam, nem o inverso, uma vez que pais pouco adaptados podem ocasionalmente gerar descendentes mais aptos para o ambiente (LINDEN, 2012). Era exatamente isso que Charles Darwin desconhecia: o funcionamento do mecanismo de transmissão das características físicas dos indivíduos aos seus descendentes. A compreensão

deste mecanismo de hereditariedade teve início com o descobrimento da unidade básica de informação das características das espécies, o gene, assim denominado por Gregor Mendel, monge e botânico austríaco, em seu artigo sobre a hibridização de plantas (MENDEL, 1865).

Anos mais tarde, com a descoberta da molécula de DNA (MIESCHER, 1871), o ácido desoxirribonucleico, presente no núcleo das células e que armazenam as informações de hereditariedade para gerar as proteínas necessárias para o desenvolvimento dos seres vivos, ficou compreendido que o processo de transmissão das características se dá em nível molecular. Os cromossomos, presentes no núcleo das células, se organizam em pares e contêm moléculas de DNA com cadeias de genes codificadores das características de cada indivíduo (LEVENE, 1919). As moléculas de DNA contêm toda a informação necessária para se replicar e também para se transformar em RNA, ácido ribonucleico, que é o estágio intermediário para o DNA sintetizar proteínas. Estas são cadeias de aminoácidos responsáveis por todo o trabalho realizado dentro de uma célula. Os diferentes tipos de proteínas são responsáveis pelo desenvolvimento das diferentes células existentes no organismo de qualquer ser vivo e, ainda, as diferentes combinações na cadeia de aminoácidos que formam as proteínas produzem diferentes indivíduos, cada um com características próprias, que o tornam único. Em outras palavras, o DNA codifica toda a informação necessária para sintetizar os tipos e as quantidades corretas de proteínas para a formação de um indivíduo de uma determinada espécie, com características próprias, que podem ser vantajosas ou não para a sua sobrevivência no meio em que se encontra (AVERY et al, 1944).

A informação genética de um indivíduo é passada à geração seguinte por meio da reprodução. A reprodução pode ser assexuada, quando é necessário apenas um indivíduo para gerar outros da mesma espécie - comum em bactérias - ou sexuada, quando dois indivíduos de sexos distintos trocam material genético para produzir um ou mais descendentes – como ocorre, por exemplo, com os seres humanos (SMITH, 1978).

A troca de material genético na reprodução sexuada é chamada de cruzamento, processo em que parte do cromossomo de um participante é trocada com parte do cromossomo do outro, conforme ilustrado na Figura 1.1, para gerar um ou mais indivíduos (LINDEN, 2012).

Os cromossomos podem realizar o cruzamento em vários pontos e, assim, trocar muitos genes simultaneamente. Durante o cruzamento, o DNA realiza um complexo processo de replicação em que pode cometer falhas, ou seja, alterar equivocadamente o tipo de alguns genes do cromossomo. Essa falha é chamada de mutação (RIDLEY, 1996). A mutação pode ser causada também por fatores externos como, por exemplo, a incidência de radiação do

ambiente nas células. Há mecanismos de correções na atividade celular que garantem o baixo índice de ocorrência de mutação, porém quando elas ocorrem o efeito nem sempre é prejudicial ao indivíduo (ZEBULUM, 2001). A mutação pode ter consequências positivas ou simplesmente neutras do ponto de vista de adaptação ao meio ambiente (LINDEN, 2012). A mutação genética, desconhecida por Darwin, também foi determinante para a evolução das espécies, visto que, uma mutação favorável durante a reprodução gera um indivíduo mais adaptado, que tenderá a ter vida mais longa, reproduzir-se mais e repassar a sua vantajosa carga genética aos seus descendentes.

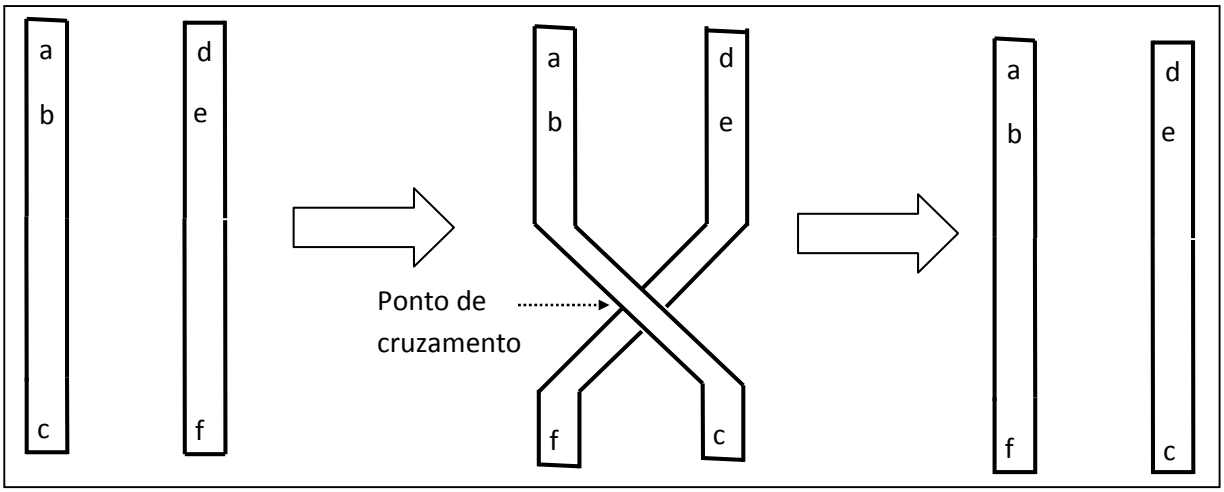


Figura 1.1 Cruzamento entre cromossomos, adaptado de (LINDEN, 2012).

A relação entre a teoria da evolução com a genética serviu de inspiração para os algoritmos evolucionários, os quais foram desenvolvidos de forma a imitar a natureza para solucionar problemas de difícil tratamento e nas mais diversas áreas do conhecimento científico.

1.2 Computação Inspirada na Evolução Biológica

No final da década de 1950 e início dos anos 1960, alguns cientistas começaram a aplicar os princípios da evolução biológica para desenvolver novos modelos computacionais buscando soluções ótimas (FRASER, 1958) (BARRICELLI, 1962). De maneira independente e na mesma época, outros cientistas desenvolveram algoritmos baseados em princípios semelhantes, gerando algoritmos mais poderosos, aplicados a uma grande variedade de problemas (RECHENBERG, 1965) (HOLLAND, 1975). Essas novas ferramentas

computacionais, inspiradas na teoria da evolução, receberam o nome de Algoritmos Evolucionários (AE).

Segundo Zebulum (2001), os AEs são utilizados para resolver problemas da forma:

$$f : S \rightarrow R \qquad (1.1)$$

Onde S é o espaço de busca por soluções, constituído de todas as possíveis soluções para o problema considerado e pode ser um valor n-dimensional de valores binários, inteiros, reais ou qualquer estrutura de dados mais complexa. A todo valor em S está associado um valor real em R, representando sua medida de adequação ao problema (ZEBULUM, 2001). Em outras palavras:

Para todo s $\in$ S, f(s) $\in$ R é a função de avaliação de qualquer solução em S e deve ser usada para nortear a busca por soluções melhores.

A Figura 1.2 apresenta um pseudocódigo que resume o comportamento de um algoritmo evolucionário:

```
t:=0;                                    //inicia o contador de gerações;
Inicializa_População P(0)                //primeira população (aleatória);
Enquanto não terminar faça               //termina por geração ou avaliação;
    Avalie_População P(t)                //cada indivíduo é avaliado ;
    P':=Selecione_Pais P(t)              //escolhe uma subpopulação P';
    P':=Recombinação_e_mutação P'        //modificar P' com operadores
                                         //probabilísticos;
    Avalie_População P'                  //avaliar os indivíduos de P';
    P(t+1)=Selecione_sobreviventes P(t),P' //escolha dos melhores;
    t:=t+1                               //incrementa o n° de gerações;
Fim enquanto                             //condição de parada satisfeita.
```

Figura 1.2 Pseudocódigo básico para um algoritmo evolucionário (LINDEN, 2012).

O algoritmo se inicia com a geração aleatória de uma população inicial, que servirá como ponto de partida para a evolução dos indivíduos. A parte principal do algoritmo é executada enquanto não houver uma condição de parada satisfeita, que pode ser a obtenção de

uma solução cuja avaliação supera o valor mínimo exigido ou o limite máximo de gerações alcançado.

O ciclo de execução é composto pela execução da avaliação de todos os indivíduos, atribuindo uma nota a cada um deles; a seleção de forma probabilística de alguns indivíduos para participarem da geração de filhos; o procedimento de recombinação entre os pais escolhidos e aplicação de mutação aleatória; a avaliação para os filhos gerados; a escolha dos melhores entre os filhos gerados e o restante da população; e, finalmente, o incremento da contagem de gerações para o controle de parada.

Desde a obtenção da população inicial até os procedimentos de seleção de pais e geração de filhos, o processo é essencialmente estocástico e isto significa que os resultados obtidos em várias execuções do algoritmo não são perfeitamente repetidos. Apesar do nome, os algoritmos evolucionários são heurísticas que não asseguram a obtenção da melhor solução em todas as execuções (LINDEN, 2012).

Os problemas para os quais é possível empregar um algoritmo cujo tempo de execução é suficientemente curto, não há a necessidade do uso de um AE (LINDEN, 2012) e, neste caso, é preferível optar por um algoritmo dedicado, como, por exemplo, um algoritmo para ordenar de forma crescente uma sequência de 1000 números inteiros. Como existem vários algoritmos dedicados à ordenação, não é viável usar um AE para essa tarefa. Os AEs são técnicas de busca que devem ser usadas quando não há um algoritmo exato para a solução do problema em questão ou, quando há, o seu uso se torna inviável devido ao tempo de execução ser extremamente longo. Exemplo: encontrar a solução para o problema do caixeiro viajante, que consiste em determinar a menor rota para percorrer uma série de cidades (visitando cada uma pelo menos uma vez), retornando à cidade de origem.

Diferentemente de outros métodos de busca, os AEs não são puramente probabilísticos, pois usam as informações a respeito da adequação das soluções candidatas em cada ciclo de execução para guiar a sua pesquisa. Essa característica pode ser explorada para se obter AEs mais eficientes em diferentes campos de aplicação, como: cálculo do roteamento das trilhas em placas de circuito impresso; otimização da distribuição de professores e disciplinas na grade de aulas; obtenção da topologia com portas lógicas em circuitos integrados digitais; otimização dos coeficientes proporcional, integral e diferencial de um compensador PID para uma planta de controle; obtenção da distribuição de pesos para uma rede neural artificial, entre outros.

1.3 Objetivos

Este trabalho tem por objetivo propor um Algoritmo Evolucionário, mais especificamente um Algoritmo Genético (AG), que é detalhado nas seções seguintes, construído com uma linguagem orientada a objetos, Java™, a partir de um modelo descrito por um diagrama de classes que descreve os relacionamentos entre os elementos que compõe o problema de se projetar um circuito elétrico analógico que execute a função de um filtro passivo RLC a partir de alguns requisitos de desempenho inseridos pelo usuário.

O modelo proposto tem por objetivo servir de base para construção de AGs mais elaborados que sintetizem circuitos mais complexos.

Outro objetivo é explorar a capacidade do AG em gerar circuitos de topologias variadas com desempenho semelhante, ou até melhor, quando comparados aos circuitos que poderiam ser desenvolvidos por um projetista, justificando assim o uso do AG como uma ferramenta para busca de soluções alternativas àquelas que poderiam ser concebidas por um profissional, que utiliza sua experiência e intelecto para elaborar um projeto.

1.4 Estrutura do Trabalho

Este capítulo introdutório apresentou a teoria da evolução e como esta serviu de inspiração para o desenvolvimento dos algoritmos baseados na evolução biológica. Em seguida, os objetivos deste trabalho de pesquisa foram expostos para que fiquem claros a importância e o contexto em que estão inseridos os temas dos próximos capítulos, de forma que sirvam de embasamento teórico para o projeto final apresentado.

O segundo capítulo trata dos Algoritmos Evolucionários (AE), apresentando conceitos, operadores genéticos, ciclos básicos de execução e destacando quatro categorias de AE: a Programação Evolucionária, a Estratégia Evolucionária, o Algoritmo Genético e a Programação Genética.

No capítulo 3, é proposto um modelo de uso geral de Algoritmo Genético para ser desenvolvido com orientação a objetos: o AGOO (Algoritmo Genético Orientado a Objetos). Assim, a descrição do funcionamento dos operadores genéticos do AG, a descrição das classes construídas para o modelo e os diagramas de classes e de sequência da UML são apresentados neste capítulo.

O Capítulo 4 é dedicado à análise dos filtros passivos, abordando o método das tensões nodais para obtenção da tensão do nó de saída e alguns conceitos de resposta em frequência

dos filtros, pois estes são os circuitos a serem sintetizados de forma automática pelo algoritmo do projeto final.

O capítulo 5 descreve o desenvolvimento de um AGOO aplicado à síntese de filtros passivos, incluindo a forma de representação do circuito solução, a função de avaliação e a descrição dos atributos e métodos das classes do modelo proposto. No final do capítulo são apresentados os circuitos obtidos após várias execuções do algoritmo, para diferentes requisitos de projeto.

O capítulo 6 apresenta a conclusão do trabalho, ou seja, as considerações finais sobre o AG proposto e as possibilidades de aplicação em projetos futuros.

2. ALGORITMOS EVOLUCIONÁRIOS

O objetivo de qualquer AE é realizar uma busca entre as possíveis soluções de um problema para garantir que seja encontrada uma solução satisfatoriamente boa, ainda que não seja a melhor possível. Para realizar essa tarefa, três conceitos fundamentais para execução de um AE devem ser definidos: a representação da solução, a sua função de avaliação e os operadores genéticos aplicados às soluções provisórias (LINDEN, 2012). A maneira como os conceitos fundamentais dos AEs são utilizados podem variar de acordo com o tipo de problema a ser resolvido, o que os divide em algumas categorias particulares, que são apresentadas no item 2.7.

A seguir são definidos e discutidos os conceitos necessários para o desenvolvimento de um AE e algumas formas de processar computacionalmente esses conceitos.

2.1 Representação

Para todo problema a ser tratado com AE deve ser definida uma representação cromossômica da sua solução para que esta possa ser processada por computadores digitais.

Há várias formas de representar uma solução, que variam conforme o problema a ser resolvido. Exemplos: um número real pode representar bem a solução para uma função matemática; uma sequência binária pode representar um caminho de roteamento para as trilhas condutoras de uma placa de circuito impresso; uma estrutura de dados pode representar a topologia e os componentes de um circuito elétrico; uma estrutura em árvore pode representar um programa computacional, e assim por diante.

A representação da solução será o correspondente ao cromossomo dos seres vivos no meio ambiente, pois ela, a partir de uma escolha aleatória inicial, deve ser evoluída para se tornar apta a resolver o problema proposto. Para a escolha da representação utilizada no AE, devem ser considerados dois atributos importantes: a cardinalidade e o comprimento (ZEBULUM, 2001).

2.2 Cardinalidade

Um conceito importante que deve ser compreendido para o desenvolvimento de qualquer AE é a cardinalidade da representação escolhida para a solução do problema. A cardinalidade refere-se ao número de possibilidades para o valor de cada unidade básica do

cromossomo (ZEBULUM, 2001) e corresponde aos tipos atribuídos a cada gene do cromossomo dos seres vivos. Citosina (C), Guanina (G), Adenina (A) e Timina (T) são os quatro tipos possíveis de bases nitrogenadas presentes em cada gene no DNA, portanto a sua cardinalidade é 4. Voltando aos AEs, no caso de uma representação binária, cada bit do cromossomo assume valor 0 ou 1, portanto, a cardinalidade é 2. Para uma estrutura de dados contendo apenas valores inteiros (de um byte, por exemplo) em cada campo, a cardinalidade é 256.

2.3 Comprimento

O comprimento se refere à quantidade de genes usada na representação cromossômica da solução para o problema. Qualquer que seja a representação escolhida, devem-se considerar duas formas de tratar o seu comprimento: a representação de comprimento fixo e a de comprimento variável (ZEBULUM, 2001). Em geral, toda solução pode ser representada das duas formas, cabendo ao desenvolvedor determinar qual é a mais adequada para o seu problema.

A representação cromossômica com comprimento fixo limita a quantidade de genes que compõem a solução do problema, mas pode ser usada em muitos casos. Por exemplo, a solução para uma equação polinomial de grau *n* tem necessariamente *n* soluções e, portanto, pode ser representada por uma estrutura de comprimento fixo para conter todas as raízes da equação, ainda que uma estrutura de comprimento variável também seja possível, desde que possa crescer até conter todos os valores das raízes. Uma representação para um circuito elétrico pode ter comprimento fixo se a quantidade de componentes para o circuito que se procura for conhecida; ou ter comprimento variável, se a quantidade de componentes do circuito procurado for ignorada.

O valor da cardinalidade elevada ao comprimento fornece o número de combinações distintas para a representação cromossômica da solução, ou, em outras palavras, o conjunto do espaço de busca a ser explorado pelo AE (ZEBULUM, 2001). Se o comprimento da representação for variável ou a cardinalidade for infinita, o espaço de busca será ilimitado. Por outro lado, representações com comprimento fixo e cardinalidade finita impõem um limite para o espaço de busca, embora, geralmente, esse limite seja inatingível pelo algoritmo, devido ao grande número de combinações possíveis. Por exemplo, se uma representação de solução é formada por um vetor fixo de 8 bytes, ou seja, comprimento igual a 8 e

cardinalidade igual a 256, o número total de soluções distintas candidatas ao problema será: $256^8 = 2^{64} = 1{,}84 \times 10^{19}$. Considerando que um programa computacional avalie cada solução em 1ns (10^{-9}s), o seu tempo de execução para avaliar sequencialmente todas as combinações possíveis de solução seria de 584 anos! Como é apresentada adiante, a execução de um AE, aplicado a problemas com espaço de busca com ordem de grandeza semelhante ao do exemplo dado, é capaz de encontrar soluções satisfatórias em alguns minutos.

2.4 Avaliação

A função de avaliação executada pelo AE é o equivalente ao meio ambiente em que as soluções candidatas ao problema devem se adaptar. O valor que a representação de uma solução candidata obtém nesta avaliação nos diz o quão boa é essa solução. E isso tem impacto nas chances desta solução transmitir suas características para as próximas gerações. As soluções cuja representação tem as melhores notas de avaliação são as mais adaptadas ao seu ambiente, ou seja, à solução do problema proposto.

Uma representação que contém uma solução x_n candidata à raiz da função f(x) deve receber nota alta se o valor da função se aproxima de zero quando a variável independente x assume o valor da solução candidata x_n. Neste caso, a função de avaliação da solução de ordem n poderia ser definida, apenas para apresentar uma possibilidade, como mostrado na equação (2.1):

$$A_n = \frac{100}{1 + |f(x_n)|} \tag{2.1}$$

Na forma de avaliação escolhida acima, observa-se que quando $f(x_n)$ se aproxima de zero, sendo positivo ou negativo, o valor de sua avaliação, A_n, se aproxima de 100, que seria a nota máxima. A importância de estabelecer uma nota máxima deve-se, simplesmente, ao fato de se ter uma referência para a verificação da condição de parada para o algoritmo, como é visto mais adiante.

2.5 Operadores Genéticos

Para que haja evolução na busca por soluções, o AE deve prover uma ou mais operações que permitam a obtenção de uma nova solução a partir de outras já existentes. Cada ciclo de renovação de soluções é chamado de geração (LINDEN, 2012)**.** As principais operações genéticas utilizadas na obtenção de novas gerações são: o cruzamento, a mutação e

a seleção de pais. Cada operação genética pode ser executada de diferentes formas, como é visto a seguir.

2.5.1 Cruzamento

Como na reprodução dos seres vivos, o cruzamento de partes do cromossomo de soluções existentes permite o surgimento de uma nova solução, que fará parte de uma nova geração de soluções candidatas ao problema a ser resolvido.

Se, por exemplo, as soluções são representadas por cromossomos binários, deve-se escolher um ou mais pontos de corte para realizar o cruzamento (LINDEN, 2012). Os pontos de corte são posições de bit ao longo do cromossomo e são gerados por alguma função randômica da linguagem de programação escolhida. No caso do Java™, a função para gerar valores aleatórios é a `Math.random()`, que deve ser interpretada como a chamada do método “random()” da classe “Math”, que retorna um valor de ponto flutuante entre 0 e 1. Multiplicando-se o valor de retorno de random() por qualquer valor inteiro N e tomando-se a parte inteira do resultado, tem-se um número inteiro gerado aleatoriamente entre 0 e N-1. A Figura 2.1 exemplifica um cruzamento de dois pais de 16 bits; cada par gera dois filhos usando dois pontos de corte. Os filhos gerados são formados por partes de cada pai, definidas pelos pontos de corte: do 1º. ao 5º. bit (parte A), do 6º. ao 13º. bit (parte B) e do 14º. ao 16º. bit (parte C). O filho 1 recebe a parte A e a parte C do primeiro pai, e a parte B do segundo; já o filho 2 recebe a parte A e a parte C do segundo pai, e a parte B do primeiro. Os valores dos pais e os pontos de corte usados neste exemplo foram escolhidos de forma aleatória, apenas para apresentar um caso prático e, assim, facilitar a compreensão do mecanismo de cruzamento que deve ser executado pelo algoritmo.

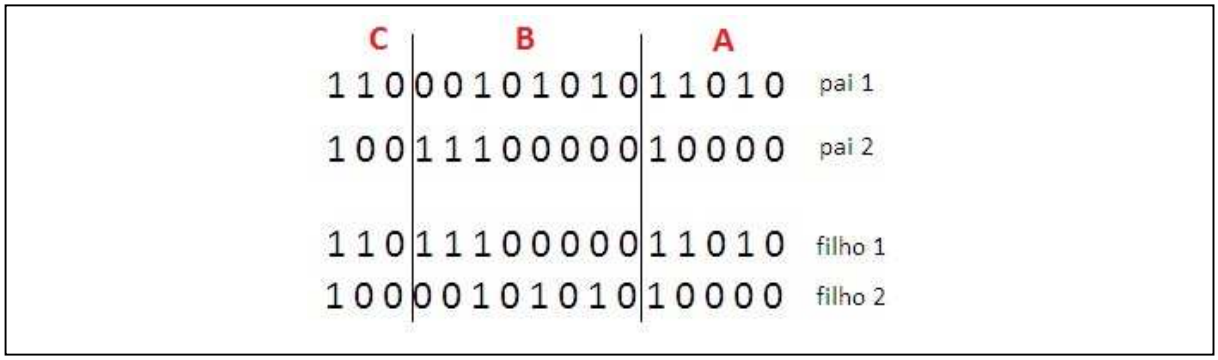


Figura 2.1 Operador genético de cruzamento para cromossomos binários com pontos de corte no 5º e 13º bits.

Outra possibilidade de realização é o denominado cruzamento uniforme, que, em vez de usar pontos de corte, faz uso de um indivíduo auxiliar que serve de máscara para a troca de genes. A máscara indica qual gene dos pais vai para o filho 1 e qual vai para o filho 2, conforme está ilustrado na Figura 2.2. A cada bit 1 da máscara, o filho 1 recebe o bit correspondente do primeiro pai e o filho 2 recebe o bit correspondente do segundo; e a cada bit 0 da máscara a atribuição se inverte: o filho 1 recebe o bit correspondente do segundo pai e o filho 2 recebe o bit correspondente do primeiro.

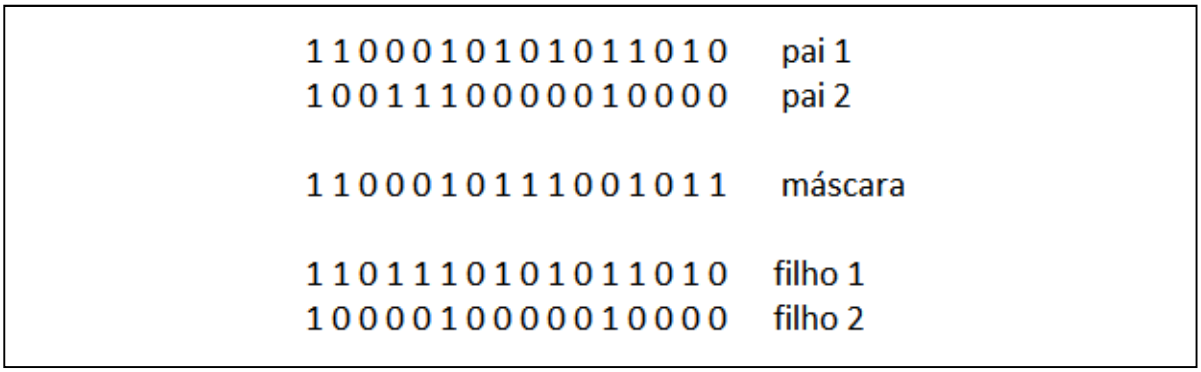


Figura 2.2 Operador genético de cruzamento uniforme para representação binária.

O cruzamento pode ser realizado também com representações que usam números reais. Neste caso, os pontos de corte são definidos por um valor α, $0 \leq \alpha \leq 1$,e gerado aleatoriamente pelo algoritmo, que permite extrair frações do valor dos genes de cada pai para gerar os genes dos filhos (LINDEN, 2012).

Novamente, tomem-se, como exemplos, dois valores para os pais e um valor para α, aleatórios, para que a descrição do mecanismo seja feita a partir de um caso prático:

Exemplo 1: Pai 1 = 12.30 7.50 231.27 0.68
Pai 2 = 7.20 9.90 325.00 0.14

Considere que foi gerado o ponto de corte $\alpha = 0.25$:

Filho 1 = $12.3*\alpha + 7.2(1 - \alpha)$ $= 12.3*0.25 + 7.2*0.75 =$ 8.48
$7.5*\alpha + 9.9(1 - \alpha)$ $= 7.5*0.25 + 9.9*0.75 =$ 9.30
$231.27*\alpha + 325(1- \alpha)$ $= 231.27*0.25 + 325*0.75 =$ 301.57
$0.68*\alpha + 0.14(1- \alpha)$ $= 0.68*0.25 + 0.14*0.75 =$ 0.28

Filho 2 =	$12.3(1 - \alpha) + 7.2*\alpha$	$= 12.3*0.75 + 7.2*0.25 =$ 11.03
	$7.5(1 - \alpha) + 9.9*\alpha$	$= 7.5*0.75 + 9.9*0.25 =$ 8.10
	$231.27(1 - \alpha) + 325*\alpha$	$= 231.27*0.75 + 325*0.25 =$ 254.70
	$0.68(1 - \alpha) + 0.14*\alpha$	$= 0.68*0.75 + 0.14*0.25 =$ 0.55

Portanto,

Filho 1 =	8.48	9.30	301.57	0.28
Filho 2 =	11.03	8.10	254.70	0.55

Outra forma de realizar o cruzamento para cromossomos representados com números reais é através de um vetor máscara com valores de α aleatórios. Assim é possível realizar um cruzamento uniforme, semelhante ao que foi feito para a representação binária.

Neste caso, se na posição i da máscara, $\alpha_i \geq 0.5$ o filho1 herda o gene *i* do pai1 e o filho2, o gene *i* do pai2; e se $\alpha_i < 0.5$ a herança é inversa. O valor 0.5 escolhido se deve ao fato de que este é o ponto médio entre 0 e 1, faixa de valores gerados pela função randômica, o que faz com que as chances de se aplicar cada forma de herança sejam iguais.

Tomem-se os mesmos pais do exemplo anterior e um vetor máscara aleatório como exemplo:

Exemplo 2:	Pai 1 =	12.30	7.50	231.27	0.68
	Pai 2 =	7.20	9.90	325.00	0.14
	Máscara =	0.28	0.57	0.33	0.81
	(	<0.5	≥0,5	<0.5	≥0,5)
	Filho 1 =	7.20	7.50	325.00	0.68
	Filho 2 =	12.30	9.90	231.27	0.14

Esta segunda forma de cruzamento, na maioria dos problemas com representação usando reais, é menos vantajosa, pois exige mais sorteios para gerar o vetor dos α_i (pode-se também usar uma máscara binária, como indicado no exemplo acima) e gera menos diversidade de cromossomos na população, pois os α_i (ou os bits da máscara binária) servem apenas como indicação do sentido da troca de genes completos entre os pais, em vez de particionarem cada gene, como foi visto no caso anterior. Este método de cruzamento apenas adapta o cruzamento uniforme binário, aplicando-o aos vetores de números reais, mas pode

ser satisfatório quando o comprimento da representação é grande o suficiente para garantir uma boa diversidade de cromossomos.

2.5.2 Mutação

Após o cruzamento, a representação cromossômica de cada filho resultante é submetida à operação de mutação, respeitando a chamada taxa de mutação, que é um dos parâmetros estabelecidos para a execução do AE.

A taxa de mutação se refere à probabilidade de ocorrer a mutação em cada solução gerada pelo cruzamento dos cromossomos de seus pais. Por exemplo, se a taxa de mutação de um AE for de 5% significa que, em média, uma em cada vinte soluções geradas é submetida ao operador genético de mutação, que pode ser executado de diversas formas, como a mutação simples, a múltipla ou a uniforme.

A mutação simples ocorre quando apenas um gene do cromossomo é escolhido para sofrer uma alteração, como pode ser observado na Figura 2.3. Para realizar a mutação, neste caso, sorteia-se uma posição de gene a ser modificada com os possíveis valores definidos por sua cardinalidade.

A operação de mutação é múltipla quando dois ou mais genes do cromossomo são escolhidos para a alteração, conforme representado na Figura 2.4.

Já a mutação uniforme necessita de uma máscara para os genes, que define os genes a serem modificados. É usada geralmente para a representação cromossômica binária (LINDEN, 2012). A Figura 2.5 ilustra uma mutação uniforme em que os bits do cromossomo correspondentes a cada bit 1 da máscara sofrem alteração de valor; portanto, os bits que sofrem mutação são invertidos.

As formas de mutação aplicadas aos cromossomos binários podem ser adaptadas aos cromossomos representados como um vetor de números reais: para a mutação simples, deve ser sorteada uma posição do vetor e um novo valor para substituir o elemento desta posição; para a mutação múltipla, sorteiam-se várias posições e novos valores de substituição para cada posição sorteada. As Figuras 2.3b e 2.4b ilustram esses dois casos de mutação com vetor de números reais.

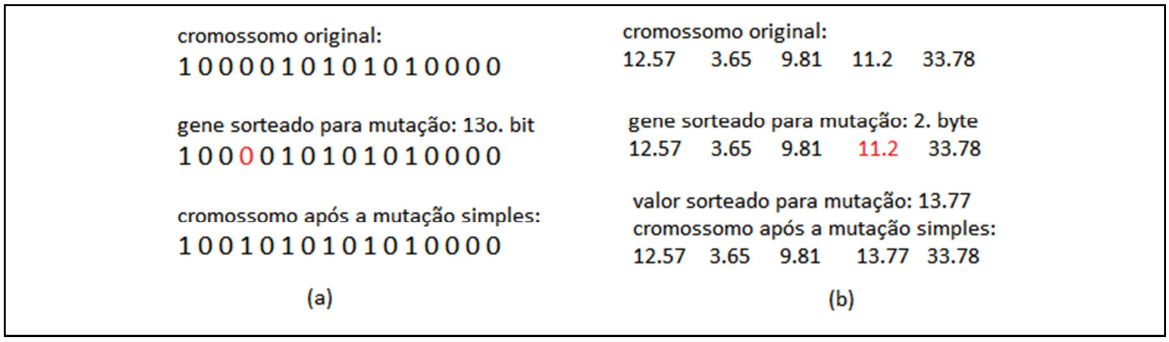


Figura 2.3 Mutação simples: (a) cromossomo binário; (b) cromossomo vetor de reais

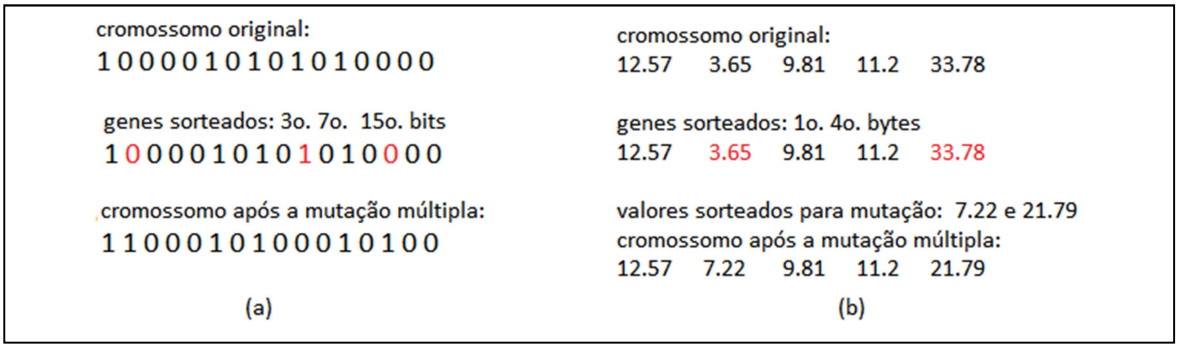


Figura 2.4 Mutação múltipla: (a) cromossomo binário; (b) cromossomo vetor de reais

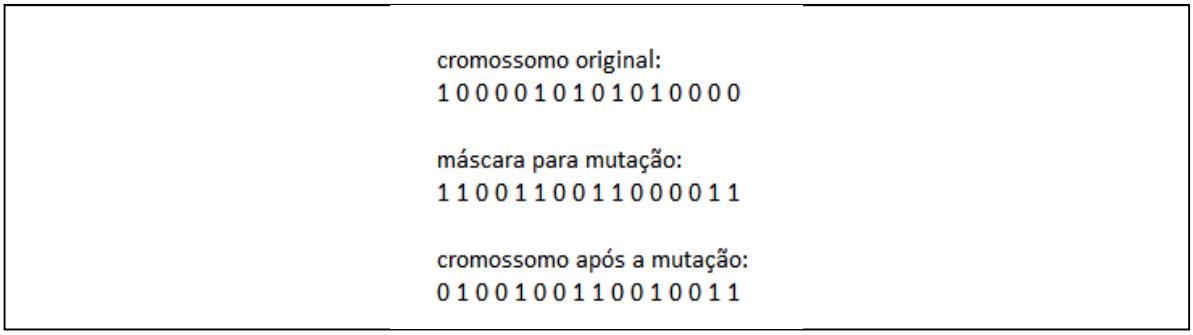


Figura 2.5 Mutação uniforme: utilizada geralmente em cromossomos binários, mas a máscara pode ser usada para indicar os elementos do vetor de reais que devem sofrer mutação.

2.6 Seleção de Pais

Para a realização do cruzamento deve-se antes empregar uma técnica para selecionar os pais que vão gerar os novos indivíduos, porém é importante destacar que nem todo AE faz uso de alguma técnica para seleção dos pais, simplesmente porque as gerações são obtidas apenas com mutações dos indivíduos existentes ou porque não há uma população numerosa de soluções candidatas ao cruzamento que justifique o uso de alguma escolha de pais

(ZEBULUM, 2001). Por exemplo, se a população de um AE for de apenas dois indivíduos, estes são submetidos ao cruzamento para gerar duas novas soluções, que, por sua vez, após uma mutação probabilística, serão então usadas no cruzamento para obtenção de outras duas soluções, e assim por diante, mantendo a população constante de apenas dois indivíduos.

Para os AEs que utilizam uma população numerosa, como os algoritmos genéticos, a forma como será feita a escolha de uma ou mais soluções para o cruzamento é uma das características do projeto que deve ser definida pelo desenvolvedor para que o algoritmo seja o mais adequado para o problema em questão. As principais técnicas de seleção de pais descritas na literatura sobre AGs são: a Roleta Viciada, que promove um sorteio tendencioso para garantir que as soluções com maior nota sejam mais frequentemente sorteadas ao longo da execução; o Torneio, que consiste em se escolher algumas soluções de forma aleatória na população (independentemente de suas notas) e só então escolher as mais bem pontuadas dentre as poucas sorteadas; e o Ranking, que privilegia as melhores colocadas na população, mas levando em conta apenas a posição de cada solução na classificação da população e não o valor de suas notas (LINDEN, 2012). Todas essas técnicas são detalhadas no capítulo dedicado aos algoritmos genéticos, em que são discutidos os aspectos positivos e negativos de cada uma delas.

2.7 Os Principais Algoritmos Evolucionários

Os AEs podem ser divididos nas seguintes categorias: Programação Evolucionária (PE), Estratégia Evolucionária (EE), Algoritmo Genético (AG) e Programação Genética (PG), conforme foram descritas por (ZEBULUM, 2001), e que são apresentadas de forma resumida nesta seção. Cada categoria surgiu em momentos diferentes na história do desenvolvimento dos AEs, guiadas por diferentes demandas de pesquisas científicas e experimentos industriais, a partir da década de 1950.

Fraser (1957) e Friedberg (1958) são considerados os pioneiros em usar processos evolucionários na solução de problemas via computador. Na mesma época, Box (1957) desenvolveu um método de *operação evolucionária* que consistia no uso de uma técnica evolucionária para aplicação industrial em análise e projeto. Anos mais tarde, Bremermann (1962) descreveu a primeira tentativa de solucionar problemas de otimização numérica por evolução simulada. Na década de 1960 já estavam estabelecidos dois modelos evolucionários: a Programação Evolucionária e a Estratégia Evolucionária. Essas técnicas deram origem aos Algoritmos Genéticos e à Programação Genética (ZEBULUM, 2001).

2.7.1 Programação Evolucionária (PE)

Lawrence J. Fogel em (FOGEL, 1962) concebeu este algoritmo evolucionário considerando que a inteligência deveria ser baseada no comportamento adaptativo para atingir objetivos em ambientes variados, conforme descrito em (BÄCK, 1997). A capacidade de predição, segundo Fogel, é a chave para o comportamento inteligente. Fogel sugeriu uma série de experimentos usando evolução simulada de máquinas de estados finitas, chamada de Programação Evolucionária (PE), para prever uma série não estacionária com relação a um critério escolhido arbitrariamente.

A PE introduziu dois conceitos importantes da evolução natural: o processamento de uma população de soluções e o uso do operador de mutação. Conforme foi resumido por De Jong (1997), a ideia básica da PE é:

"Uma população de máquinas de estados finitas é exposta ao ambiente, ou seja, à sequência de símbolos observados ao longo do tempo até o instante atual. Uma função de penalização é usada para avaliar a adequação da máquina de estados. As máquinas filhas são criadas por mutação aleatória – cada pai produz um descendente. Há cinco procedimentos possíveis para a mutação: alteração do símbolo de saída, alteração da transição de estado, adição de um estado, remoção de um estado e alteração do estado inicial. As duas últimas possibilidades só são permitidas quando a máquina de estado geradora contém mais de um estado. As mutações geralmente são escolhidas segundo uma distribuição probabilística, mas também podem ser fixas. Esses descendentes são então evoluídos da mesma maneira como foram seus pais."

A nota de avaliação é atribuída de acordo com o desempenho da máquina de estados finita para uma tarefa em particular. Um conjunto de máquinas de estados é escolhido de forma determinística a partir das notas atribuídas às máquinas. Por exemplo, a escolha pode ser a metade da população de máquinas de estado que tem as melhores notas. Estas sofrem então mutações, criando uma nova população. O processo se repete ciclicamente, a cada nova população. O algoritmo termina quando se obtém uma máquina de estados cuja avaliação atende ao requisito de parada.

2.7.2 Estratégia Evolucionária (EE)

Esta técnica (na verdade, as Estratégias Evolucionárias – EEs - são constituídas por uma família de algoritmos) foi desenvolvida em 1964 por Ingo Rechenberg e Hans-Paulo Schwefel na Universidade Técnica de Berlim para encontrar os parâmetros ótimos em problemas na área de mecânica dos fluidos (RECHENBERG, 1965) (SCHWEFEL, 1965). Posteriormente, a técnica foi generalizada para a resolução de funções reais (FOGEL, 1995).

Os estudos iniciais de Rechenberg e Schwefel tinham como aplicação a robótica e o algoritmo processava um único indivíduo por muitas gerações usando um operador de mutação com distribuição binomial (ZEBULUM, 2001).

Segundo Bäck (1997), a EE mais simples pode ser descrita como uma técnica que processa um indivíduo $X \in R^n$, realizando uma mutação que consiste em somar a X um vetor aleatório com distribuição normal, Z, multiplicado por um escalar $\sigma > 0$. A decisão de seleção é simplesmente determinar se o valor da função de avaliação, f: $R^n \rightarrow R$, do novo elemento é melhor que o do elemento anterior. Supondo que o objetivo seja encontrar X de forma a minimizar f(X), o processo de evolução deve obedecer à regra da equação (2.2):

$$X_{k+1} = \begin{cases} X_k + (\sigma \cdot Z_k) & \text{se } f(X_k + (\sigma * Z_k)) \leq f(X_k) \\ X_k & \text{caso contrário} \end{cases} \qquad (2.2)$$

As mais recentes versões de EE usam o conceito de uma população de μ indivíduos candidatos à solução do problema. A população tem por finalidade a geração de λ filhos, formando uma (μ+λ)EE. Outra diferença encontrada nas EEs atuais é o uso de parâmetros individuais que controlam a distribuição individual de mutação. Assim, o cromossomo é representado por uma tupla $(X,\sigma) \in R^n \times R$ submetida aos operadores genéticos. O vetor $\boldsymbol{\sigma}$ é o parâmetro de controle para o operador de mutação, executada em dois passos (ZEBULUM, 2001):

$$\sigma_{t+1} = \sigma_t \cdot e^{(t \cdot Z_t)} \qquad (2.3)$$

$$X_{t+1} = X_t + \sigma_{t+1} \cdot Z \qquad (2.4)$$

onde Z_t é uma variável de distribuição normal, Z é um vetor aleatório com valores de uma distribuição normal e σ_t é um parâmetro de controle.

É possível, portanto, obter uma família de EEs indicadas por (p+f)EE, onde p é a quantidade de pais disponíveis para a reprodução e f, a quantidade de filhos gerados em cada ciclo do algoritmo:

(1+1)EE: Estratégia evolucionária de dois membros – a mais simples, sendo um pai gerando um filho, de acordo com a regra de mutação da equação (2.2).

(μ+1)EE: Primeira estratégia evolucionária multimembros que introduz o conceito de população, onde μ pais (μ>1) podem participar da geração de um único filho (RECHENBERG, 1973). Usando um operador de recombinação é gerado um filho (X',σ') a partir dos vetores pais (X_0, σ_0), (X_1, σ_1), (X_2, σ_2),..., (X_μ, σ_μ), segundo a regra de cruzamento da equação (2.5):

$$\begin{aligned} X'_k &= X_{\alpha k} \\ \sigma'_k &= \sigma_{\alpha k} \end{aligned} \qquad \forall \alpha \in \{0,1,...,\mu\} \ \forall k \in \{0,1,...,l\} \tag{2.5}$$

onde μ é a quantidade de pais envolvidos no cruzamento para gerar um filho e l é a dimensão do vetor (X', σ'), que representa o filho gerado. O valor de α é um inteiro sorteado entre 0 e μ para cada elemento k do vetor filho. Este cruzamento é denominado *operador de recombinação discreto* e é equivalente ao cruzamento uniforme da representação binária (BÄCK et al, 2000).

Após a recombinação o filho é submetido à mutação da forma apresentada em (2.3) e (2.4). Dentre todos os indivíduos, o menos apto (com menor nota de avaliação) é removido da população e o processo se repete a partir da escolha de outros μ pais para participarem de nova recombinação.

(μ+λ)EE: Estratégia evolucionária multimembros que recombina μ pais para gerarem λ filhos. A seleção de pais para produzir a geração seguinte é escolhida entre os μ+λ indivíduos e em seguida os μ melhores permanecem no próximo ciclo de execução do algoritmo. Assim, é possível haver recombinação (cruzamento) entre pai e filho da mesma geração e ao longo das gerações os pais tendem a ser substituídos pelos filhos que têm melhor nota de avaliação.

(μ,λ)EE: É uma variação da estratégia anterior, proposta por Schwefel (1995), em que apenas os λ filhos participam da seleção de pais para a próxima geração. Assim, o período de

“vida” de cada indivíduo está restrito a uma única geração. Essa estratégia explora o “esquecimento” de parâmetros inapropriados usados nas recombinações e mutações (SCHWEFEL, 1987).

2.7.3 Algoritmo Genético (AG)

Os Algoritmos Genéticos (AGs) foram concebidos por John Holland e sua equipe na Universidade de Michigan (HOLLAND, 1975) e tinham como propósito investigar os processos adaptativos em sistemas naturais para criar programas computacionais com as mesmas características dos sistemas naturais investigados. As pesquisas buscavam obter um sistema artificial com as mesmas propriedades dos sistemas naturais, como a reprodução e a autocorreção. Assim os altos custos associados ao processo de refazer o projeto de um sistema seriam eliminados (ZEBULUM, 2001).

Ao contrário de outras técnicas de otimização, os AGs não se utilizam de informações extras que direcionam a busca para soluções melhores; apenas exploram o espaço de busca que contém todas as soluções candidatas.

Algumas características básicas de todo algoritmo evolucionário, também são aplicadas aos AGs, como a seleção de parte da população para participar das recombinações, os operadores de mutação e cruzamento, e o uso de forma probabilística desses operadores.

Duas novas características foram introduzidas nos AGs que os diferem dos demais algoritmos: a técnica de realizar a seleção de pais e o modo de representar os indivíduos.

A seleção de pais no AG realizada para gerar uma nova população passa a ser probabilística, e não mais determinística. A seleção probabilística dá mais chances às melhores soluções para participarem dos cruzamentos, mas não descarta a participação de soluções ruins, pois estas podem conter genes que contribuam para a evolução das soluções, além de favorecer a diversidade na população. A seleção determinística escolhe necessariamente as melhores soluções, extinguindo, com o passar das gerações, as soluções mal avaliadas e, assim, o algoritmo tende a convergir mais rapidamente, o que nem sempre é bom quando se deseja percorrer todo o espaço de busca por soluções. Outra característica introduzida pelo AG é a representação das soluções por uma sequência binária para manter uma analogia com a representação usada nos computadores digitais (ZEBULUM, 2001). Essa representação é mais natural para o processamento computacional do algoritmo.

Os AGs podem ser descritos pelo seu ciclo de execução, que se repete até que uma solução satisfatória seja obtida, conforme ilustrada na Figura 2.6.

As etapas denominadas “Seleciona Elite” e “Preserva Elite”, vistas na figura 2.6, são opcionais para a execução do AG e têm por objetivo manter a melhor solução de uma geração para a seguinte. Assim, não se corre o risco de perder uma boa solução ao longo da execução do algoritmo.

Os AGs são algoritmos altamente não-lineares, tornando difícil a previsão do seu comportamento quando seus parâmetros de implementação variam (método de seleção de pais, formas de cruzamento e mutação, uso de elitismo, tamanho da população, etc.) (ZEBULUM, 2001). O modelo matemático mais aceito para descrever o comportamento de um AG é o que foi concebido e nomeado por John Holland de Teoria dos Esquemas (HOLLAND, 1975). Este modelo não é capaz de explicar vários aspectos do comportamento complexo apresentado pelos AGs, mas ajuda a entender a maneira como os AGs funcionam (ZEBULUM, 2001).

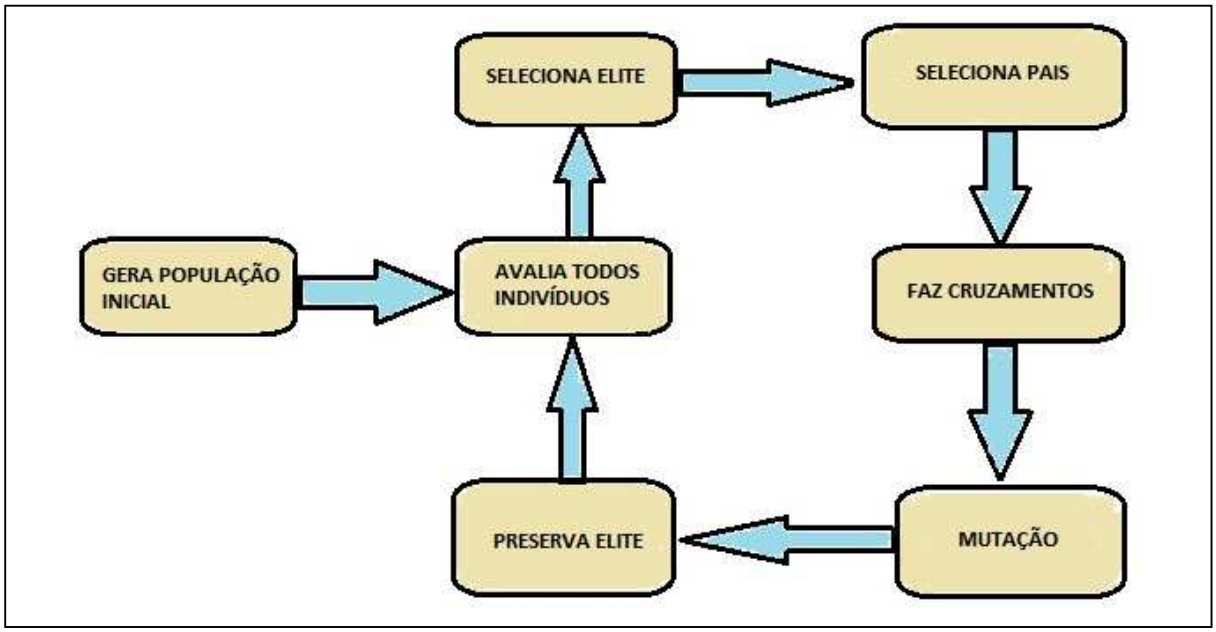


Figura 2.6 Ciclo de execução de um GA.

Um esquema é um padrão particular que descreve um conjunto de cromossomos do espaço de busca. Os cromossomos de um determinado padrão apresentam símbolos idênticos em algumas posições da representação usada. Para apresentar um esquema de representação binária, usa-se um símbolo extra “*”, significando “qualquer valor permitido”, além do “0” e do “1”. Assim, para que um determinado cromossomo pertença ao esquema dado ele deve apresentar símbolos idênticos ao esquema em todas as posições diferentes de “*”. Seguem alguns exemplos para melhor compreensão:

esquema = 0 1 1 * 1 * *

cromossomo1 = 0 1 1 0 1 1 0 (pertence ao esquema)

cromossomo2 = 0 1 1 1 1 0 1 (pertence ao esquema)

cromossomo3 = 0 1 0 1 1 0 1 (não pertence ao esquema)

O total de esquemas obtidos com uma representação cujo alfabeto possui K símbolos e de comprimento L será $(K+1)^L$ (acrescenta-se o símbolo extra "*" à cardinalidade da representação). Para um cromossomo binário: $(2+1)^L$. Exemplo: uma representação binária de 8 bits, terá $(2+1)^8$ igual a 6561 esquemas, apesar de se ter apenas 2^8 igual a 256 indivíduos distintos: o que mostra que cada indivíduo pertence a vários esquemas simultaneamente.

A partir dos conceitos apresentados, definem-se a ordem e o comprimento de um esquema. A ordem de um esquema X, O(X), é definida pela quantidade de símbolos presentes no esquema que sejam diferentes de "*". Assim, para os esquemas de representação binária abaixo, têm-se:

X_1 = 0 1 1 * 1 * * → $O(X_1) = 4$

X_2 = 1 * * * * * 0 → $O(X_2) = 2$

X_3 = 1 * * * * * * → $O(X_3) = 1$

O comprimento de um esquema X, ρ(X), é definido como a distância entre o primeiro e o último símbolo presentes no esquema que sejam diferentes de "*". Tomando os mesmos esquemas anteriores, têm-se:

X_1 = 0 1 1 * 1 * * → $\rho(X_1) = 4$

X_2 = 1 * * * * * 0 → $\rho(X_2) = 6$

X_3 = 1 * * * * * * → $\rho(X_3) = 0$

Com esses dois últimos conceitos, Holland apresentou o principal resultado da teoria dos esquemas, também chamado de Teorema Fundamental dos AGs (HOLLAND, 1975). Este teorema procura prover informações a respeito do destino de um determinado esquema presente na população de um AG a partir de uma geração específica, ou seja, se os indivíduos pertencentes a um determinado esquema tendem a se proliferar ou a se extinguir ao longo das gerações (ZEBULUM, 2001).

O resultado obtido por Holland é apresentado na equação (2.6) (HOLLAND, 1975):

$$m(X,t+1) \geq m(X,t) \cdot \frac{f(X)}{f_m}\left[1 - p_c \cdot \frac{\rho(X)}{L-1}\right] \cdot \left[1 - O(X) \cdot p_m\right] \quad (2.6)$$

$m(X,t)$: número de indivíduos pertencentes ao esquema X na geração t;

$f(X)$: avaliação média do esquema X;

f_m: avaliação média da população;

p_c: taxa de ocorrência de cruzamento;

p_m: taxa de ocorrência de mutação.

Conclui-se dos resultados de Holland (1975) que esquemas com pequeno comprimento e de pequena ordem têm mais chances de sobreviver às gerações. Outro fator que deve ser considerado para a estimativa da sobrevivência de um esquema ao longo das gerações é a relação entre a avaliação do esquema e a avaliação média da população. Holland mostrou que esquemas com avaliação acima da média tendem a ocorrer com mais frequência nas gerações seguintes e, por outro lado, esquemas com baixa avaliação tendem a desaparecer (LINDEN, 2012).

Pela equação (2.6), quanto maior a taxa de ocorrência do operador de cruzamento e maior o comprimento do esquema, menos chances desse esquema sobreviver nas gerações seguintes, pois esquemas longos possuem mais pontos de cortes, aumentando a probabilidade de sua destruição. De maneira semelhante, quanto maior a taxa de mutação e maior a ordem do esquema, menos chances de sobrevivência do esquema, pois esquemas com ordem elevada possuem mais genes para serem destruídos pela mutação, diminuindo suas chances de sobrevivência nas gerações seguintes.

A ação dos operadores genéticos foi chamada por Holland de tensão entre a exploração (a busca por novos indivíduos semelhantes aos que possuem boa avaliação com a esperança de que sejam ainda mais bem adaptados) e o aproveitamento (busca por soluções em diferentes regiões do espaço de busca). Os cruzamentos e as mutações são potencialmente destruidores de esquemas, mas o cruzamento atua como explorador local de novas soluções; ao passo que a mutação promove o aproveitamento global, estimulando a obtenção de soluções em qualquer região do espaço de busca, evitando que o algoritmo fique preso a um ótimo local (ZEBULUM, 2001).

A forma final do teorema dos esquemas pode ser apresentada da seguinte forma (LINDEN, 2012):

“O AG tende a preservar com o decorrer do tempo aqueles esquemas com maior avaliação média e com menores ordem e comprimento, combinando-os como blocos de forma a buscar a melhor solução”.

Os parâmetros importantes para a construção dos AGs são apresentados com mais detalhes no capítulo 3, que é dedicado exclusivamente a essa categoria de algoritmo evolucionário.

2.7.4 Programação Genética (PG)

O modelo da programação genética (PG) dá continuidade ao processo de tratar o problema de representação cromossômica em AG aumentando a complexidade das estruturas sujeitas à adaptação (KOZA, 1992).

No início dos anos de 1990, John Koza, da Universidade de Stanford, EUA, concebeu a ideia de se obter automaticamente programas computacionais utilizando-se a mesma estratégia aplicada aos AGs, com uma diferença fundamental: os cromossomos deixam de ser representados como uma sequência binária e passam a ser representados por uma estrutura em árvore (ZEBULUM, 2001).

A escolha dessa estrutura de dados para o cromossomo se deve ao fato de que a pesquisa de Koza tinha por objetivo gerar programas na linguagem LISP, cuja sintaxe pode ser facilmente descrita como uma estrutura em árvore, como no exemplo da Figura 2.7.

As árvores da PG são constituídas de dois tipos de nós: nós de função (vértices internos da árvore), que podem representar operações aritméticas (+. -, *, /, %, ^,...), funções matemáticas (seno, cosseno, logaritmo, raiz quadrada,...), funções lógicas (e, ou, não,...), operadores condicionais (se... então... senão...), operadores para iteração (repita... até... enquanto...) e qualquer outra função no domínio do problema específico, e os nós terminais (vértices externos, também conhecidos por folhas), que podem assumir o valor de variáveis do problema ou constantes.

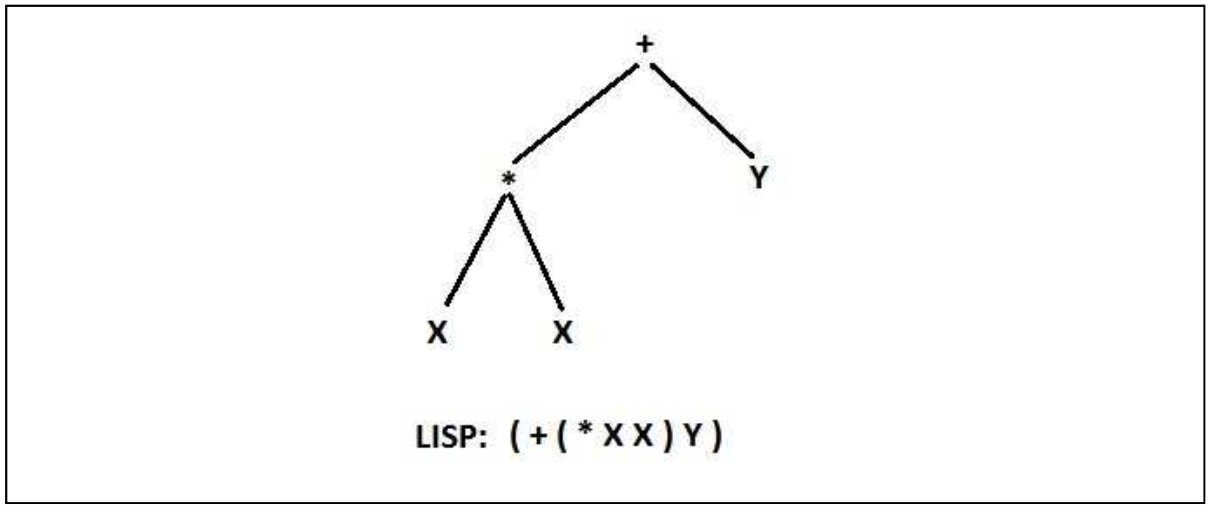


Figura 2.7 Exemplo de representação da expressão x^2 + y em árvore e em LISP, adaptado de (ZEBULUM, 2001).

O ciclo de execução da PG é basicamente o mesmo do AG, acrescentando-se as fases de definição do conjunto de terminais e do conjunto de funções, que serão usados para a geração das árvores representantes dos cromossomos dos programas gerados. A PG utiliza também o operador unário de reprodução, que consiste em escolher um indivíduo da geração atual com base no critério de avaliação, para reproduzi-lo na geração seguinte sem alteração do seu cromossomo. A Figura 2.8 apresenta o diagrama de fluxo de execução de uma PG (KOZA, 1992). Os termos P_r e P_c representam a taxa de ocorrência para aplicação dos operadores de reprodução e cruzamento, respectivamente, pois estes operadores genéticos são probabilísticos; e o valor M representa o tamanho da população.

Em resumo, a PG gera programas computacionais a partir de execução de 3 passos (KOZA, 1992):

(1) Gera uma população inicial, aleatória, de conjuntos de nós de função e nós terminais para a resolução do problema (programas candidatos);

(2) Por meio de iterações, executa os seguintes sub passos até encontrar um critério de parada:

   a. Executa cada programa na população e atribui-lhe uma nota, de acordo com o seu desempenho para resolver o problema proposto.
   b. Cria uma nova população de programas aplicando dois operadores genéticos primários. A escolha é baseada em probabilidade por avaliação, isto é, quanto maior a avaliação, mais chances de ser escolhido:
      i. Reprodução: copia o programa escolhido na população seguinte;

ii. Cruzamento: cria dois novos programas na população seguinte recombinando dois pais escolhidos;

(3) O melhor programa de cada geração é o candidato à solução (ou a uma solução aproximada) do problema proposto.

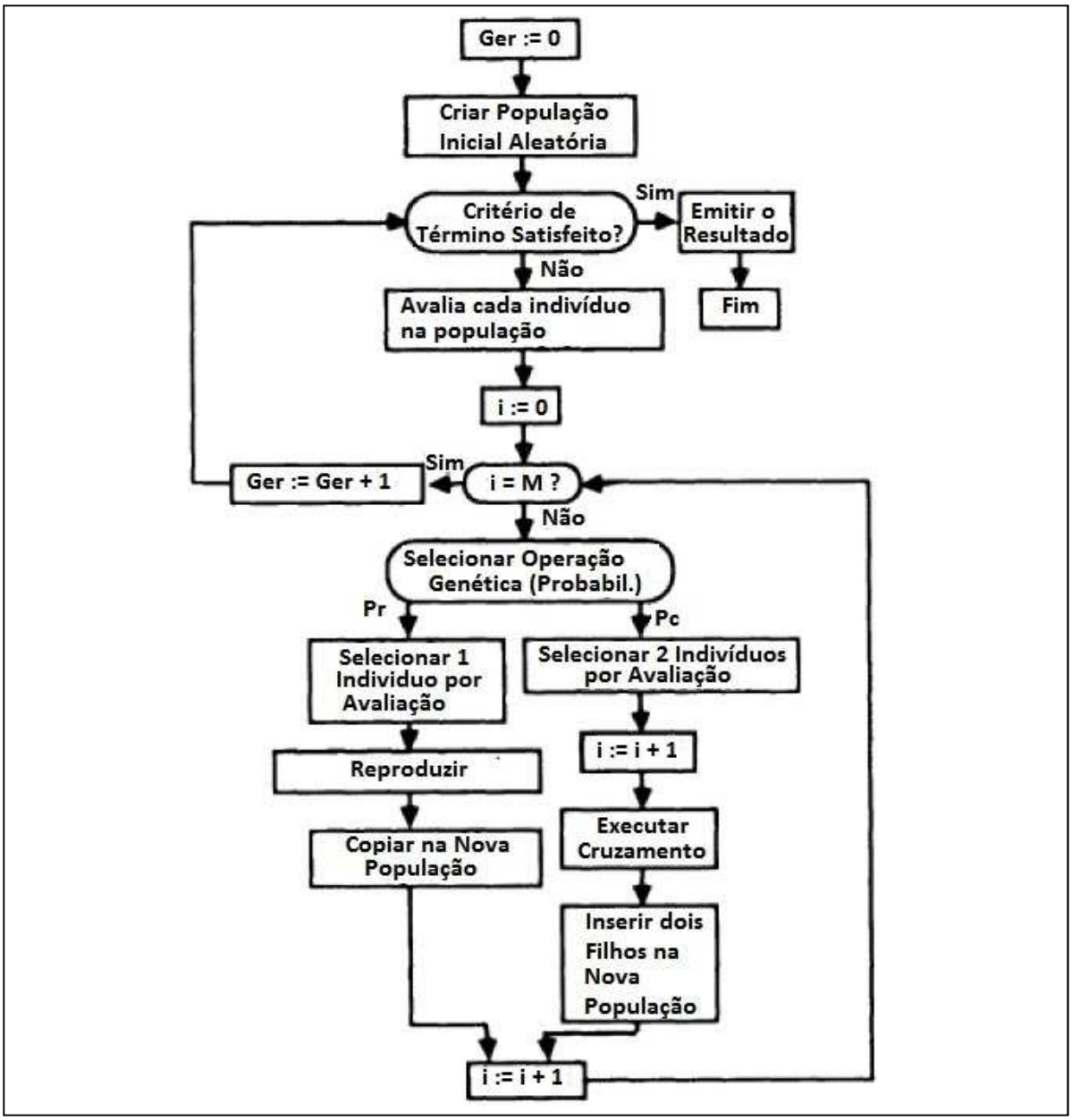


Figura 2.8 Diagrama de fluxo de execução da PG, traduzido livremente de (KOZA, 1992).

O cruzamento é realizado com o sorteio de um nó de cada pai selecionado. As sub árvores obtidas a partir de cada nó são intercambiadas entre os indivíduos gerando duas novas árvores, conforme o exemplo da Figura 2.9.

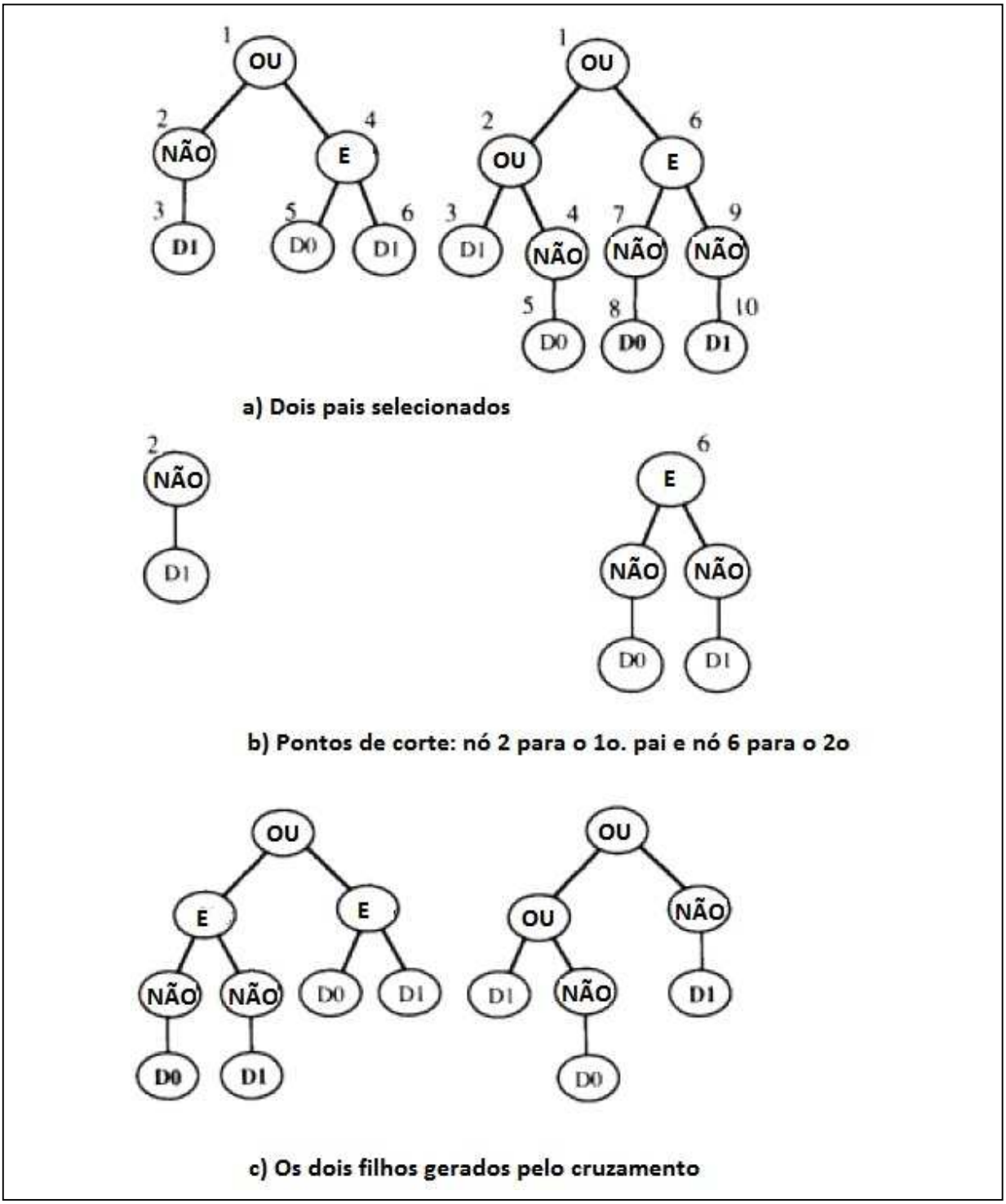


Figura 2.9 Processo de cruzamento representado em 3 etapas: a) escolha dos pais; b) sorteio dos pontos de corte em cada pai; c) resultado da recombinação: dois novos indivíduos, adaptado de (KOZA, 1992).

Ao ciclo de execução descrito anteriormente, pode ser incluído o operador de mutação com probabilidade P_m. Dessa forma, quando a mutação é selecionada como operador genético, um nó da árvore, N_m, deve ser sorteado para receber a mutação e uma sub árvore deve ser

gerada aleatoriamente para substituir a sub árvore original a partir de N_m, conforme o exemplo ilustrado na Figura 2.10.

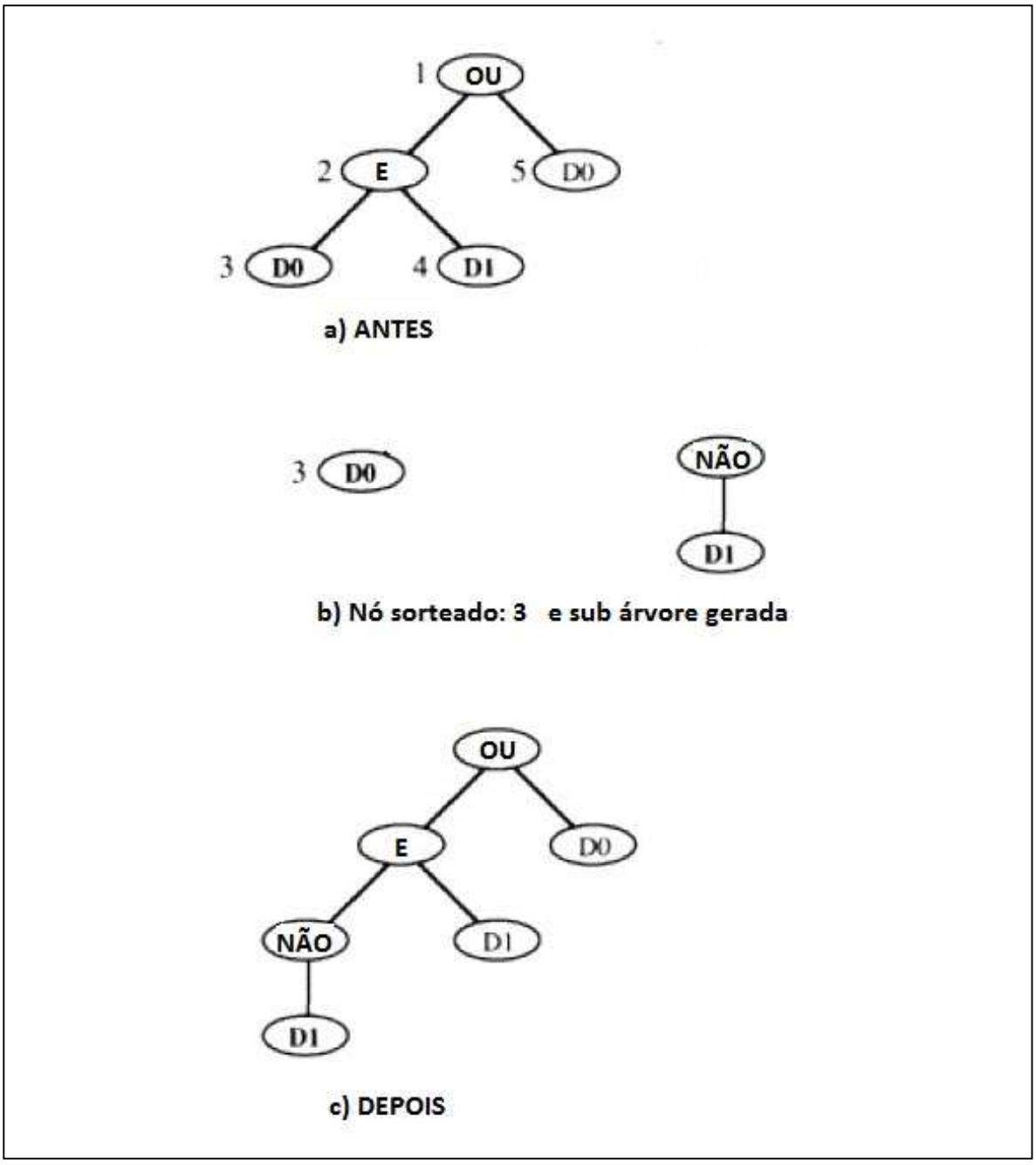


Figura 2.10 Processo de mutação para a PG (KOZA, 1992).

Apesar de muitas semelhanças no ciclo de execução de uma PG quando comparada ao AG, há algumas particularidades que tornam a PG diferente dos outros algoritmos evolucionários. O operador de cruzamento tem muito mais importância na PG, devido às características da representação do cromossomo em estrutura de árvore (ZEBULUM, 2001). No AG, dois pais idênticos geram filhos idênticos, independentemente dos pontos de corte

escolhidos. Na PG, diferentemente, dois pais idênticos dificilmente vão gerar filhos idênticos, pois para que isso ocorra os nós sorteados em cada pai devem ser os mesmos, o que é pouco provável, levando em conta que a árvore pode conter um número grande de nós. Assim o cruzamento da PG gera muito mais diversidade na população em comparação com o AG.

Apesar dos avanços introduzidos na PG como um algoritmo evolucionário com maior capacidade de exploração do espaço de busca, seu campo de aplicação é mais voltado aos problemas de desenvolvimento automático de programas computacionais. Assim, a programação genética não foi empregada no objeto de estudo deste trabalho, que são os circuitos elétricos analógicos gerados automaticamente.

3. ALGORITMO GENÉTICO ORIENTADO A OBJETOS

3.1 Introdução

Este capítulo é dedicado aos detalhes de desenvolvimento do ciclo de execução e dos operadores genéticos dos AGs. Os parâmetros utilizados na execução do algoritmo são apresentados e discutidos quanto à sua importância e seus efeitos nos resultados obtidos. Como é abordado adiante, o AG é altamente genérico, pois muitos de seus componentes são independentes de particularidades do problema e, portanto, podem ser aplicados em diversas áreas do conhecimento. Essa característica favorece o desenvolvimento em linguagem orientada a objeto, permitindo o reaproveitamento do código para vários problemas diferentes (LINDEN, 2012).

Todos os componentes apresentados no capítulo anterior, sobre os algoritmos evolucionários, são descritos agora como classes, que representam a natureza dos elementos pertencentes à modelagem do problema a ser tratado. As classes apresentadas podem ter suas versões mais específicas, chamadas de subclasses, ou classes filhas. Estas podem conter novas funções ou adaptar as funções da classe mãe a um contexto específico e podem ainda conter novos dados que as caracterizam.

Com todas as classes definidas, deve-se montar o diagrama de relacionamento entre elas a partir do entendimento dos requisitos do algoritmo. Esse diagrama, chamado de Diagrama de Classes, é a base para o desenvolvimento do ciclo de execução do AG, como apresentado adiante. Outro diagrama importante é o Diagrama de Sequência, que apresenta o comportamento dos objetos, que são instâncias de cada classe, na forma de uma sequência de mensagens enviadas de um objeto para outro, de forma a representar a execução do ciclo evolutivo do algoritmo. As mensagens enviadas pelos objetos deste diagrama correspondem aos métodos implementados nas classes (BOOCH et al, 1999).

Nas seções seguintes são descritas as classes que fazem parte da proposta deste trabalho, e os diagramas de classes e de sequência, como uma forma de desenvolver o Algoritmo Genético Orientado a Objetos (AGOO).

3.2 Classe População

É responsável por controlar o conjunto de indivíduos do AG e é considerado, por questão de simplicidade de desenvolvimento, que a população é constante, ou seja, os filhos

gerados substituem seus pais para que não haja aumento de indivíduos de uma geração para a outra. Essa estratégia de controle corresponde à (μ,λ)EE, em que todos os λ filhos, gerados a partir de uma população de μ indivíduos, compõem a geração seguinte, fazendo com que o ciclo de vida de qualquer indivíduo seja de apenas uma geração. Como a quantidade de indivíduos da população é um parâmetro relevante para o desempenho do AG, o tamanho da população será um dos seus atributos e, sabendo-se que um dos critérios de parada do algoritmo é o número de iterações realizadas, outro atributo necessário para a esta classe será o contador de gerações, armazenado no atributo geração.

Outros atributos desta classe são:

soma: mantém a soma das avaliações de todos os indivíduos da população, que é um parâmetro para classificar a adequação de cada indivíduo;

melhor e pior: indica qual é o melhor e qual o pior indivíduo da população, para fins de política de elitismo, ou seja, no início do ciclo obtém-se o "melhor" para mantê-lo salvo; no fim do ciclo, obtém-se o "pior" para ser substituído pelo "melhor", salvo previamente;

escolhaPai e escolhaMae: atributos usados para receber o resultado da seleção de pais em cada operação genética de cruzamento no ciclo de execução do algoritmo;

taxaCruzamento: índice de probabilidade de ocorrência do cruzamento genético, de 0 a 1. Quanto maior, mais cruzamentos são realizados em cada ciclo do AG. Quando o cruzamento não é realizado entre dois pais selecionados, os filhos gerados são idênticos aos pais.

taxaMutacao: índice de probabilidade de ocorrência da mutação genética, de 0 a 1. Quanto maior, mais frequentemente as mutações são aplicadas aos indivíduos da população, gerando maior exploração do espaço de busca, porém tornando o algoritmo mais próximo da busca aleatória simples.

Os métodos desta classe são os que desempenham funções relacionadas com:

A criação de uma população inicial aleatória (IniciarPopulacao);

A seleção de pais para o cruzamento (selecionarPais);

A avaliação de todos os indivíduos (avaliarTodos);

A aplicação dos operadores genéticos (novaPopulação);

Adoção da política de elitismo (gerarElite).

Alguns métodos implementados nesta classe fazem uso da relação mantida com outras classes, por exemplo, o método selecionarPais deve fazer chamadas aos métodos

implementados na classe Seleção, como veremos adiante, pois a classe Seleção é a responsável pela tarefa de executar os procedimentos que selecionam os indivíduos que serão submetidos aos operadores genéticos. O método avaliarTodos deve percorrer todos os indivíduos da população e solicitar a avaliação de cada um. Como se vê a seguir, cada indivíduo é modelado pela classe Cromossomo, que por sua vez, se relaciona com a classe Avaliação, responsável por calcular a sua nota.

O método novaPopulação controla a execução dos cruzamentos e mutações e o método gerarElite implementa a política de elitismo do algoritmo, visando preservar a melhor solução de cada geração.

A representação da classe População, com indicação do nome, atributos e métodos no retângulo dividido da UML (*Unified Modeling Language*) (BOOCH et al, 1999), é mostrada na Figura 3.1.

3.3 Classe Cromossomo

Todas as informações referentes à representação da solução do problema devem ser mantidas na classe Cromossomo. Esta classe é responsável também pela execução das operações genéticas, ou seja, o cruzamento e a mutação, e seus atributos devem corresponder aos dados contidos no cromossomo da solução.

O atributo *genes* deve conter a sequência binária do cromossomo e pode ser declarado como do tipo inteiro, desde que a quantidade de bits necessária para a representação cromossômica seja menor que a quantidade de bits usada para a representação de inteiros da linguagem de programação escolhida. Em Java™, por exemplo, a variável do tipo inteira possui 32 bits (faixa de valores: de -2.147.483.648 a 2.147.483.647). Se desconsiderar-se o bit mais significativo da representação, que é usado para indicar o sinal negativo, a representação passa a 31 bits de valores apenas não negativos, com faixa de 0 a 2.147.483.647. Se for necessário ou conveniente, a representação cromossômica pode ser quebrada em vários atributos do tipo inteiro, mesmo que haja alocação de bits além do necessário para cada atributo. Esse procedimento pode dar mais clareza ao código desenvolvido, facilitando a compreensão do programa, uma vez que cada atributo representaria um segmento do cromossomo, o que corresponde a um parâmetro do indivíduo representado. Assim se teria gene1, gene2, gene3, etc., todos declarados como inteiro e cada um representando uma característica do indivíduo. Na Figura 3.2 se vê a representação UML desta classe.

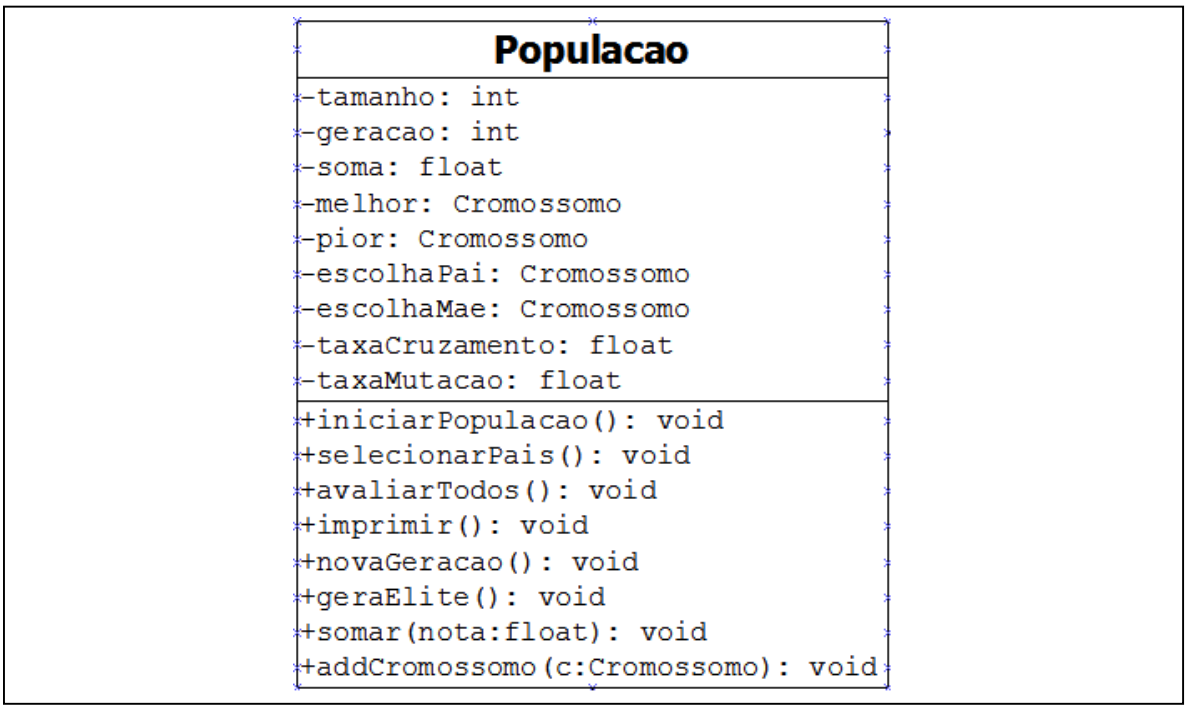


Figura 3.1 Representação da classe População usando a convenção da UML.

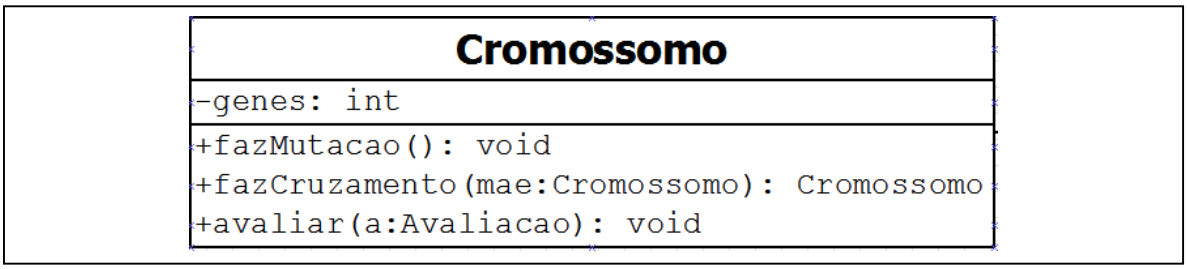


Figura 3.2 Representação da classe Cromossomo usando a convenção da UML.

3.4 Classe Seleção

O método *selecionar* é o único desta classe, que contém subclasses com a reescrita deste método. Assim, cada subclasse executa a sua técnica de seleção de pais para o cruzamento, conforme a representação UML na Figura 3.3.

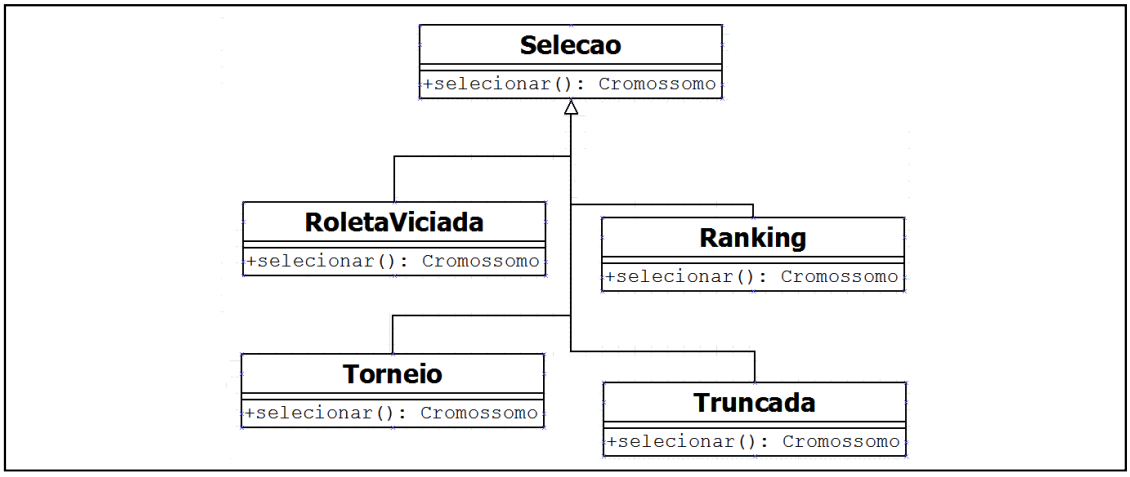


Figura 3.3 Classe Seleção e suas subclasses com reescritas do método *selecionar*.

A classe Seleção propriamente dita pode ter seu método *selecionar* vazio, sem código, o que caracteriza um método abstrato, cuja única função é permitir reescritas nas classes filhas (subclasses). A declaração de método abstrato é explícita em Java™, com o uso da diretiva *abstract* no início da declaração do método.

As classes filhas propostas neste trabalho, e suas técnicas de seleção, são:

3.4.1 Seleção por Roleta Viciada

Esta técnica de seleção executa o processo de sorteio com uma roleta cuja porcentagem de ocupação de cada indivíduo na roda é proporcional à sua avaliação, conforme exemplificado na Figura 3.4, com os dados da Tabela I, para uma população de 10 indivíduos:

A execução do método de seleção pela Roleta Viciada não exige que os indivíduos sejam previamente ordenados pelo valor das avaliações, basta seguir as etapas do algoritmo a seguir (LINDEN, 2012):

1. Sorteia-se um número real $\boldsymbol{s}$, $0 \leq \boldsymbol{s} < T$, onde T é o valor da soma das notas de todos os indivíduos da população. No exemplo, T=486;
2. $\boldsymbol{k}$=1, índice para o primeiro indivíduo da população;
3. $\boldsymbol{p}$=0, valor inicial do passo de execução;
4. <Repita>

   $p = p + n_k$, (n_k é a nota do k-ésimo indivíduo da população)

   Se $p > s$, o indivíduo k é o escolhido e fim da execução.

   Caso contrário, $k = k+1$;

   <Fim-Repita>

Tabela I Distribuição de notas entre 10 indivíduos para a Roleta Viciada.

Indivíduo	Nota	Fração da roleta	Setor circular (°)
1	83	83/486=17,1%	61,5
2	91	91/486=18,7%	67,3
3	55	55/486=11,3%	40,7
4	13	13/486=02,7%	9,7
5	45	45/486=09,3%	33,5
6	69	69/486=14,2%	51,1
7	25	25/486=05,1%	18,4
8	45	45/486=09,3%	33,5
9	08	08/486=01,6%	5,8
10	52	52/486=10,7%	38,5
Total	486	100%	360

A Tabela I apresenta, como exemplo, uma população com dez indivíduos identificados de 1 a 10, com suas respectivas notas de avaliação (as notas que constam na tabela são apenas exemplos numéricos para facilitar a descrição deste método de seleção de pais com o uso de um caso prático). A coluna “Fração da roleta” contém a fração da roleta, em porcentagem, que cada indivíduo deve ocupar para o sorteio. Numericamente o valor desta fração é a nota do indivíduo dividida pela soma de todas as notas da população. A última coluna contém os respectivos setores circulares da roleta (em graus) ocupados pelos indivíduos, considerando que a soma das notas de toda a população ocupa os 360° da roleta, como visto na Figura 3.4.

Analisando o algoritmo descrito para o exemplo da Tabela I, observa-se que o valor do passo ***p***, que foi iniciado com zero, no primeiro ciclo é somado com o valor da nota do indivíduo 1. Se o valor ***s*** sorteado for menor ***p***, o indivíduo 1 é o escolhido; caso contrário, soma-se a ***p*** a nota do segundo indivíduo e repete-se a comparação de ***s*** com ***p***: se $s < p$, o indivíduo 2 é o escolhido; caso contrário, soma-se a ***p*** a nota do terceiro indivíduo e repete-se a comparação de ***s*** com ***p***: se $s < p$, o indivíduo 3 é o escolhido, e assim por diante. Se o valor ***p*** chegar a ser somado com a nota do último indivíduo da população, este último será o escolhido. Os indivíduos com maior faixa de valores para comparar com ***s*** têm mais chances de serem escolhidos, independentemente da ordem em que estiverem dispostos, pois, para a seleção do indivíduo 1, por exemplo, a probabilidade de se escolher um valor de ***s*** entre 0 e 83 é mesma de se escolher ***s*** entre 403 e 486, caso em que o indivíduo 1 estaria na última posição dentro da população. O raciocínio vale para todos os outros indivíduos. Em outras palavras, a

ordem da disposição dos setores coloridos na roda da Figura 3.4 não altera a probabilidade de escolha de cada uma das cores; portanto não há necessidade de se executar qualquer ordenação dos indivíduos.

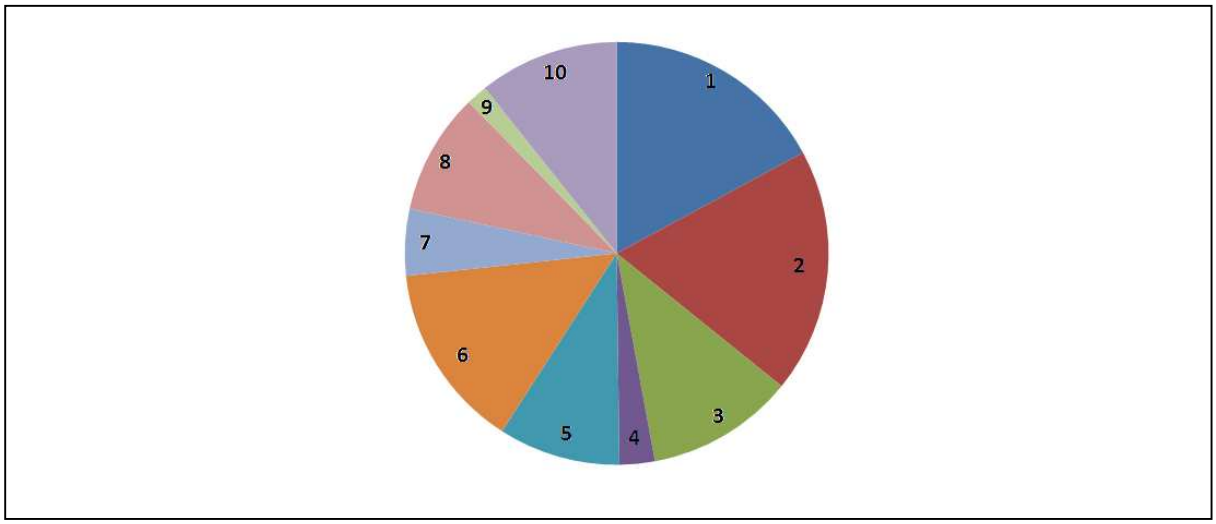


Figura 3.4 Representação gráfica do método de seleção por Roleta Viciada com a distribuição dada pela Tabela I.

A Roleta Viciada oferece maior probabilidade de escolha para os indivíduos mais bem avaliados, assim, os melhores tendem a participar mais dos cruzamentos e transmitir suas boas características para as próximas gerações. Essa característica do algoritmo de seleção para impelir os esquemas contidos nas melhores soluções para a próxima geração é chamada de pressão seletiva (LINDEN, 2012), que é um fator positivo para a seleção, pois favorece o aspecto de aproveitamento do AG. Há, porém, uma desvantagem presente no método da roleta viciada quando há um super indivíduo na população, ou seja, um elemento com nota muito superior às notas dos demais, fazendo com que a faixa deste indivíduo ocupe quase todo o espaço da roleta, causando a escolha do super indivíduo em praticamente todas as suas execuções. Em poucas gerações, não haverá mais diversidade na população, pois quase todos os indivíduos serão iguais. A população só não será completamente homogênea por conta do operador de mutação, que deve ser aplicado a uma pequena parcela da população.

3.4.2 Seleção por Torneio

Para minimizar o efeito da pressão seletiva causada por um eventual super indivíduo da população ou por qualquer outro indivíduo bem avaliado, há variações da técnica de seleção que restringem a importância da avaliação na escolha dos pais. Uma dessas variações é o método do Torneio, que consiste em escolher duplas de indivíduos, escolhidos

aleatoriamente dentre todos na população independentemente de suas avaliações, e só então determinar como vencedor o indivíduo com maior avaliação. Assim, a avaliação é determinante apenas para definir o vencedor de cada confronto e não para pressionar a escolha de cada indivíduo.

A execução do algoritmo do Torneio para escolha de dois pais é muito simples e é descrito nas três etapas a seguir (LINDEN, 2012):

1. Escolhem-se quatro indivíduos da população aleatoriamente para os confrontos: I_1, I_2, I_3 e I_4, cujas notas de avaliação são n_1, n_2, n_3 e n_4, respectivamente;
2. Se $n_1 > n_2$. I_1 é escolhido como o primeiro pai
   Caso contrário, I_2 é escolhido como o primeiro pai
3. Se $n_3 > n_4$. I_3 é escolhido como o segundo pai
   Caso contrário, I_4 é escolhido como o segundo pai

A título de exemplo, tome-se a população listada na Tabela I para aplicar o método do torneio. Supondo que os quatro indivíduos escolhidos aleatoriamente sejam, na sequência, os elementos 3, 6, 4, e 9. A Figura 3.5 apresenta um esquema da execução da seleção pelo Torneio com os esses quatro indivíduos sorteados, em que, devido ao arranjo dos confrontos, os elementos 6 e 4 são os selecionados pelo método.

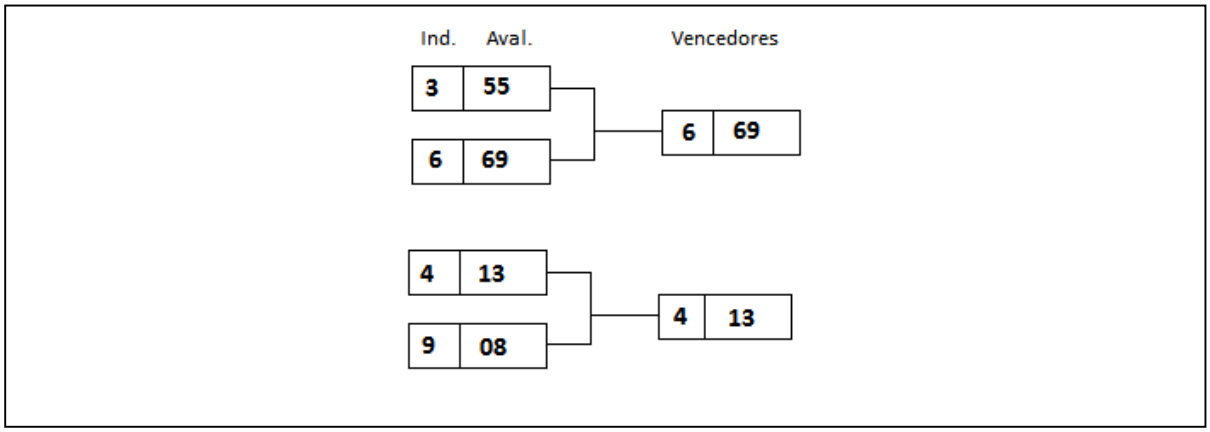


Figura 3.5 Torneio realizado com os indivíduos 3,6,4 e 9 da Tabela I. Os selecionados são os indivíduos 6 e 4.

Há um parâmetro para este método que é o tamanho do torneio (K), ou seja, a quantidade de indivíduos que devem participar dos confrontos. O valor mínimo para K é 2, caso contrário, não haveria competição para a escolha. O valor máximo não tem limite teórico, mas para não haver repetição de indivíduos na escolha, o K máximo deve ser o tamanho da

população. Neste caso, haveria o inconveniente de que o melhor indivíduo da população sempre seria selecionado (LINDEN, 2012). Cada confronto, é claro, pode haver mais de dois competidores, porém sempre haverá apenas dois confrontos se o objetivo for selecionar dois pais. Serão K/2 indivíduos em cada confronto, portanto é desejável que o valor de K seja par, de modo a manter o arranjo de confrontos simétrico. O algoritmo pode ser adaptado para 3 ou mais confrontos, se o número de pais que participam do cruzamento for maior que dois. Todas essas possibilidades fazem parte das opções disponíveis para o desenvolvedor do AG para melhor adaptar o algoritmo ao problema a ser tratado.

3.4.3 Seleção por Ranking

Outra opção para minimizar o efeito da pressão seletiva, causada pelos indivíduos com avaliação muito alta é o uso da seleção por Ranking, proposta por Baker (1985).

A ideia é ordenar os indivíduos segundo suas notas de avaliação e, então, as chances de cada indivíduo ser selecionado será função do peso de sua posição no ranking e não do valor absoluto da sua avaliação.

Para definir o novo valor de peso de cada indivíduo *i* na geração *t*, *P(i,t)*, usa-se um método linear proposto por Mitchel (1996):

$$P(i,t) = \min + (\max - \min) \cdot \frac{rank(i,t) - 1}{N - 1} \tag{3.1}$$

Sendo:

min = valor da avaliação atribuído ao indivíduo da última colocação no ranking;

max = valor da avaliação atribuído ao indivíduo da primeira colocação no ranking;

N = número de indivíduos na população;

rank(i,t) = pontuação do indivíduo *i* no ranking da geração *t (de 1, para o pior, até N, para o melhor).*

Para a escolha dos valores “max” e “min” de (3.1) , Baker (1985) propôs uma relação linear entre o tamanho da população e os pesos dos indivíduos, de maneira que a área abaixo dessa reta seja numericamente igual ao tamanho da população. Os extremos da reta proposta seriam o zero e o peso máximo permitido a um indivíduo da população, cujo valor é 2, conforme calculado por (3.2), a partir da representação gráfica na Figura 3.6.

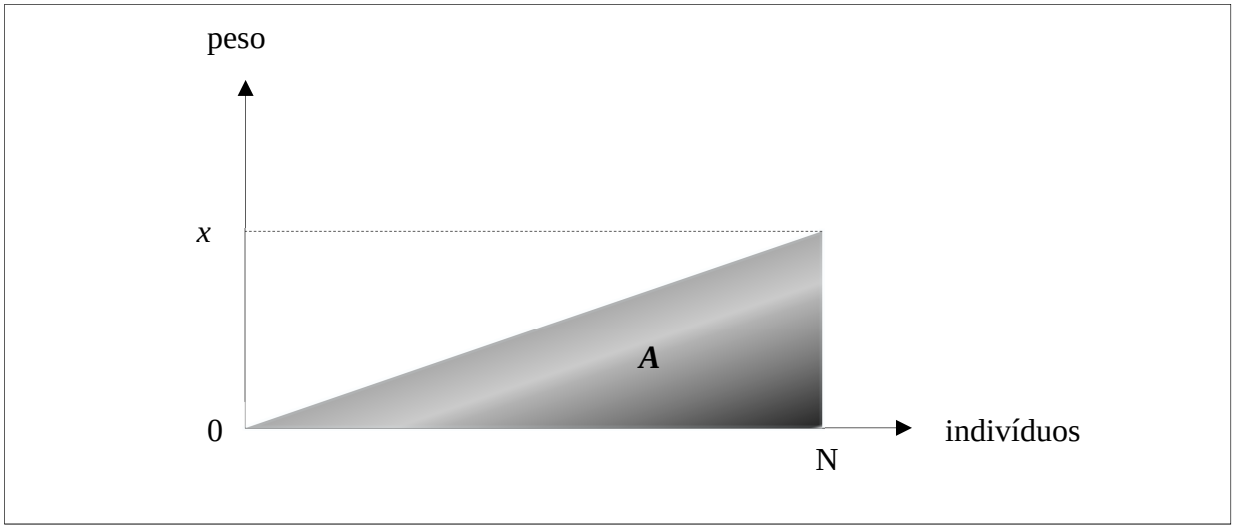


Figura 3.6 Distribuição linear de pesos para a seleção por Ranking.

$$A_{\Delta} \equiv N \Rightarrow \frac{N \cdot x}{2} = N \Rightarrow x = 2 \tag{3.2}$$

Baker (1985) apresentou também a definição de “percentual de envolvimento” da população, que corresponde à fração da população que efetivamente é selecionada para o cruzamento ao longo das gerações. Segundo Baker, o ideal é que esse percentual de envolvimento esteja na faixa de 94% a 100%, o que significa que quase todos os indivíduos da população são de fato selecionados.

Experimentalmente, fazendo max=2 e min=0, como representado na Figura 3.6, o percentual de envolvimento é de 75%, portanto 25% da população tendem a não serem selecionados, o que não é desejável para a manutenção da diversidade (BAKER, 1985). Observe que se max=1 e min=1 (reta horizontal) todos os indivíduos teriam peso 1, ou seja, teriam as mesmas chances de serem selecionados, garantindo à população obviamente 100% de percentual de envolvimento, porém não haveria pressão seletiva a favor dos indivíduos com melhor avaliação.

Finalmente, Baker (1985) afirma, baseado em seus experimentos, que os valores de “max” e “min” devem ser de aproximadamente 1,1 e 0,9, respectivamente, para garantir que o percentual de envolvimento da população atinja a faixa desejada; mantendo, ainda, uma pressão seletiva sobre os melhores indivíduos.

Aplicando a equação (3.1) aos indivíduos da Tabela I previamente ordenados, adotando max=1,1 e min=0,9, obtêm-se os pesos apresentados na Tabela II e representados graficamente na Figura 3.7:

Tabela II. Pesos dos indivíduos da população após o ranking.

ordenação	*Nota*	*rank(i,t)*	*Peso*	*Setor circular (°)*
2	91	10	1,10	40
1	83	9	1,08	39
6	69	8	1,06	38
3	55	7	1,03	37
10	52	6	1,01	36,5
5	45	5	0,99	35,5
8	45	4	0,97	35
7	25	3	0,94	34
4	13	2	0,92	33
9	08	1	0,90	32
TOTAL	486	-	10	360

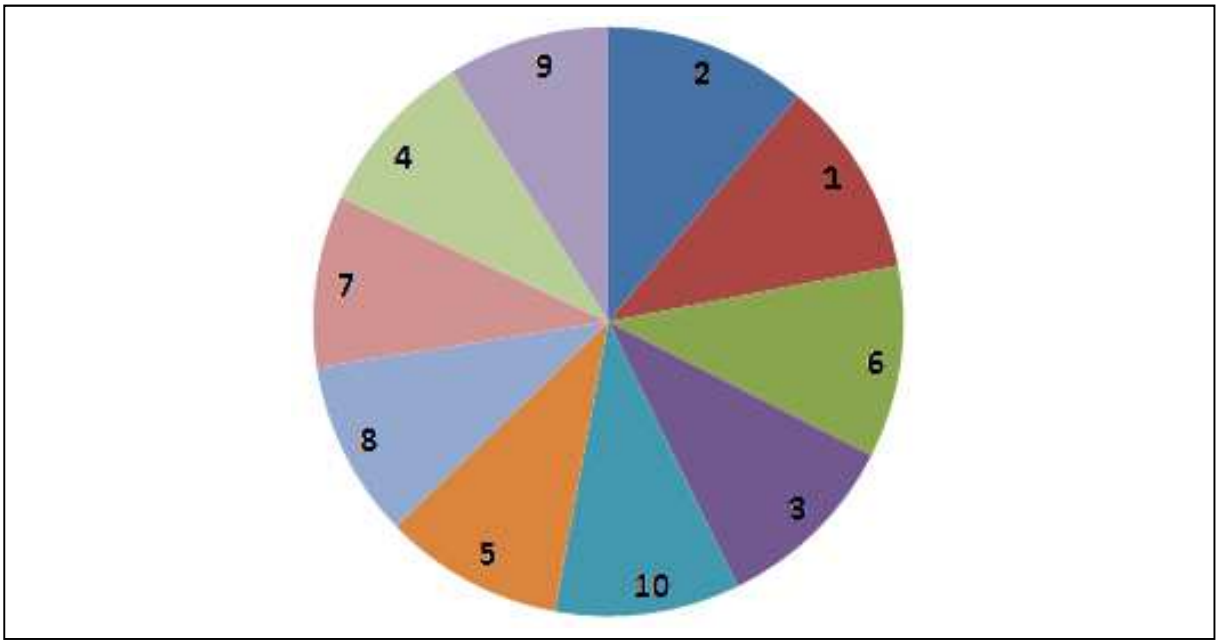


Figura 3.7 Representação gráfica do método de seleção por Ranking com a distribuição de pesos dada pela Tabela II.

A partir da distribuição de pesos aos indivíduos ordenados pelo *ranking*, pode-se aplicar o mesmo algoritmo da Roleta Viciada para fazer a seleção de pais, porém, desta vez, a pressão seletiva pelo melhor indivíduo será bem menor devido ao maior equilíbrio na formação da roleta. Devem-se fazer algumas adaptações no algoritmo da roleta: em vez de se utilizar a soma das avaliações como limite para o sorteio de *s*, usa-se 360, que é o valor em graus da volta completa da roleta; e, em vez de comparar *s* com as avaliações de cada indivíduo, deve-se compará-lo com o setor circular, em graus, ocupado pelos indivíduos na roleta, conforme indicado na última coluna da Tabela II.

3.4.4 Seleção Truncada

A seleção truncada tem o efeito inverso da seleção por Torneio ou seleção por *Ranking*, uma vez que a pressão seletiva se torna maior neste método do que em qualquer um dos anteriores.

A estratégia desta seleção é restringir a seleção a uma parte da população: os *n* melhores indivíduos. Assim, é preciso ordenar toda a população e aplicar um corte no *n*-ésimo indivíduo. A seleção é realizada apenas entre os *n* indivíduos previamente classificados para competir. Os valores empregados para *n* mais usuais correspondem à faixa de 10% a 50% da população e deve ser escolhido empiricamente levando em conta que quanto menor o valor de *n*, maior a perda de diversidade na população; por outro lado, *n* correspondente a 100% faz com que a seleção, é claro, deixe de ser truncada (LINDEN, 2012). A partir da segregação dos *n* melhores, aplica-se qualquer um dos métodos anteriores.

Segundo Blickle (1997), este método é o que causa maior perda de diversidade na população, pois o AG passará a se concentrar apenas nos melhores indivíduos, o que pode ser uma vantagem devido à possibilidade de se encontrar boas soluções mais rapidamente. A desvantagem é que com a pressão seletiva maior, a busca se torna mais restrita nas proximidades da melhor solução de cada geração, prejudicando a exploração de outras regiões do espaço de busca para as soluções.

A seleção truncada deve sempre ser usada em conjunto com outro método de seleção e nada impede que durante a execução do AG, vários métodos sejam aplicados. Por exemplo, no início, quando há maior diversidade na população, pode ser usada a seleção truncada com o torneio; depois de alcançado um determinado número de gerações, pode-se mudar para o método de seleção por *ranking* para favorecer a preservação da diversidade obtida até o momento. A variação dos métodos de seleção visa aproveitar as vantagens de cada um de

acordo com as mudanças de característica da população ao longo da execução do AG (LINDEN, 2012).

3.5 Classe Avaliação

A classe Avaliação, representada na Figura 3.8, é dependente do problema a ser tratado e deve representar uma estratégia para mensurar a qualidade dos cromossomos da população como uma solução procurada, portanto não há como generalizar a forma de construir esta classe para qualquer tipo de problema.

O método “calcular” deve ser desenvolvido com base no conhecimento sobre a solução que o AG deve encontrar, e incorporar tanto as restrições quanto os objetivos de qualidade, atribuindo uma nota a cada cromossomo. Como grande parte do AG pode ser desenvolvida de forma generalizada, a função de avaliação, em muitos casos, é a única ligação do programa com o problema real (LINDEN, 2012).

Como regra geral, a função de avaliação deve impor punições a informações contidas no cromossomo que o afasta das boas soluções e premiar as características que o aproxima, fazendo com que a nota final seja a mais gradual possível, evitando avaliações que simplesmente resultem tudo ou nada, boa ou ruim, 0 ou 1. Quanto maior o gradualismo da avaliação, maior a colaboração dessa classe para que o AG produza mais diversidade na população, pois cromossomos com notas diferentes devem necessariamente representar soluções diferentes, embora dois cromossomos que tenham notas iguais nem sempre representam a mesma solução, visto que estes podem representar soluções equivalentes, ainda que distintas.

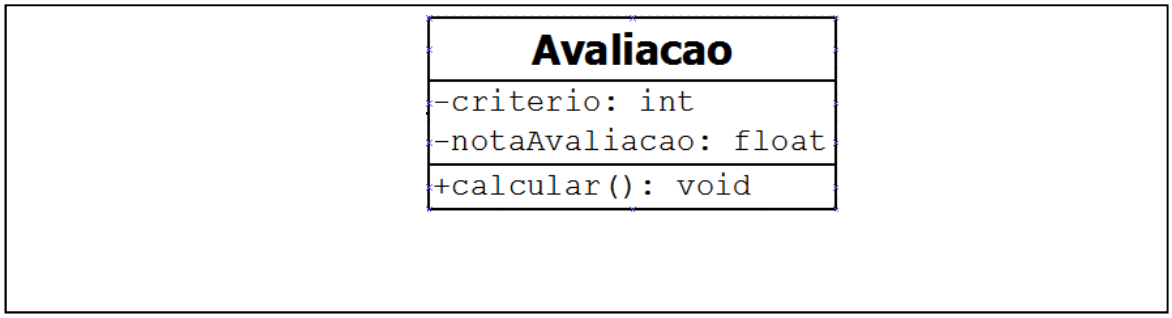


Figura 3.8 Classe Avaliação, responsável pelo cálculo da nota do cromossomo.

É desejável que a nota de avaliação tenha um valor máximo, para que haja um limite no somatório de notas da população, pois este total pode ser usado como limitador do *loop* de execução em alguns algoritmos de seleção de pais, submetidos aos operadores genéticos.

Uma forma de garantir a nota máxima é comparar a nota calculada com a nota teórica ideal, calculando de acordo com a equação (3.3). A forma de condicionar o valor da avaliação é uma adaptação de Murakawa (1998):

$$A_i = \frac{Nota_{máx}}{1 + \sum_j w_j \left| O(c_i, r_j) - I(c_i, r_j) \right|} \qquad (w_j > 0) \qquad (3.3)$$

Onde:

i:	índice do cromossomo na população ($1 \leq i \leq$ tamanho da população)
A_i :	nota condicionada de avaliação do cromossomo i;
$Nota_{máx}$:	nota máxima desejada para a avaliação;
j:	índice do critério de avaliação ($1 \leq j \leq$ total de critérios)
$O(c_i,r_j)$:	avaliação calculada do cromossomo c_i para o critério r_j;
$I(c_i,r_j)$:	avaliação ideal do cromossomo c_i para o critério r_j;
w_j:	peso do critério r_j.

Portanto, $0 < A_i \leq Nota_{máx}$

3.6 Relacionamento entre as classes e diagramas UML

O diagrama de classes apresentado na Figura 3.9 é um dos diagramas definidos na UML para a modelagem do problema a ser resolvido e tem por finalidade representar todas as classes definidas para o desenvolvimento do projeto com orientação a objetos, que auxilia na tarefa de escrever o código de cada classe, além de indicar a relação estabelecida entre essas classes, que auxilia na construção das dependências entre elas no código do programa.

A classe População é formada por vários representantes da classe Cromossomo (formando uma população de cromossomos) e interage com a classe Seleção para realizar a seleção de um pai para cruzamento. A classe População envia para a classe Seleção tantas mensagens quantos forem os pais necessários para a composição da nova geração. A classe Seleção possui várias subclasses, RoletaViciada, Torneio, *Ranking*, e Truncada, que, como já visto,

implementam um critério específico para sortear um elemento da população para participar das operações genéticas.

A classe Cromossomo, que é responsável pela execução das operações genéticas, cruzamento e mutação, mantém um relacionamento com a classe Avaliação para que esta realize o cálculo da nota de um cromossomo. Cada cromossomo tem a sua nota e cada nota calculada pertence a um único cromossomo.

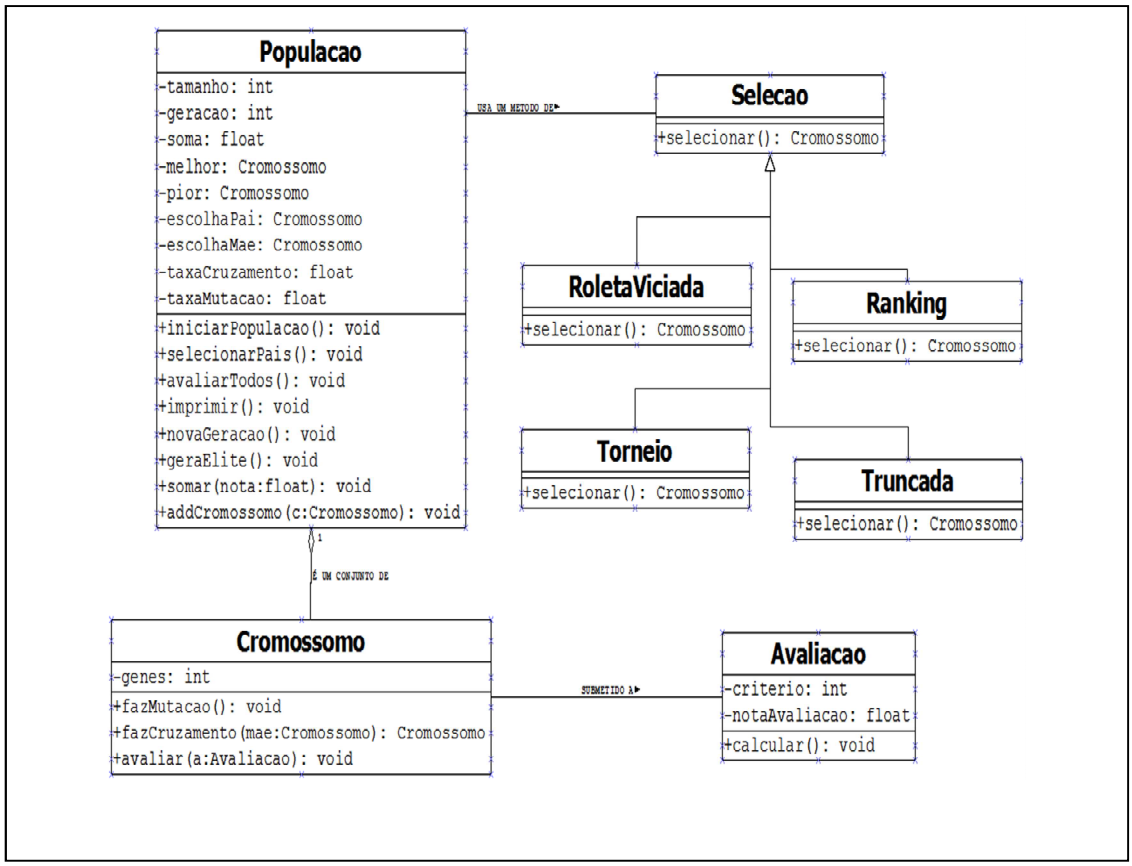


Figura 3.9 Diagrama de Classes – apresenta o relacionamento entre as classes do AG.

O diagrama da Figura 3.10, por sua vez, descreve a sequência de mensagens, ou chamadas de métodos, que um objeto faz a outro, de maneira que fique claro como a execução do AG deve ser construída no código, qualquer que seja a linguagem de programação escolhida.

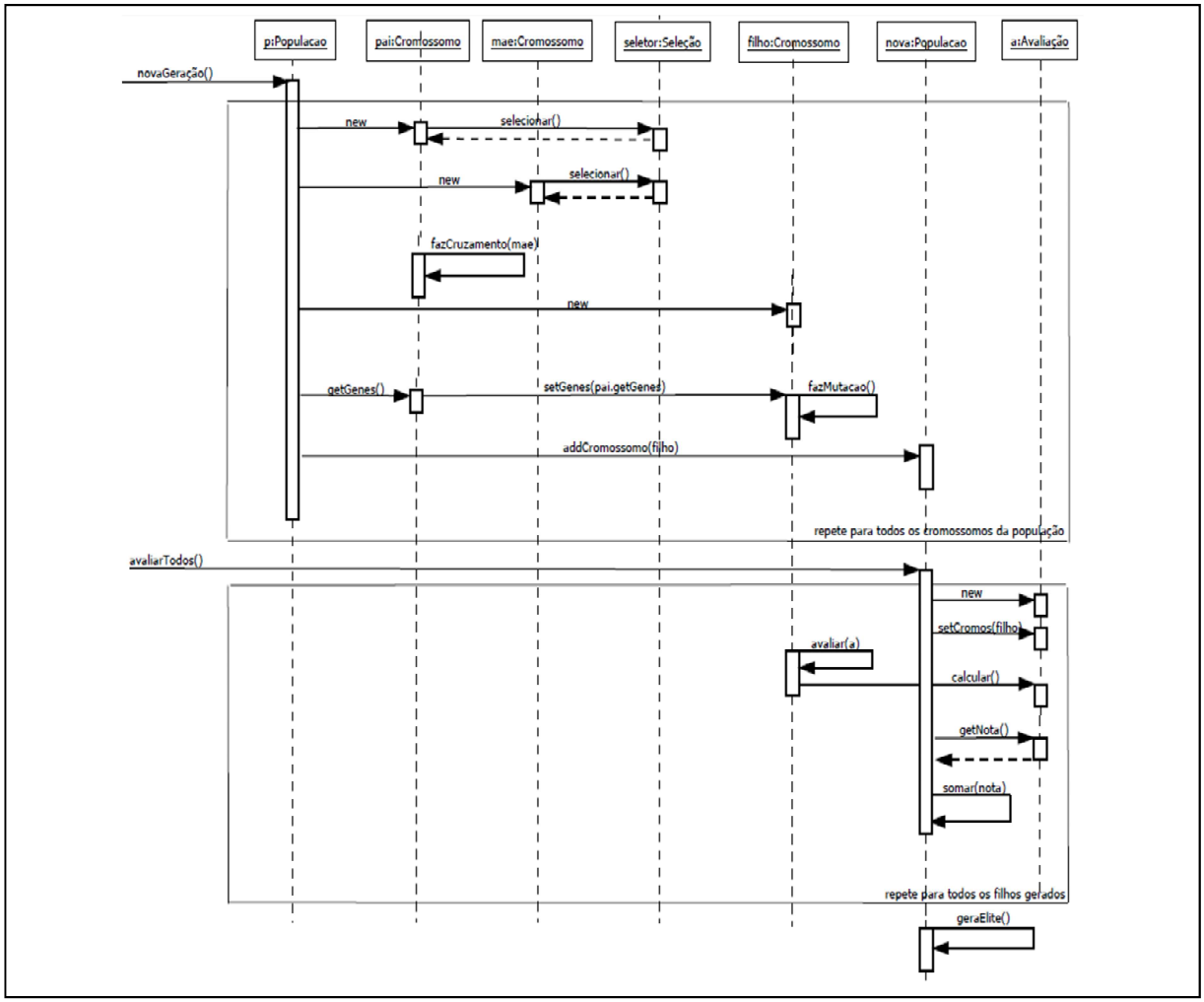


Figura 3.10 Diagrama de Sequência - mensagens trocadas entre objetos em ordem cronológica de execução.

3.7 Considerações Finais

Neste capítulo foi apresentada uma proposta de modelagem do Algoritmo Genético Orientado a Objetos (AGOO), permitindo assim que se construam suas classes, com atributos e métodos, além do diagrama de relacionamento entre classes e diagrama de sequência de troca de mensagens entre os objetos.

Os diagramas apresentados são orientações para escrever o código de um AG genérico com abordagem orientada a objetos. Basicamente, a escolha da representação para os genes do cromossomo e o conteúdo dos métodos de cálculo da avaliação são o que torna o AG específico para solucionar um determinado problema. Isso é feito no capítulo 5, pois a modelagem do AG é aplicada para a síntese de filtros passivos, em que os cromossomos são usados para representar circuitos elétricos.

4. ANÁLISE DE FILTROS PASSIVOS

4.1 Introdução

Os filtros são circuitos elétricos cuja função é bloquear o sinal de entrada para alguma faixa de frequência escolhida. Se a frequência de entrada for diferente da faixa de bloqueio, a saída deve ter a amplitude do sinal de entrada. Em outras palavras, o filtro ideal é o que apresenta ganho zero para frequências pertencentes à faixa de bloqueio e ganho unitário, caso contrário, conforme ilustrado na Figura 4.1 (ORSINI, 1991). A frequência limite (inicial ou final) de uma faixa de bloqueio é denominada frequência de corte (f_c).

O termo passivo se refere aos componentes que formam o circuito do filtro, que são componentes que não necessitam de alimentação para seu funcionamento: são eles os resistores, capacitores e indutores.

No que diz respeito à seletividade em frequência, os filtros podem ser divididos em: Passa-Baixa, Passa-Alta, Passa-Banda e Rejeita-Banda.

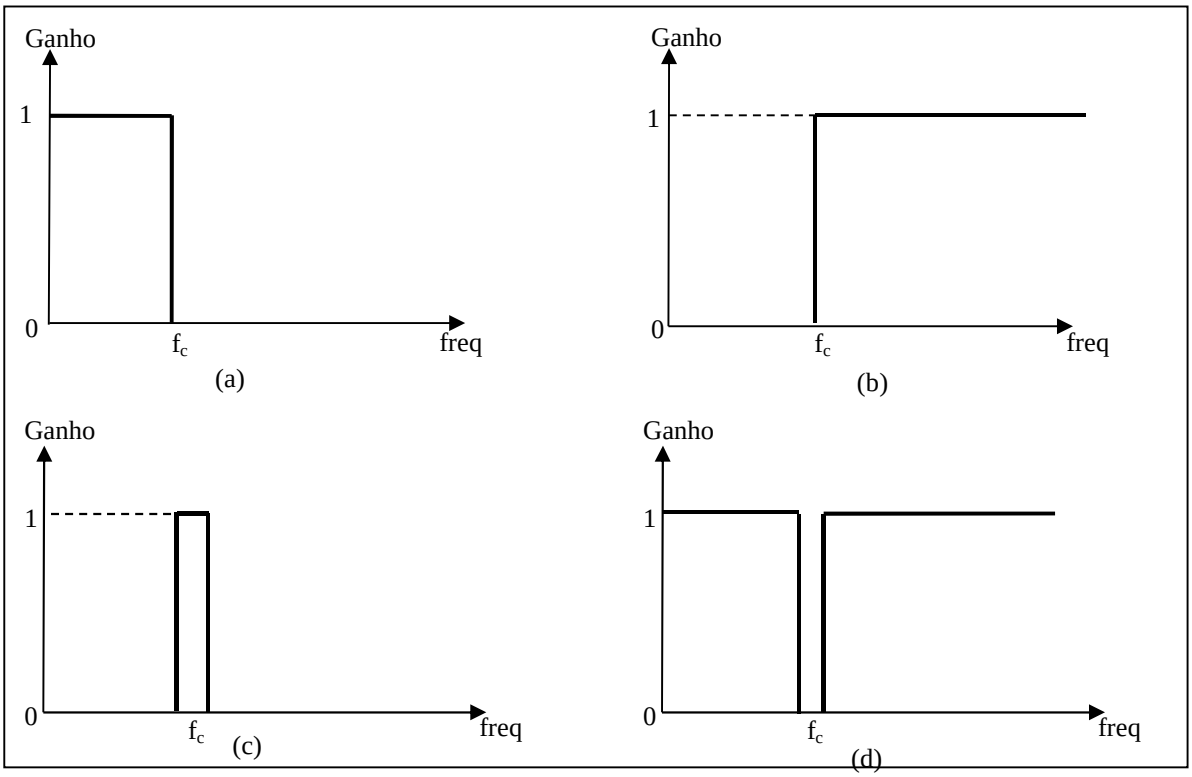


Figura 4.1 Resposta em frequência dos filtros ideais (a) passa-baixa; (b) passa-alta; (c) passa-banda; e (d) rejeita-banda; com frequência de corte f_c. Adaptado de (ORSINI, 1991)

Por ser um circuito seletivo de frequências, os filtros devem ser avaliados no domínio da frequência. A função de transferência que é utilizada neste trabalho é a relação de tensão não dimensional entre a saída e a entrada, conforme visto na Figura 4.2.

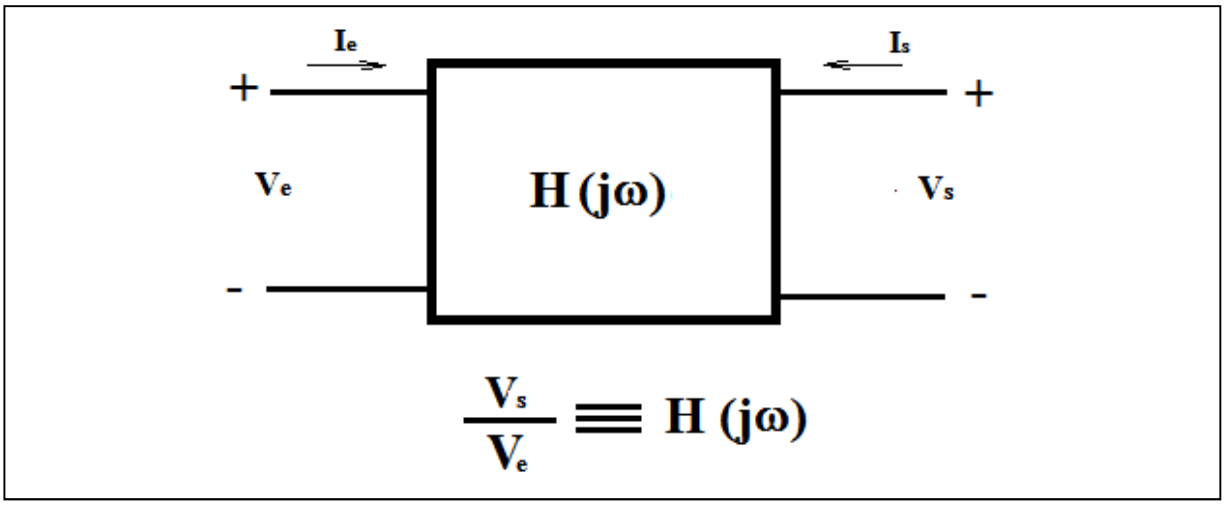


Figura 4.2 Relação de tensão entre saída e entrada do circuito, em função da frequência ω, usada na análise do filtro passivo (EDMINISTER, 1985).

A frequência de corte de um filtro está associada a presença do que se denomina de polos da sua função de transferência. O número de polos, por sua vez, depende da quantidade de elementos reativos (capacitores e indutores). Cada polo de sua função de transferência, correspondendo a um elemento reativo presente no filtro, provoca uma atenuação de 20dB/década na saída (JACOB, 2004). Para um filtro de até três componentes, como o que é sintetizado pelo AGOO neste trabalho, a atenuação máxima que se pode obter na região de transição é de 60 dB/década.

4.2 Análise das Tensões Nodais (ORSINI, 1991)

Considerando um circuito com até 3 componentes, existem diversas possibilidades para montar uma rede RLC: variando a maneira de interconexão dos componentes (topologia), escolhendo a natureza dos componentes (R, L ou C) e, ainda, os valores de cada componente (em Ohm para resistores, Henry para indutores ou Farad para capacitores).

Independentemente do circuito resultante que forma a rede RLC, pode-se aplicar um método computacional de análise de circuito para calcular a tensão de saída do circuito e, assim, obter o ganho do circuito para qualquer que seja a frequência do sinal de entrada.

O método que é utilizado nas avaliações dos circuitos é o chamado método por Análise Nodal (AN), que fornece a tensão em cada nó do circuito em função da frequência, resolvendo o sistema de equações onde os coeficientes são as admitâncias (o inverso da impedância) e as variáveis dependentes são as correntes que incidem em cada nó (ORSINI, 1991). A AN pode ser descrita pela execução dos seguintes passos:

a) Numerar os nós da entrada até a saída, exceto o nó de referência (terra);

b) Montar, por inspeção do circuito, a matriz Y de coeficientes das equações nodais:

   i. Os elementos da diagonal principal, a_{ii}, igual à soma das admitâncias ligadas ao nó i;

   ii. Os outros elementos, a_{ij}, devem conter a admitância entre os nós i e j;

c) Resolver a equação matricial $Y \cdot \underset{\sim}{e} = \underset{\sim}{i}$

   onde

   Y é a matriz de coeficientes, montada conforme descrito em b);

   $\underset{\sim}{e}$ é o vetor das tensões nodais (incógnitas), seguindo a numeração de a);

   $\underset{\sim}{i}$ é o vetor das fontes de correntes ligadas aos nós numerados em a).

d) Calcular a tensão do nó de saída usando a regra de Cramer para resolução de equações matriciais. O módulo da tensão de saída, $|V_S|$, corresponde ao módulo do ganho do circuito, pois:

$$|\text{Ganho}| = |V_S| / |V_e| = |V_S| / 1 = |V_S| \qquad (4.1)$$

A Figura 4.3b apresenta um exemplo de circuito com três admitâncias, onde o sinal de entrada V_e e sua resistência série interna r_e (vistos na Figura 4.3a) foram substituídos pela fonte de corrente I_e com condutância paralela interna g_e, com $I_e = g_e V_e$. Ainda para facilitar o desenvolvimento do método por AN, todas as impedâncias do circuito foram substituídas pelas respectivas admitâncias, fazendo Y=1/Z. Foi também considerada uma resistência de carga R_L para o circuito, substituída por sua condutância G_L.

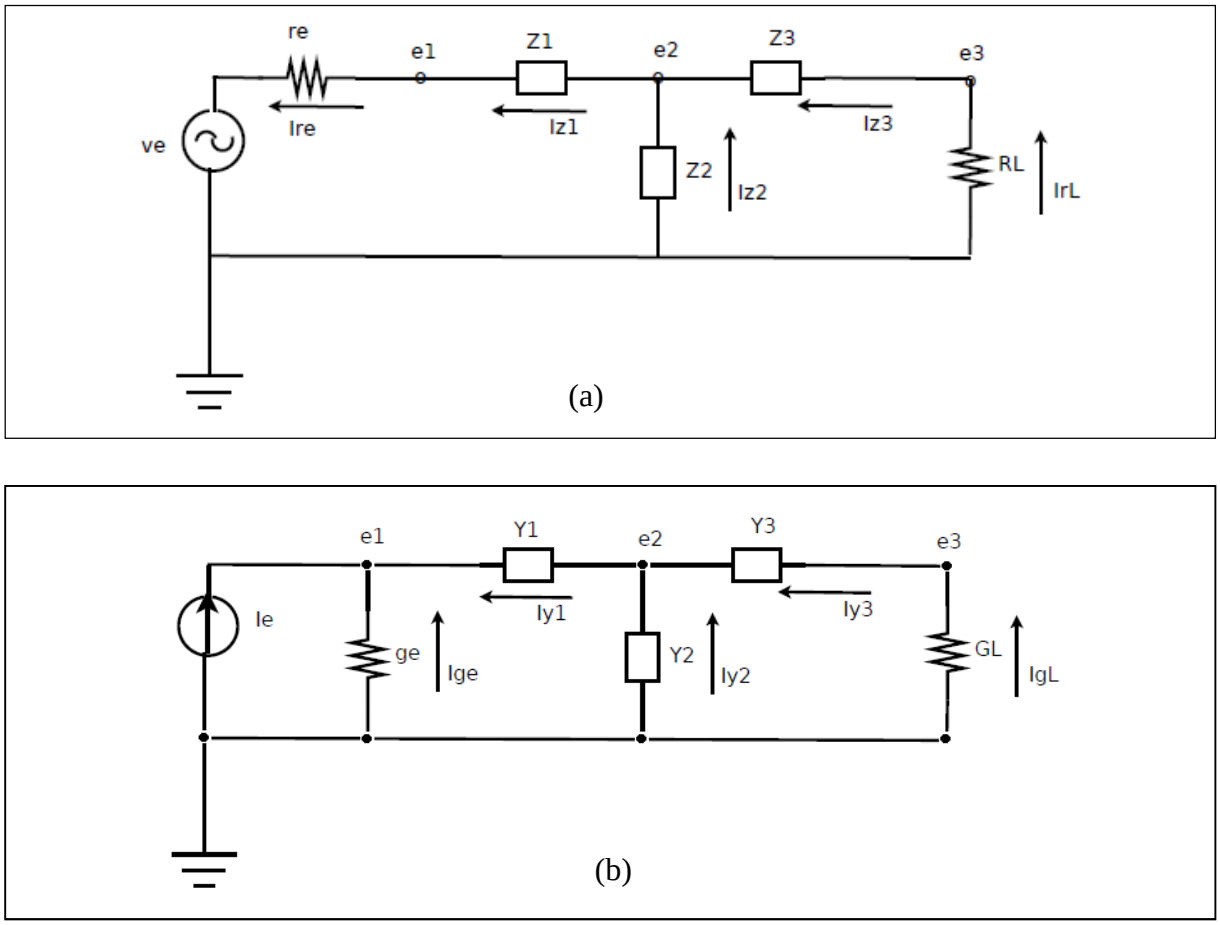


Figura 4.3 Exemplo de circuito com 3 componentes que deve ser avaliado de acordo com seu desempenho como um filtro passivo: (a) circuito com fonte de tensão e impedâncias; (b) o circuito equivalente com fonte de corrente e admitâncias para ser avaliado por AN.

Pela KCL (lei de Kirchhoff para a corrente) aplicada a cada nó do circuito, a soma das correntes em cada nó do circuito é zero (o sentido adotado aqui para as correntes será sempre "saindo" do nó em que se aplica a KCL). As tensões nodais correspondentes aos nós 1, 2 e 3 são indicadas por e_1, e_2 e e_3, respectivamente. Assim:

Nó 1: $-I_e + e_1 g_e + (e_1 - e_2) Y_1 = 0$ (4.2)

Nó 2: $(e_2 - e_1) Y_1 + e_2 Y_2 + (e_2 - e_3) Y_3 = 0$ (4.3)

Nó 3: $(e_3 - e_2) Y_3 + e_3 G_L = 0$ (4.4)

Agrupando os coeficientes das tensões nodais e_1, e_2 e e_3, temos o sistema de equações:

$$\begin{aligned} (g_e + Y_1)\mathbf{e_1} \quad & - Y_1\mathbf{e_2} \quad && 0\mathbf{e_3} \quad && = I_e \\ -Y_1\mathbf{e_1} \quad & +(Y_1+Y_2+Y_3)\mathbf{e_2} \quad && -Y_3\mathbf{e_3} \quad && = 0 \\ 0\mathbf{e_1} \quad & -Y_3\mathbf{e_2} \quad && +(Y_3+G_L)\mathbf{e_3} \quad && = 0 \end{aligned} \tag{4.5}$$

Escrevendo o sistema de equações (4.4) na forma matricial:

$$\begin{pmatrix} (g_e + Y_1) & - Y_1 & 0 \\ -Y_1 & (Y_1+Y_2+Y_3) & -Y_3 \\ 0 & -Y_3 & (Y_3+G_L) \end{pmatrix} \begin{pmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \mathbf{e_3} \end{pmatrix} = \begin{pmatrix} I_e \\ 0 \\ 0 \end{pmatrix} \tag{4.6}$$

A matriz do primeiro membro de (4.6) contém os coeficientes das tensões nodais $\mathbf{e_1}$, $\mathbf{e_2}$ e $\mathbf{e_3}$ e a sua diagonal principal, contendo os elementos a_{11}, a_{22} e a_{33}, apresenta a soma das admitâncias ligadas a cada nó, enquanto que nas outras posições da matriz, elementos a_{ik}, tem-se o oposto das admitâncias que ligam o nó *i* ao nó *k*, portanto $a_{ik} = a_{ki}$.

Para calcular o ganho do circuito para três admitâncias escolhidas Y_1, Y_2 e Y_3, devemos fixar arbitrariamente os valores de V_e, r_e e R_L ou I_e, g_e e G_L, para que o ganho do circuito varie apenas com a escolha dos componentes RLC.

Deve ser calculado, então, o valor da relação e_3/v_e, que corresponde ao ganho do circuito, pois $e_3 = v_s$.

Resolvendo a equação matricial (4.6):

$$V_S = \hat{E}_3 = \frac{\begin{vmatrix} (g_e + Y_1) & - Y_1 & Ie \\ -Y_1 & (Y_1+Y_2+Y_3) & 0 \\ 0 & -Y_3 & 0 \end{vmatrix}}{\begin{vmatrix} (g_e + Y_1) & - Y_1 & 0 \\ -Y_1 & (Y_1+Y_2+Y_3) & -Y_3 \\ 0 & -Y_3 & (Y_3+G_L) \end{vmatrix}} \tag{4.7}$$

Para simplificar os cálculos, escolhem-se:

$v_e = 1\cos(\omega t + 0°)$ ou, na forma de fasor, $V_e = 1 \angle 0°$;

$g_e = 1$ S; e

$G_L = (1/100x10^6)$ S - resistência de carga igual a 100MΩ para se aproximar da situação de saída em aberto;

Para obter o módulo do ganho do circuito, o que corresponde $|H(j\omega)|$, basta calcular o fasor $\hat{E}_3$, correspondente à tensão nodal e_3, pois:

$$\text{Ganho} = |H(j\omega)| = |V_S / Ve| = |\hat{E}_3 / (1 \angle 0°)| = |\hat{E}_3| \qquad (4.8)$$

As admitâncias da Figura 4.3 correspondem aos componentes passivos do filtro, cujos valores são calculados conforme a terceira coluna da Tabela III.

Tabela III. Impedâncias e admitâncias dos componentes passivos (EDMINISTER, 1985).

Componente	***Impedância (Z) em Ω***	***Admitância (Y) em S***
Resistor (R)	$R + j0$	$(1/R) + j0$
Capacitor (C)	$0 - j(1/\omega C)$	$0 + j\omega C$
Indutor (L)	$0 + j\omega L$	$0 - j(1/\omega L)$

Onde $\omega = 2\pi f$ (rad/s) é a frequência do sinal de entrada do filtro (v_e).

Como o valor da impedância formada por indutores ou capacitores depende da frequência do sinal, o ganho deve ser calculado para diversos valores de ω, a fim de que seja obtida a resposta em frequência do circuito. Assim, será definida uma forma de avaliação do circuito a partir do erro entre o ganho calculado, H(jω), e o ganho esperado, O(jω), para cada frequência ω_i usada em (4.7):

$$Erro = \sum_i \left| H(j\omega_i) - O(j\omega_i) \right| \qquad (4.9)$$

Outras configurações podem ser consideradas para o circuito do filtro passivo mantendo a mesma metodologia para o cálculo das tensões nodais:

A Tabela IV apresenta quinze topologias possíveis, usando apenas três componentes para a construção do filtro, e as respectivas matrizes de coeficientes do método por AN. Com

o intuito de simplificar o circuito, a fonte de corrente e as condutâncias de entrada e saída foram omitidas, porém g_e e $G_{L,}$ conectadas ao nó de entrada e de saída, respectivamente, são considerados na matriz de coeficientes da segunda coluna. Algumas topologias não foram consideradas na tabela porque correspondem a casos triviais em que só existe um nó interno (e1) que é igual ao nó de saída e, portanto, entre a tensão de entrada, Ve, e a tensão de saída, Vs, há apenas a resistência interna, r_e, da fonte do sinal de entrada, que foi assumida como um resistor de apenas 1Ω. O efeito prático desta condição é que se pode considerar |Vs|=|Ve| e, portanto, ganho igual a 1, independentemente da frequência do sinal de entrada. Um exemplo de topologia desse tipo é o da Figura 4.4.

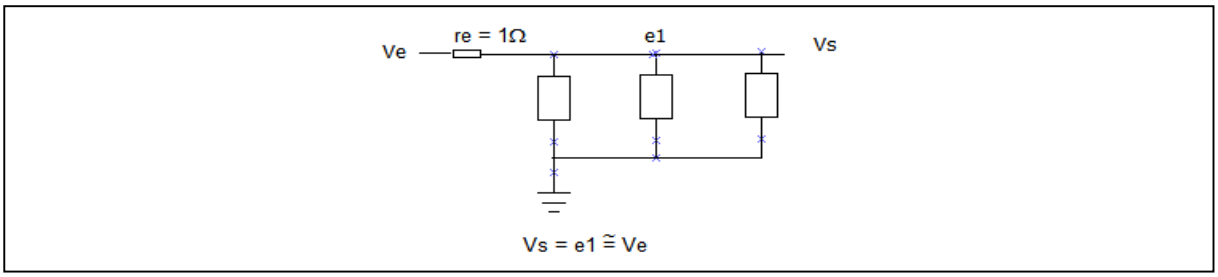


Figura 4.4 Caso de topologia trivial: $Vs=e_1 \approx Ve$. Para qualquer que seja a frequência do sinal de entrada, o ganho do circuito, |Vs|/|Ve|, será aproximadamente unitário.

Tabela IV. Topologias possíveis do circuito com três componentes e as respectivas matrizes de coeficientes das tensões nodais.

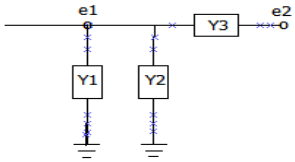
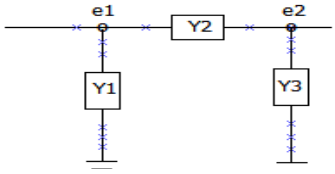
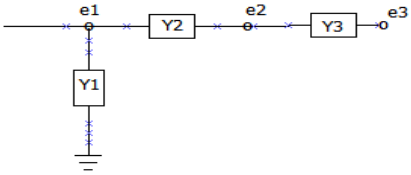
Circuito	Topologia	Matriz de coeficientes
1		$\begin{bmatrix} (g_e+Y_1+Y_2+Y_3) & -Y_3 \\ -Y_3 & (Y_3+G_L) \end{bmatrix}$
2		$\begin{bmatrix} (g_e+Y_1+Y_2) & -Y_2 \\ -Y_2 & (Y_2+Y_3+G_L) \end{bmatrix}$
3		$\begin{bmatrix} (g_e+Y_1+Y_2) & -Y_2 & 0 \\ -Y_2 & (Y_2+Y_3) & -Y_3 \\ 0 & -Y_3 & (Y_3+G_L) \end{bmatrix}$

Tabela IV. Topologias possíveis do circuito com três componentes e as respectivas matrizes de coeficientes das tensões nodais (cont.).

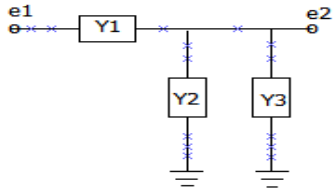
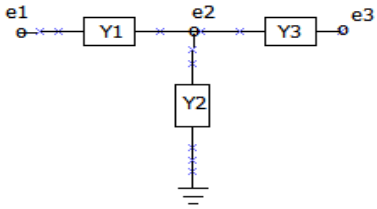
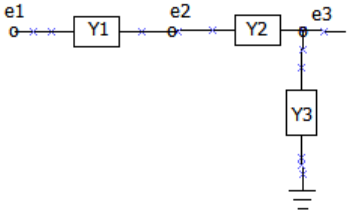
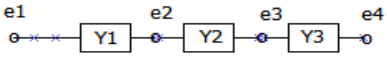
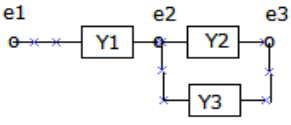
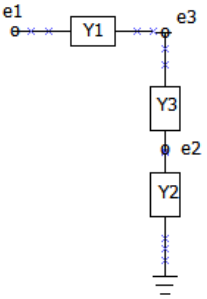
Circuito	Topologia	Matriz de coeficientes
4		$\begin{pmatrix} (g_e+Y_1) & -Y_1 \\ -Y_1 & (Y_1+Y_2+Y_3+G_L) \end{pmatrix}$
5		$\begin{pmatrix} (g_e+Y_1) & -Y_1 & 0 \\ -Y_1 & (Y_1+Y_2+Y_3) & -Y_3 \\ 0 & -Y_3 & (Y_3+G_L) \end{pmatrix}$
6		$\begin{pmatrix} (g_e+Y_1) & -Y_1 & 0 \\ -Y_1 & (Y_1+Y_2) & -Y_2 \\ 0 & -Y_2 & (Y_2+Y_3+G_L) \end{pmatrix}$
7		$\begin{pmatrix} (g_e+Y_1) & -Y_1 & 0 & 0 \\ -Y_1 & (Y_1+Y_2) & -Y_2 & 0 \\ 0 & -Y_2 & (Y_2+Y_3) & -Y_3 \\ 0 & 0 & -Y_3 & (Y_3+G_L) \end{pmatrix}$
8		$\begin{pmatrix} (g_e+Y_1) & -Y_1 & 0 \\ -Y_1 & (Y_1+Y_2+Y_3) & -(Y_2+Y_3) \\ 0 & -(Y_2+Y_3) & (Y_2+Y_3+G_L) \end{pmatrix}$
9		$\begin{pmatrix} (g_e+Y_1) & 0 & -Y_1 \\ 0 & (Y_2+Y_3) & -Y_3 \\ -Y_1 & -Y_3 & (Y_1+Y_3+G_L) \end{pmatrix}$

Tabela IV. Topologias possíveis do circuito com três componentes e as respectivas matrizes de coeficientes das tensões nodais (cont.).

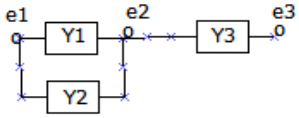
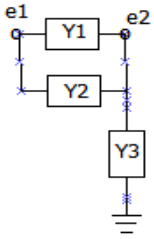
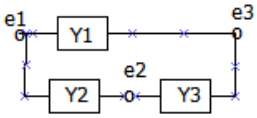
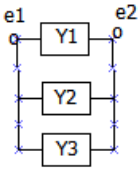
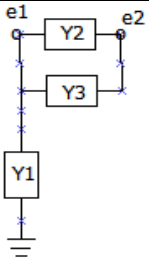
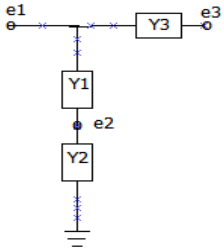
Circuito	Topologia	Matriz de coeficientes
10		$\begin{pmatrix} (g_e+Y_1+Y_2) & -(Y_1+Y_2) & 0 \\ -(Y_1+Y_2) & (Y_1+Y_2+Y_3) & -Y_3 \\ 0 & -Y_3 & (Y_3+G_L) \end{pmatrix}$
11		$\begin{pmatrix} (g_e+Y_1+Y_2) & -(Y_1+Y_2) \\ -(Y_1+Y_2) & (Y_1+Y_2+Y_3+G_L) \end{pmatrix}$
12		$\begin{pmatrix} (g_e+Y_1+Y_2) & -Y_2 & -Y_1 \\ -Y_2 & (Y_2+Y_3) & -Y_3 \\ -Y_1 & -Y_3 & (Y_1+Y_3+G_L) \end{pmatrix}$
13		$\begin{pmatrix} (g_e+Y_1+Y_2+Y_3) & -(Y_1+Y_2+Y_3) \\ -(Y_1+Y_2+Y_3) & (Y_1+Y_2+Y_3+G_L) \end{pmatrix}$
14		$\begin{pmatrix} (g_e+Y_1+Y_2+Y_3) & -(Y_2+Y_3) \\ -(Y_2+Y_3) & (Y_2+Y_3+G_L) \end{pmatrix}$

Tabela IV. Topologias possíveis do circuito com três componentes e as respectivas matrizes de coeficientes das tensões nodais (cont.).

Circuito	Topologia	Matriz de coeficientes
15		$\begin{pmatrix} (g_e+Y_1+Y_3) & -Y_1 & -Y_3 \\ -Y_1 & (Y_1+Y_2) & 0 \\ -Y_3 & 0 & (Y_3+G_L) \end{pmatrix}$

A matriz de coeficientes de algumas topologias pode ser reduzida em dimensão fazendo associação de admitâncias para eliminar algum nó do circuito, como, por exemplo, o circuito 3 da tabela IV. A Figura 4.5 apresenta a modificação do circuito com a finalidade de eliminar o nó entre Y_2 e Y_3, substituindo estas admitâncias por sua equivalente da associação em série entre elas:

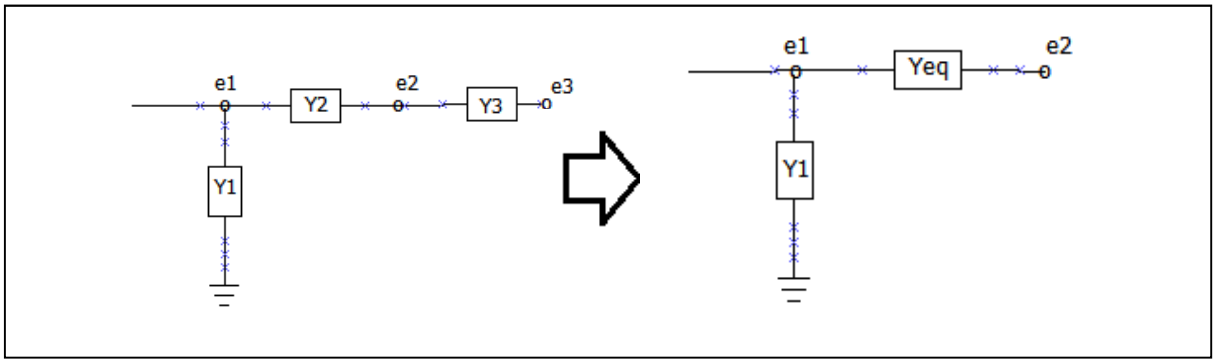


Figura 4.5 Substituição de Y_2 e Y_3 do circuito da esquerda por sua admitância equivalente, Y_{eq}, (circuito da direita) para diminuir a quantidade de nós do circuito.

Assim, o sistema de equações do circuito terá uma matriz de coeficientes das tensões nodais reduzida a 2x2:

$$\begin{pmatrix} (g_e+Y_1+Y_{eq}) & -Y_{eq} \\ -Y_{eq} & (Y_{eq}+G_L) \end{pmatrix} \begin{pmatrix} \mathbf{e_1} \\ \mathbf{e_2} \end{pmatrix} = \begin{pmatrix} I_e \\ 0 \end{pmatrix} \qquad (4.10)$$

Onde,

$$Yeq = \frac{Y_2 Y_3}{Y_2 + Y_3} \tag{4.11}$$

Com a matriz de coeficientes reduzida, basta calcular o módulo da tensão e_2 usando as equações (4.10) e (4.11) para obter o ganho do circuito para cada valor de ω escolhido.

4.3 Análise da Resposta em Frequência

Realizando o cálculo da relação de tensão entre a saída e a entrada do circuito para uma série conveniente de valores de frequência, pode-se obter um perfil do comportamento do circuito para as diversas frequências consideradas, o que é chamado de resposta em frequência do circuito (EDMINISTER, 1985). A resposta em frequência do circuito permite avaliá-lo quanto à sua adequação ao filtro que se procura com a execução do algoritmo.

As respostas em frequência apresentadas anteriormente na Figura 4.1 são idealizações para os filtros, pois a transição entre a região de corte e a região de passagem é representada por um degrau, o que seria a resposta de um filtro perfeito. Na prática, existe uma rampa na curva da resposta em frequência para a transição entre as regiões de corte e de passagem, conforme se observa na Figura 4.6. A inclinação dessa rampa está relacionada com a ordem do filtro, ou seja, com a quantidade de componentes reativos (capacitores e indutores) presentes no circuito, pois estes são os componentes capazes de introduzir polos na função de transferência, H(jω), do circuito RLC.

Pela Figura 4.6, definem-se ω_{ci} e ω_{cs} como as frequências de corte inferior e superior, respectivamente, que são usadas para medir (em dB/década) a inclinação da rampa de transição entre as regiões da resposta em frequência dos filtros. Também está indicada na Figura 4.6 a frequência de corte central ω_{cc}, que corresponde à frequência de corte (f_c) escolhida como requisito de projeto para o filtro:

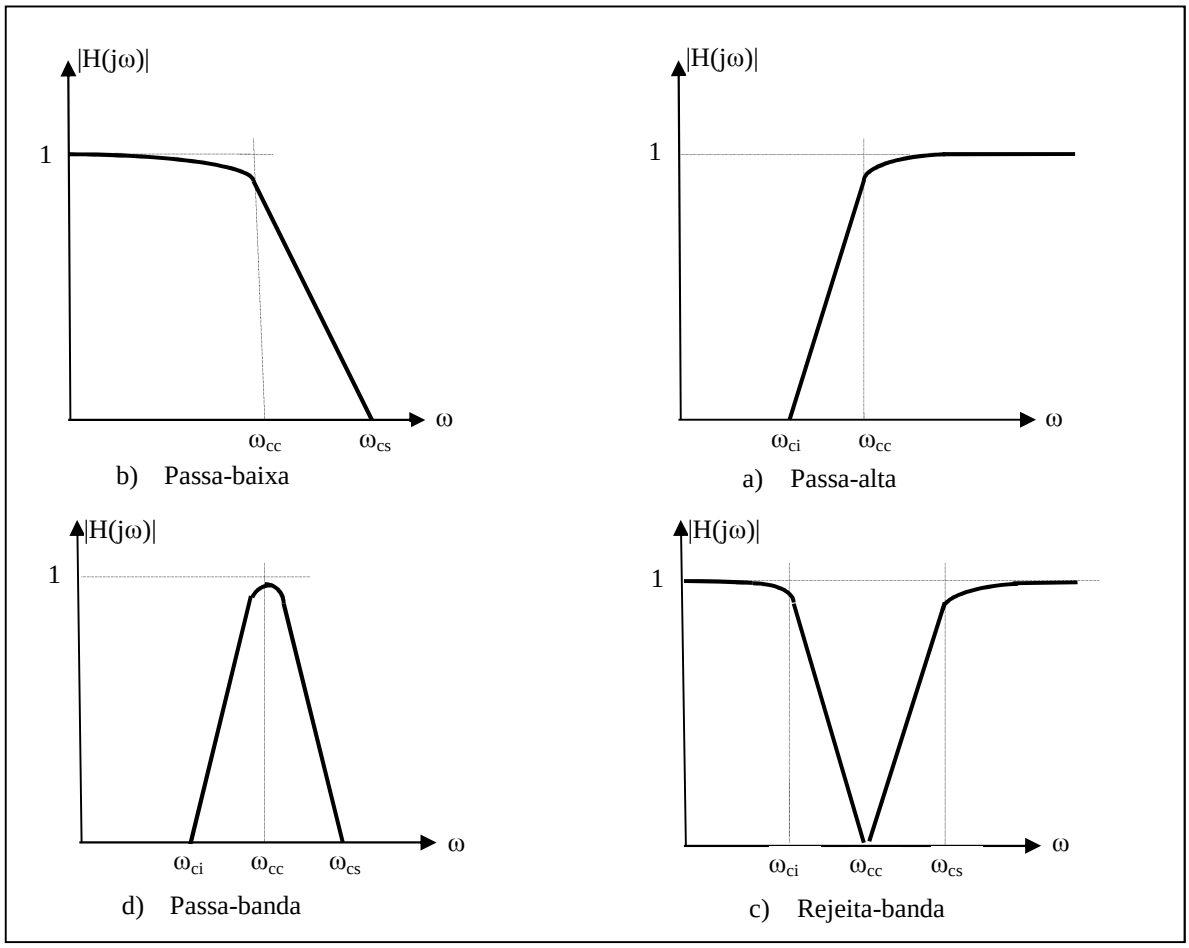


Figura 4.6 Resposta em frequência mais realística dos filtros RLC com a indicação das frequências angulares de corte inferior, central e superior.

Cada elemento armazenador de energia inserido no circuito, ou seja, capacitor ou indutor, causa um incremento na ordem da equação diferencial que descreve seu comportamento (JACOB, 2004), pois:

$$v_L = L\frac{di}{dt} \qquad i_c = C\frac{dv}{dt} \tag{4.12}$$

Passando (4.12) para o domínio da frequência, o operador de derivação d/dt usado no domínio do tempo é substituído por jω no cálculo de tensão e corrente no domínio da frequência, assim como o operador de integração ∫dt é substituído por -j/ω (EDMINISTER, 1985), pois:

Seja $i(t) = Isen\omega t$ um sinal senoidal para a corrente elétrica do circuito com frequência angular ω. Então:

$$v_L = L\frac{di}{dt} = L\frac{d}{dt}[Isen\omega t] = LI\omega sen(\omega t + 90^\circ) = j\omega LI = V_L \qquad (4.13)$$

A unidade imaginária *j* indica a defasagem de 90° em avanço da tensão no indutor em relação à corrente que o percorre e o valor ωL é a sua reatância indutiva (ORSINI, 1991).

Analogamente, para o capacitor, tem-se: $i_C = C\frac{dv}{dt}$ ou $v_C = \frac{1}{C}\int i(t)\cdot dt$

Assim,

$$v_C = \frac{1}{C}\int Isen\omega t \cdot dt = \frac{1}{\omega C} Isen(\omega t - 90^\circ) = \frac{-j}{\omega C} I = V_C \qquad (4.14)$$

Aqui o uso da unidade imaginária *-j* indica a defasagem de 90° em atraso da tensão no capacitor em relação à corrente que o percorre e o valor 1/ωC é a sua reatância capacitiva (ORSINI, 1991).

As equações (4.13) e (4.14) mostram que o valor de tensão no indutor e no capacitor depende da sua respectiva reatância, que por sua vez depende da frequência ω do sinal.

Um circuito RLC com *n* componentes armazenadores de energia será descrito por uma equação diferencial de ordem *n*, contendo um termo com $\frac{d^n}{dt^n}$ no domínio do tempo, que terá correspondência com o termo $(j\omega)^n$ no domínio da frequência. Na função de transferência do circuito, o grau *n* fornece a quantidade de polos da equação. Um filtro cuja função de transferência possui *n* polos terá uma atenuação de *20n* dB/década, pois:

$H(j\omega) = 1/\omega^n$ é a função de transferência do circuito com *n* polos;

A_1 é o ganho do circuito para $\omega = \omega_c$ → $A_1 = 20\log|1/\omega_c^n|$;

A_2 é o ganho do circuito para $\omega = 10\omega_c$ (1 década maior) → $A_2 = 20\log|1/10^n\omega_c^n|$.

Então

$$\begin{aligned} \Delta A = \quad A_1 - A_2 \quad &= 20\log|1/\omega_c^n| - 20\log|1/10^n\omega_c^n| \\ &= 20\log|(1/\omega_c^n)/(1/10^n\omega_c^n)| \\ &= 20\log|10^n| \\ &= 20n\log|10| \\ &= 20n\ dB \end{aligned} \qquad (4.15)$$

O ganho a meia potência é outro parâmetro importante para a resposta em frequência dos filtros e corresponde ao ganho para o qual metade da potência do sinal de entrada é transferida para a carga. Este ganho corresponde a uma queda de 3db em relação ao ganho máximo (atenuação de 0db) (JACOB, 2004):

$$Ganho_{1/2} = 10\log\left(\frac{\frac{Pin}{2}}{Pin}\right) = 10\log(\frac{1}{2}) \cong -3dB \qquad (4.16)$$

A importância deste parâmetro para os filtros consiste no fato de que a partir da frequência do sinal para a meia potência, indicada por f_{-3dB}, se observa a rampa de atenuação do ganho à taxa de 20*n* dB/década (JACOB, 2004), o que equivale a 6*n* dB/oitava, como se vê na Figura 4.7.

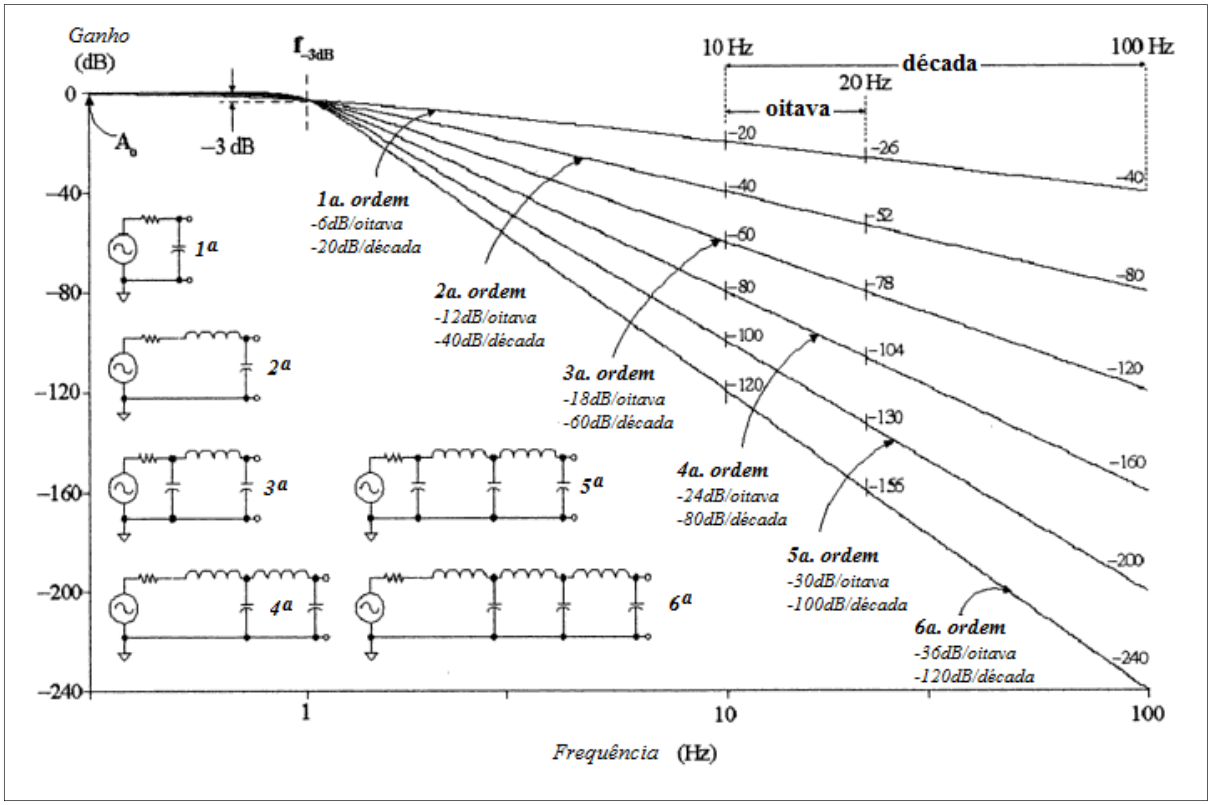


Figura 4.7 Respostas em frequência de filtros passivos passa-baixa desde a primeira até a sexta ordem (JACOB, 2004)

4.4 Considerações Finais

Neste capítulo foram apresentados alguns conceitos sobre filtros passivos e um método analítico para o cálculo das tensões nodais, chamado de Análise Nodal (AN), com o objetivo

de obter o valor de tensão na saída do circuito para compará-lo ao valor de entrada, obtendo assim a chamada função de transferência do circuito, que depende da frequência do sinal de entrada. Também foram apresentados alguns parâmetros importantes para a análise da resposta em frequência dos circuitos RLC, que são importantes para o desenvolvimento da função de avaliação do AG, pois é necessário medir o desempenho de vários circuitos candidatos à solução procurada.

A resposta em frequência tratada neste trabalho foi apenas a referente ao ganho em magnitude, ou seja, em módulo da função de transferência. Para obter a resposta em frequência completa é necessário levantar também a variação do ângulo da função de transferência em função da frequência do sinal de entrada, obtendo assim a curva de fase (EDMINISTER, 1985):

$$\varphi(\omega) = \arctan \frac{\mathrm{Im}[H(j\omega)]}{\mathrm{Re}[H(j\omega)]} \tag{4.17}$$

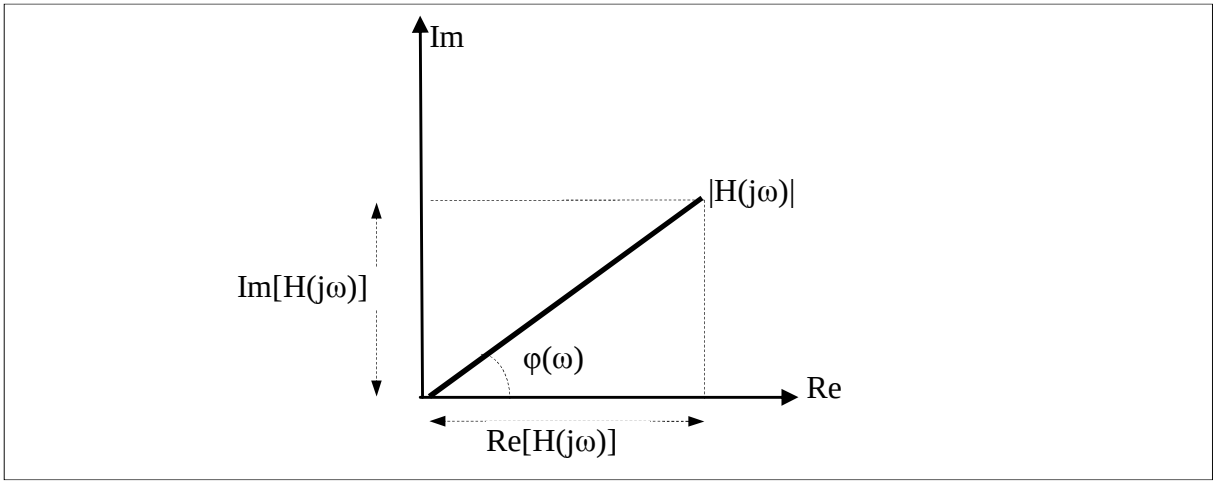


Figura 4.8 Representação da função de transferência como um valor complexo, com módulo e ângulo.

Onde Im[H(jω)] e Re[H(jω)] são, respectivamente, as partes imaginária e real da função de transferência do circuito, como se vê na Figura 4.8.

A fase da função de transferência não foi considerada para a avaliação do circuito, visto que a seletividade de frequências do filtro não depende de sua resposta em fase.

5. SÍNTESE AUTOMÁTICA DE FILTROS PASSIVOS USANDO AGOO

5.1 Introdução

A atividade de projetar um circuito elétrico, que basicamente é um processador de sinais analógicos destinado a realizar alguma função prática, é uma atividade de difícil automatização por meio computacional porque o sucesso do projeto tem alta dependência do conhecimento técnico, da experiência e da intuição do desenvolvedor, ou seja, da capacidade intelectual humana, o que, em geral, não se resume a um conjunto de regras lógicas. No entanto, se houver meios de representar um circuito por uma sequência de bits ou por uma estrutura de dados e, a partir de sua representação, for possível avaliar seu desempenho, podem-se gerar novas coleções de circuitos cujas avaliações são orientadoras para a geração de novos circuitos com melhor desempenho, de forma cíclica, até que uma coleção contenha um circuito tão bom quanto o desejado.

Neste trabalho os indivíduos são circuitos elétricos analógicos RLC (EDMINISTER, 1985) com até três componentes. O objetivo é determinar a topologia (interconexão entre os componentes), o tipo de cada componente (resistor, capacitor ou indutor) e seus valores (em Ohm, Farad ou Henry) de forma que o circuito se constitua em um filtro de sinais elétricos (ORSINI, 1991) com os seguintes requisitos atendidos: a frequência de corte desejada e a função do circuito (passa-baixa, passa-alta, passa-banda ou rejeita-banda). O motivo pelo qual o AG foi escolhido para automatizar o projeto dos filtros é porque se trata de um algoritmo de busca evolucionário que substitui o conhecimento humano do projetista de circuitos por uma série de análises e processamento das coleções de circuitos candidatos. Outra razão: as etapas envolvidas no AG compõem um ótimo exemplo para modelagem do problema do mundo real em classes que interagem entre si e, assim, a sua construção é indicada para a programação orientada a objetos, o que é outro aspecto importante na proposta deste trabalho, pois permite apresentar a distribuição de tarefas entre as classes criadas, a modularidade do código e a flexibilidade para inserir variações que atendam às mudanças no problema.

A Figura 5.1 apresenta o circuito evolucionário inserido no circuito completo do filtro, que inclui o sinal de entrada $v_e = 1 \cos(\omega t + 0°)$ (V), a resistência de entrada r_e=50 Ω, o circuito RLC a ser sintetizado e a resistência de carga R_L=100 MΩ. A escolha dos elementos externos ao circuito evolucionário, e seus respectivos valores, foi realizada deliberadamente para atender a dois objetivos:

a) Tornar o cálculo do ganho do circuito mais simples, visto que este cálculo será processado inúmeras vezes para avaliar a qualidade do circuito, substituindo o uso de simuladores externos;

b) Fazer com que a perda por inserção do filtro seja baixa, pois esta perda é diretamente proporcional a $\log[R_L/(r_e + R_L)]$ (ORSINI, 1991).

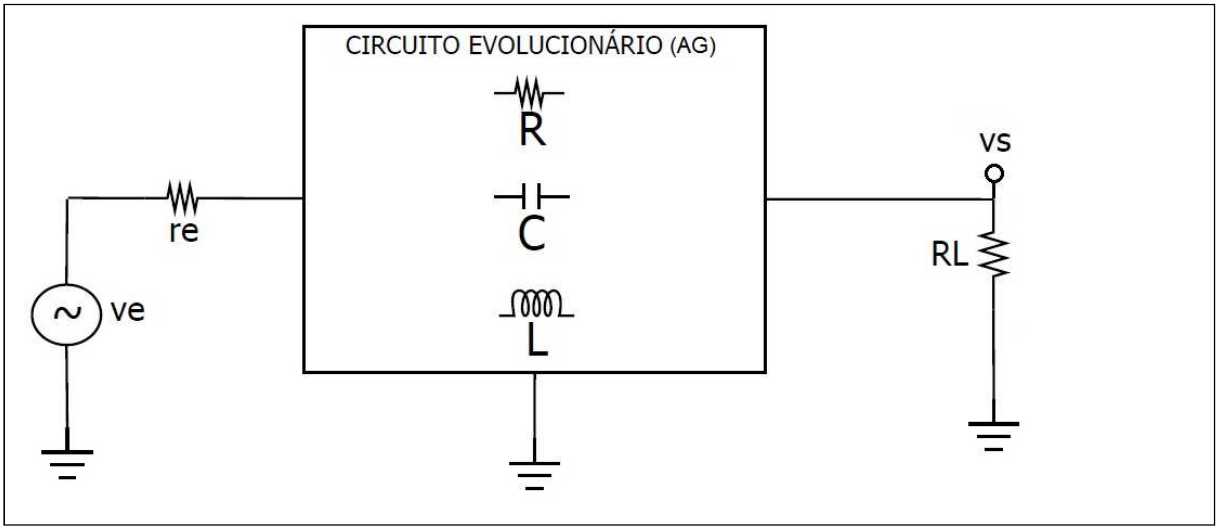


Figura 5.1 Diagrama do circuito completo indicando o bloco correspondente ao filtro passivo RLC que é evoluído por AG (Algoritmo Genético).

A seguir é apresentado o relacionamento entre as classes que foram criadas para a modelagem do AGOO (Algoritmo Genético Orientado a Objetos) aplicado à síntese de filtros passivos.

5.2 Modelamento

A Figura 5.2 apresenta o diagrama de classes em que foi baseado o modelo para o desenvolvimento do algoritmo genético que sintetiza um circuito de filtro passivo. A modelagem do problema é composta pelas seguintes classes:

População; Circuito; Cromossomo; Seleção e suas subclasses; Avaliação e suas subclasses; MatrizAN e suas subclasses; Complexo e OperadorComplexo.

Os papéis de cada classe definidos no projeto orientado a objetos do sintetizador do filtro são descritos nas subseções a seguir:

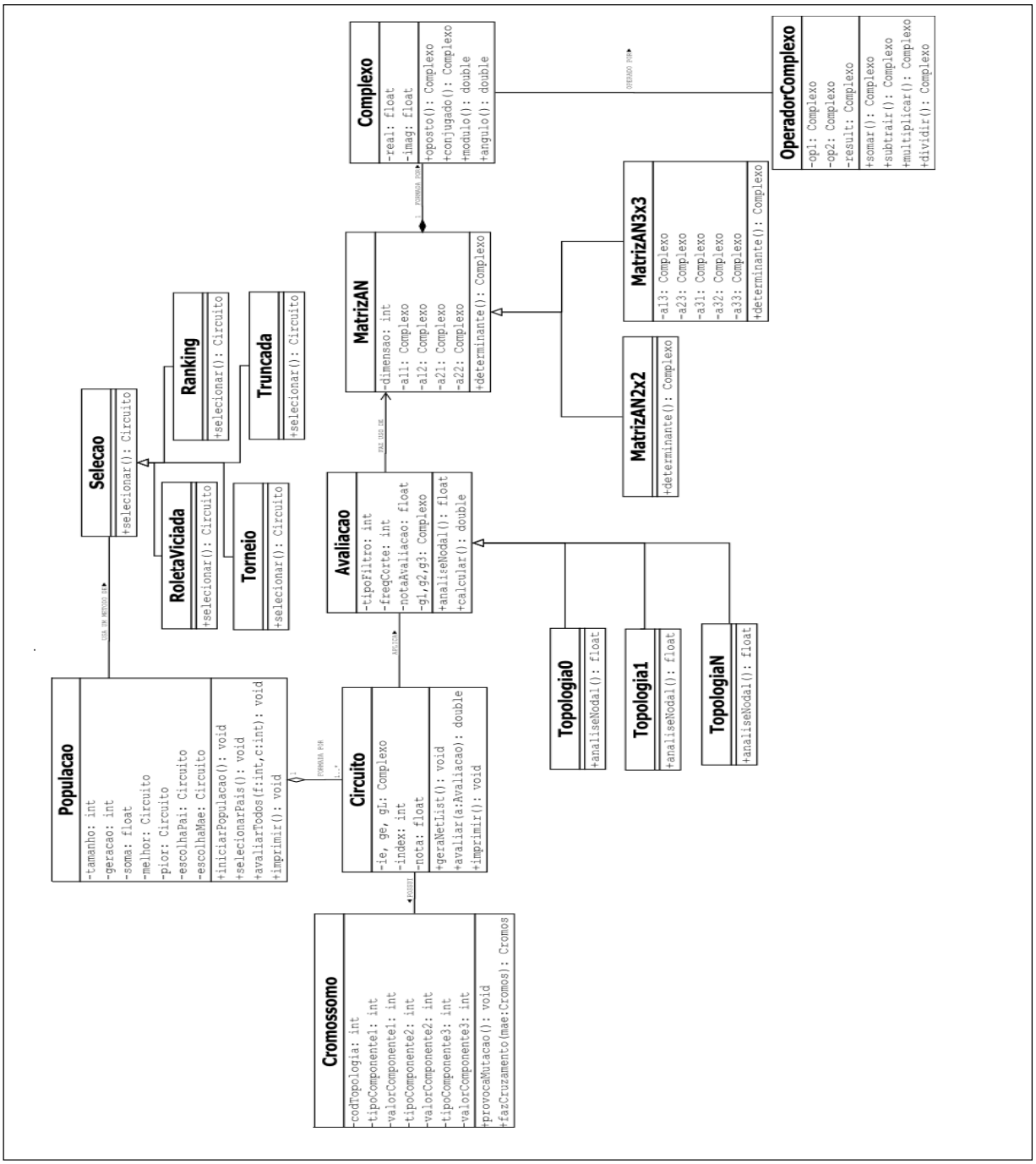


Figura 5.2 Diagrama de Classes usado no projeto orientado a objetos para a síntese do filtro passivo.

5.2.1 População

A classe População mantém as informações de tamanho e identificação da geração atual. É responsável pela criação da população inicial e alguns procedimentos do AG, tais como invocar o método de seleção e cruzamento dos pais, mutação de indivíduos, e invocar a avaliação de todos os circuitos. É formada por uma lista de objetos da classe Circuito e se

relaciona com a classe Seleção para obter, via um dos métodos de seleção escolhido, dois circuitos para a realização de cada cruzamento.

5.2.2 Seleção

Esta classe agrega ao projeto a parte do código responsável pela execução do método de seleção de circuitos e possui várias subclasses, sendo que cada uma delas realiza um algoritmo de seleção particular, cabendo ao usuário escolher o método de seleção desejado. A classe mãe, Selecao, é abstrata, pois não possui código próprio, apenas chama a execução do método *selecionar* da classe filha que foi instanciada pelo usuário. Os métodos de seleção implementados neste projeto são RoletaViciada e Torneio, porém outros podem ser incluídos, como a seleção por Ranking e a Truncada (LINDEN, 2012), graças à flexibilidade inerente de um projeto orientado a objetos.

5.2.3 Circuito

Cada circuito elétrico se relaciona com a sua representação cromossômica, que é um de seus atributos, e com sua nota de avaliação. Um circuito é um indivíduo da população e mantém as informações dos componentes e topologia que formam o filtro e realiza as tarefas pertinentes ao circuito: imprimir a lista de conexões (*netlist*), solicitar a nota de sua avaliação, entre outras.

5.2.4 Cromossomo

A classe Cromossomo é responsável pela execução dos cruzamentos entre circuitos e eventualmente pela mutação em um dos circuitos gerados pelo cruzamento. Esta classe contém a estrutura de dados que representa o circuito na forma de um cromossomo e implementa os operadores genéticos utilizados no AG.

5.2.5 Avaliação

A classe Avaliação executa a análise nodal para o circuito atribuindo-lhe uma nota de acordo com o valor de tensão na saída (v_s) obtido para uma determinada frequência do sinal de entrada (v_e) e se relaciona com a classe MatrizAN para construir a matriz das equações das tensões dos nós do circuito avaliado.

5.2.6 MatrizAN

A classe MatrizAN, por sua vez, pode ser uma matriz 2x2 ou 3x3 (representada pelas classes filhas MatrizAN2x2 e MatrizAN3x3) e é composta por números complexos associados aos valores das admitâncias do circuito. Esta classe contém o método para resolução de determinante de matrizes de números complexos.

5.2.7 Complexo

Classe que implementa a representação dos valores contidos na matriz de análise nodal e implementa as operações unárias dos números complexos, como cálculo do módulo, do ângulo e do oposto, além de chamar, por meio da relação com a classe OperadorComplexo, as operações básicas entre dois valores complexos necessárias para o cálculo do determinante de matrizes.

5.2.8 OperadorComplexo

As operações de soma, subtração, multiplicação e divisão entre dois números complexos são desenvolvidas nos métodos desta classe, que se relaciona exclusivamente com a classe Complexo.

5.3 O Cromossomo do Circuito

A classe Circuito contém atributos relativos ao circuito procurado para realização do filtro e um dos seus atributos é a classe Cromossomo, que representa a parte do circuito a ser evoluída, ou seja, o cromossomo da solução. Esta solução pode ser imaginada como uma sequência de bits, ou uma estrutura de dados, que define de forma unívoca um indivíduo da população. Neste trabalho, o cromossomo de um indivíduo define a topologia do circuito, o tipo e o valor de cada componente, distribuídos nos seguintes segmentos, que estão representados na Figura 5.3:

Bits 0 – 30: Valor do componente 3 (31 bits)
Bits 31 – 61: Valor do componente 2 (31 bits)
Bits 62 – 92: Valor do componente 1 (31 bits)
Bits 93 – 94: Tipo do componente 3 (2 bits)
Bits 95 – 96: Tipo do componente 2 (2 bits)

Bits 97 – 98: Tipo do componente 1 (2 bits)

Bits 99–104: Topologia do circuito (6 bits)

A cadeia de bits dividida nos segmentos descritos anteriormente é útil para facilitar a compreensão dos mecanismos de cruzamento e mutação empregados na classe Cromossomo. Para a representação do circuito, são propostos os seguintes atributos do cromossomo:

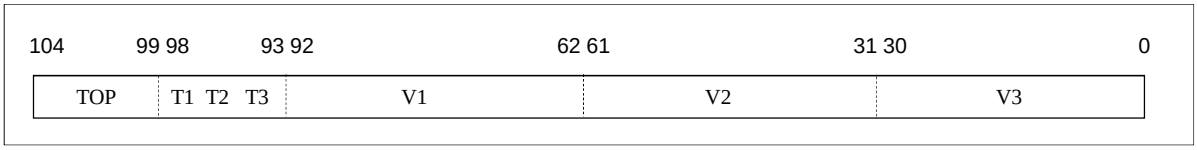


Figura 5.3 Representação binária para o cromossomo do AG sintetizador de filtros RLC.

5.3.1 Topologia do Circuito (CUANTLE, 2007)

Campo TOP: 6 bits(99-104) - bits top_5 a top_0, sendo

top_5-top_4 (Regra para inserção do componente 1):

00: Paralelo com a saída (ou seja, com o resistor R_L);

01: Paralelo com o elemento anterior (neste caso, v_e);

10: Série com a saída (ou seja, com o resistor R_L);

11: Série com o elemento anterior (neste caso, v_e).

top_3-top_2 (Regra para inserção do componente 2):

00: Paralelo com a saída (ou seja, com o resistor R_L);

01: Paralelo com o elemento anterior (neste caso, o componente 1);

10: Série com a saída (ou seja, com o resistor R_L);

11: Série com o elemento anterior (neste caso, o componente 1).

top_1-top_0 (Regra para inserção do componente 3):

00: Paralelo com a saída (ou seja, com o resistor R_L);

01: Paralelo com o elemento anterior (neste caso, o componente 2);

10: Série com a saída (ou seja, com o resistor R_L);

11: Série com o elemento anterior (neste caso, o componente 2).

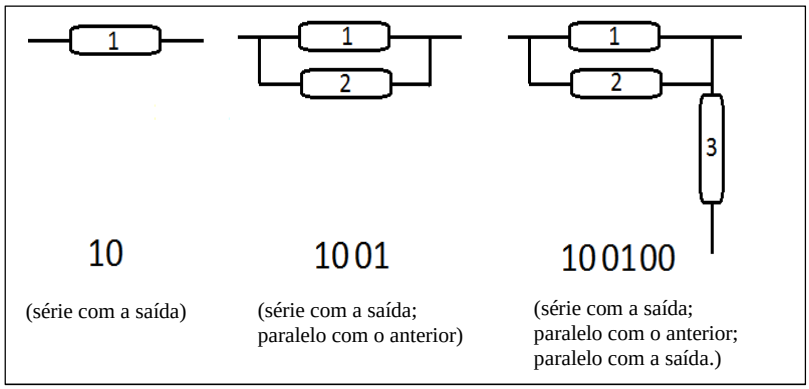
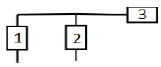
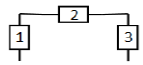
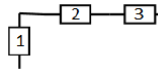
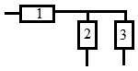
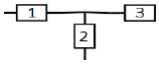
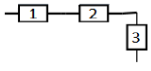


Figura 5.4 Exemplo de codificação para a topologia

Com o critério para codificação adotado, exemplificado na Figura 5.4, é possível obter as topologias apresentadas na Tabela V, onde os componentes 1, 2 e 3 podem ser resistor, capacitor, indutor ou elemento nulo (definição em 5.3.2). Observa-se, na coluna *Esquema* da tabela, o padrão de bits de cada topologia. O padrão contém sequências fixas de 0's e 1's intercaladas por asteriscos, que representam posições de irrelevância para a identificação da topologia.

Tabela V. Codificação das 15 topologias possíveis para o circuito com 3 componentes.

Índice	*Topologia*	*TOP(hex)*	*TOP(bin)*	*Esquema*
1		02 06 12 16	00 0010 00 0110 01 0010 01 0110	0*0*10
2		08 18	00 1000 01 1000	0*1000
3		0A 0B 1A 1B	00 1010 00 1011 01 1010 01 1011	0*101*
4		20 21 30 31	10 0000 10 0001 11 0000 11 0001	1*000*
5		22 32	10 0010 11 0010	1*0010
6		28 2C 38 3C	10 1000 10 1100 11 1000 11 1100	1*1*00

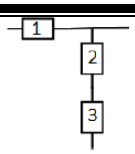
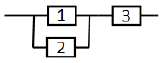
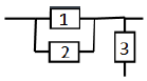
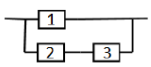
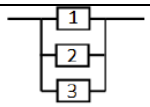
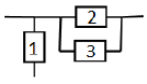
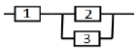
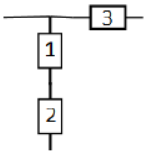
Índice	*Topologia*	*TOP(hex)*	*TOP(bin)*	*Esquema*
9		23 33	10 0011 11 0011	1*0011
10		26 36	10 0110 11 0110	1*0110
11		24 34	10 0100 11 0100	1*0100
12		27 37	10 0111 11 0111	1*0111
13		25 35	10 0101 11 0101	1*0101
14		09 19	00 1001 01 1001	0*1001

Tabela V. Codificação das 15 topologias possíveis para o circuito com 3 componentes (cont.)

Índice	*Topologia*	*TOP(hex)*	*TOP(bin)*	*Esquema*
7		2A	10 1010	1*1*1*
		2B	10 1011	
		2E	10 1110	
		2F	10 1111	
		3A	11 1010	
		3B	11 1011	
		3E	11 1110	
		3F	11 1111	
8		29	10 1001	1*1*01
		2D	10 1101	
		39	11 1001	
		3D	11 1101	

Índice	*Topologia*	*TOP(hex)*	*TOP(bin)*	*Esquema*
15		0E	00 1110	0*1110
		1E	01 1110	

5.3.2 Tipos de Componente

Campos T1, T2, T3: 6 bits (93 – 98) - 2 bits cada (t_1-t_0), sendo

t_1 t_0: tipo (natureza) de cada componente

00: resistor

01: capacitor

10: indutor

11: nulo (ramo em curto quando inserido em série ou ramo aberto quando inserido em paralelo)

5.3.3 Valores de Componente

Campos V1, V2, V3: 31 bits cada, faixa de valores: 0 a 2.147.483.647

Cada contagem no registro do valor de cada componente representa:

1 mΩ para os resistores

1 pF para os capacitores e

1 nH para os indutores

Assim, têm-se os seguintes valores máximos de representação:

Rmáx = 2147483647 x 1 mΩ =2 147 483 Ω = 2,147483 MΩ

Cmáx = 2147483647 x 1 pF = 2,1475 mF

Lmáx = 2147483647 x 1 nH = 2,1475 H

Para efeito de codificação em Java™ da classe Cromossomo, os agrupamentos de bits descritos acima (campos TOP, T1, T2, T3, V1, V2, V3) foram separados como atributos independentes da classe, todos do tipo inteiro, com escopo privado e nomenclatura apropriada para facilitar a compreensão do programa, conforme apresentado na Figura 5.5. Apesar desta separação, toda a manipulação no cromossomo, nos métodos *fazCruzamento* e *provocaMutação*, é feita como se este fosse uma cadeia única de 105 bits:

```
public class Cromossomo {
    private int codTopologia;
    private int tipoComponente1;
    private int valorComponente1;
    private int tipoComponente2;
    private int valorComponente2;
    private int tipoComponente3;
    private int valorComponente3;

}
```

Figura 5.5 Declaração em Java™ dos atributos da classe Cromossomo.

5.4 Nota de Avaliação para os Filtros

A função de avaliação (*fitness*) está implementada no método *calcular* da classe Avaliação, que por sua vez utiliza o método *analiseNodal* da mesma classe, cujo resultado é armazenado no atributo *notaAvaliacao* da classe Circuito, pois a classe Avaliação possui relacionamento unidirecional com a classe Circuito, ou seja, toda avaliação está relacionada com um circuito (instância da classe Circuito). Em outras palavras, um dos atributos da classe Avaliação é um objeto da classe Circuito, que representa o circuito avaliado. Isso foi feito em conformidade com os relacionamentos do diagrama de classes desenvolvido, que procura delegar à classe Avaliação a responsabilidade de avaliar um circuito, de forma a manter a execução da análise nodal confinada nesta classe e - através do recurso de reescrita da programação orienta a objetos - em suas classes filhas, garantindo ao desenvolvedor a facilidade de manutenção do código.

Ao se relacionar com a classe Circuito, a classe Avaliação pode solicitar à primeira informação sobre a topologia e os componentes do circuito, pois estes são atributos da classe Cromossomo, que por sua vez é atributo da classe Circuito. O relacionamento entre as classes

Circuito e Avaliação permite também a esta gravar o valor da avaliação através do método setNotaAvaliacao disponibilizado pela classe Circuito.

Para a execução deste método as classes filhas de Avaliacao, ou seja, Topologia0, Topologia1, Topologia2, até Topologia15, sobrescrevem o método AnaliseNodal da classe mãe para montar a matriz de análise nodal segundo a topologia definida pelo cromossomo do circuito e a frequência escolhida para o sinal de entrada. Esta quantidade de classes filhas de Avaliação se deve ao fato de que com a codificação escolhida para a topologia (vide 5.3.1) há 15 topologias possíveis para se construir um filtro passivo RLC com apenas três componentes. Além destas 15 topologias, há outras topologias logicamente possíveis, mas em todas elas obteremos v_s conectado a v_e pela resistência interna r_e e, assim, ganho aproximadamente 1 para qualquer frequência do sinal na entrada, o que resulta num circuito que não satisfaz os requisitos desejados para um filtro de sinais. Para estes casos a análise nodal atribui imediatamente valor 1 para o ganho do circuito independentemente da frequência considerada, o que resulta em baixa nota de avaliação para todos os tipos de filtro.

Para este trabalho, foi usada uma aplicação da expressão (3.3) para calcular a avaliação do circuito, de forma que a nota máxima seja 100:

$$A = \frac{100}{1 + \sum_{i=1}^{5} w_i \left| S(f_i) - O(f_i) \right|} \qquad (5.1)$$

Onde:

A: nota de avaliação do circuito;

i = ordem da frequência usada no cálculo (i=1, 2, 3, 4, 5), sendo:

$f_1 = 0{,}01f_c$ $f_2 = 0{,}1f_c$ $f_3 = f_c$ $f_4 = 10f_c$ $f_5 = 100f_c$;

w_i = peso do erro na frequência f_i;

$S(f_i)$ = ganho do circuito calculado para f_i;

$O(f_i)$ = ganho do circuito ideal para f_i.

O somatório do denominador de (5.1) corresponde à avaliação original apresentada em MURAKAWA (1998).

Desta forma, nota-se que a análise nodal calculada para a frequência de corte (f_c) é apenas uma etapa do procedimento de avaliação do circuito. Para a avaliação completa, o

algoritmo calcula o ganho do circuito, por Análise Nodal, para 5 frequências múltiplas da frequência de corte (f_1 a f_5). Essas frequências foram escolhidas de maneira a avaliar a função de transferência duas décadas antes e duas depois da frequência de corte, o que é razoável para o objetivo do trabalho, pois são poucos pontos de amostragem suficientemente abrangentes que não comprometem o desempenho do algoritmo (quanto mais pontos de amostragem, maior o tempo de execução). Os pesos foram tomados de maneira que na vizinhança da frequência de corte os pesos sejam bem maiores, pois esta região é a que define o comportamento do filtro. Assim, os pesos indicados em (5.2) se mostraram adequados para a função de avaliação, pois amplificam os erros da resposta em frequência calculados na região que vai desde uma década antes até uma década depois de f_c, penalizando os circuitos ruins nesta faixa de frequências, o que é essencial para avaliar a qualidade do filtro:

$$w_1 = w_5 = 1 \qquad\qquad w_2 = w_3 = w_4 = 10 \qquad\qquad (5.2)$$

5.5 Seleção dos Circuitos Genitores

A estratégia de seleção dos circuitos utilizada para sortear aqueles que participam dos cruzamentos foi a chamada Roleta Viciada (LINDEN, 2012), também denominada Seleção Proporcional (ZEBULUM, 2001). Trata-se, como visto em 3.4.1, de um sorteio em que a probabilidade de um circuito ser escolhido é diretamente proporcional à sua avaliação, o que acelera a convergência da solução, porém diminui a diversidade de circuitos. Como neste trabalho foi adotada a estratégia de elevar a taxa de mutação, a homogeneização da população é amenizada e o algoritmo se beneficia da rapidez da convergência. Os resultados apresentados na seção 5.7 foram obtidos com a estratégia da Roleta Viciada.

No diagrama de classes, pode ser observado que há outras formas de se aplicar a seleção de indivíduos para o cruzamento. Cada estratégia é construída nos métodos de uma classe filha da classe Seleção e as classes filhas podem ser desenvolvidas de forma independente, permitindo ao usuário selecionar a estratégia desejada para observar o impacto nos resultados.

Pelo conceito de probabilidade:

A probabilidade de ocorrer a escolha de um elemento do conjunto A (p_A), contido num espaço amostral S, é igual ao número de eventos favoráveis, ou seja, a quantidade de elementos de A (N_A), dividido pelo número de eventos possíveis, ou seja, a quantidade de elementos de S (N_S):

$$p_A = \frac{N_A}{N_S} \tag{5.3}$$

Para o caso da Roleta Viciada, o número de eventos favoráveis para a escolha de um circuito corresponde ao valor da sua avaliação e o número de eventos possíveis, à soma de todas as avaliações da população. Em outras palavras, a probabilidade de um circuito ser escolhido é numericamente igual à fração de contribuição da avaliação deste circuito no total das avaliações da população. Assim, a probabilidade de qualquer circuito da população ser sorteado nesta estratégia pode ser expressa da seguinte forma:

$$p_k = \frac{A_k}{\sum_{i=1}^{n} A_i} \qquad 1 \le k \le n \tag{5.4}$$

Onde

p_k : Probabilidade do circuito k ser sorteado dentre todos da população;

A_k : Valor da avaliação do circuito k;

$\sum_{i=1}^{n} A_i$: Soma das avaliações de todos os circuitos da população

n : Tamanho da população.

Para a realização do sorteio, executa-se o método *selecionar* da classe RoletaViciada, cujo algoritmo é o que foi apresentado na seção 3.4.1

5.6 Aplicando os Operadores Genéticos

5.6.1 Cruzamento

A técnica de cruzamento empregada foi a de três pontos de corte, em que há necessariamente cruzamento nos três segmentos principais do cromossomo, ou seja, topologia, tipos e valores dos componentes de cada cromossomo selecionado para participar da geração de novos indivíduos. Os pontos de corte são escolhidos por sorteio de um valor inteiro aleatório entre 1 e a quantidade de bits menos 1 de cada segmento.

Existem na literatura descrições de outros métodos de cruzamento, como a de ponto de corte único, porém, devido às características particulares do cromossomo desta aplicação, a técnica de ponto único seria ineficiente, pois, estatisticamente, a maioria dos pontos de corte cairia sobre os segmentos referentes ao valor dos componentes, visto que estes segmentos formam a maior parte do cromossomo (93 bits de um total de 105 bits = 88,57%) e assim os cruzamentos explorariam muito pouco as variações de tipos de componente e topologias, ficando fortemente concentrados apenas nos valores dos componentes.

Outra técnica descrita na literatura é a de cruzamento uniforme (seção 2.5.1 – Figura 2.2): uma sequência de bits auxiliar do tamanho do cromossomo é gerada aleatoriamente e as posições nesta sequência em que houver bit 1, o filho recebe o valor do bit correspondente do primeiro genitor e onde houver bit 0, o filho recebe o valor do bit correspondente do segundo genitor. Esta técnica tende a gerar maior diversidade à população do que se pode obter com a implementação de um único ponto de corte por segmento, visto que cada alternância entre 0 e 1 na sequência de bits auxiliar resulta um novo ponto de corte. Isso significa maior potencial de destruição de esquemas, pois quanto mais pontos de corte maiores são as chances de se quebrar esquemas e, desta forma, prejudicar a herança de boas características dos pais. Além disso, possui a desvantagem de ser custosa em processamento devido ao tamanho do cromossomo, 105 bits, requerer muita manipulação de bits. Por essas razões, essa técnica de cruzamento não foi adotada neste trabalho.

Voltando à representação binária do cromossomo e à técnica de três pontos de corte, a Figura 5.6 apresenta os três segmentos do cromossomo (S_1, S_2 e S_3) que realizam o cruzamento:

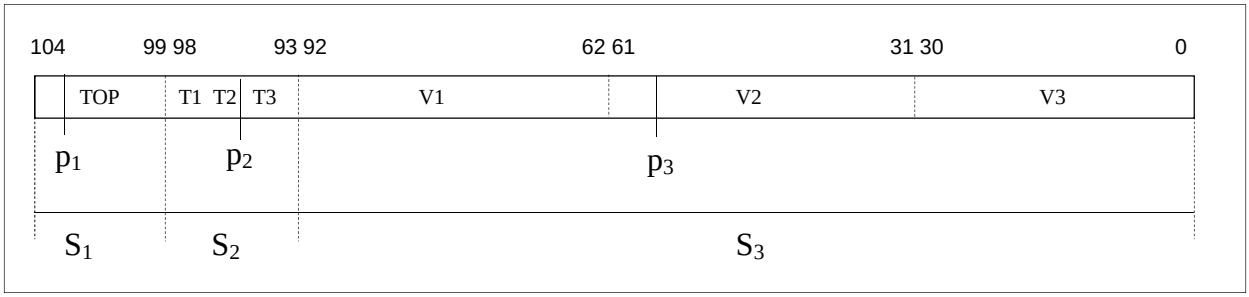


Figura 5.6 Segmentos e pontos de corte utilizados no cruzamento dos cromossomos.

p_1: ponto de corte para a topologia

S_1: segmento da topologia (bits 99 a 104)

p_2: ponto de corte para tipos

S_2: segmento dos tipos (bits 93 a 98)

p_3: ponto de corte para os valores

S_3: segmento para os valores (bits 0 a 92)

A Figura 5.7 apresenta um exemplo do cruzamento para o segmento S_1, entre a topologia (6 bits) do pai1 (0x17=23) e a topologia do pai2 (0x0E=14), sorteando o ponto de corte de valor 4 (quarto bit). O filho gerado terá, portanto, valor de topologia igual a 0x1E (=30).

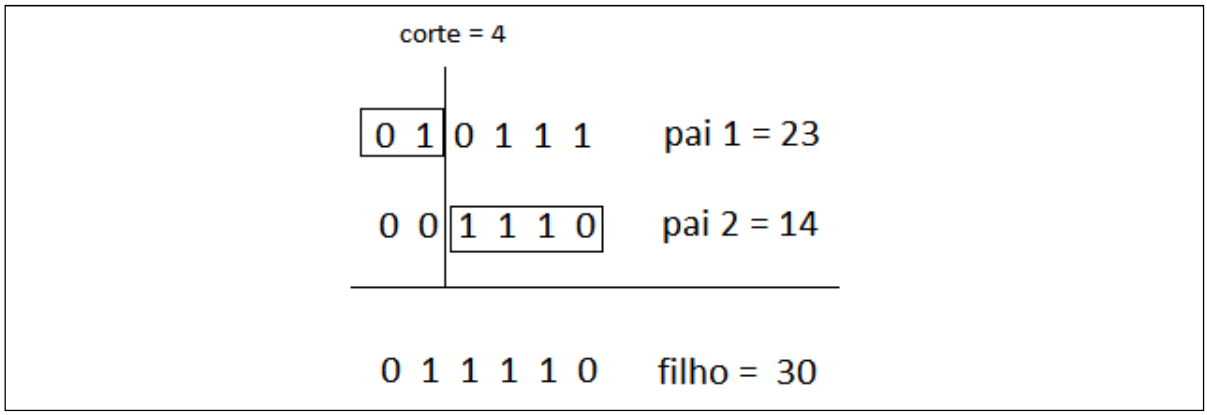


Figura 5.7 Exemplo de cruzamento entre as topologias de dois circuitos

Analogamente, o processo se repete para os outros segmentos (S_2 = T1+ T2+ T3 e S_3 = V1+V2+V3), considerando a quantidade de bits de cada segmento (6 bits para os Tipos e 93 bits para os Valores), de modo a limitar a faixa permitida aos pontos de corte.

O cruzamento é, portanto, um processo que, a partir dos cromossomos dos pais, gera o cromossomo do filho e tem a finalidade de gerar indivíduos com novas características ou, no contexto deste trabalho, circuitos novos para concorrerem à melhor solução do problema.

5.6.2 Mutação

A literatura sobre AGs recomenda o uso de uma taxa de mutação pequena: empregada aleatoriamente a uma parcela de 1% a 5% dos indivíduos gerados no cruzamento. Em algumas aplicações, porém, taxas mais elevadas de mutação proporcionam melhor desempenho do algoritmo. Liu (2009) menciona o termo hipermutação, onde se aplicam taxas de mutação acima de 10%. Para a execução deste trabalho foi constatado experimentalmente que, aumentando a taxa de mutação, a população se tornava mais diversificada, o que favoreceu a

busca do circuito ideal. Lembrando que o método de seleção empregado foi o da Roleta Viciada, que acelera a convergência, porém tende a tornar a população homogênea.

A taxa de mutação utilizada neste trabalho foi de 30%, ou seja, em média, a cada 10 circuitos gerados pelo cruzamento, 3 deles são modificados aleatoriamente com o intuito de se encontrar um circuito com novas características que possa ter boa avaliação, o que experimentalmente se mostrou eficaz em manter alguma diversidade na população mesmo após milhares de gerações, sem perder a similaridade cromossômica entre a maioria dos indivíduos, o que é indício de convergência do algoritmo. O operador de mutação utilizado foi o uniforme, exemplificado na Figura 5.9, ou seja, a cada segmento do cromossomo foi aplicada uma máscara que altera o valor de alguns bits, determinados pelo valor da máscara. Em comparação com a mutação simples, que altera apenas um bit, a mutação uniforme gera alterações muito mais significativas no cromossomo, garantindo assim uma ampla exploração do espaço de busca sem elevar o custo de processamento, visto que a operação de OU-exclusivo é de fácil codificação no *software*. Em contrapartida, quanto mais modificado o cromossomo, maior a possibilidade de se perder as boas características obtidas nos cruzamentos, pois os bits responsáveis pelas boas características do indivíduo tendem a ser substituídos por bits aleatórios provenientes da mutação.

Apesar deste aspecto negativo do uso da mutação uniforme, ela foi adotada neste trabalho porque ao longo das gerações as boas características do cromossomo tendem a predominar na população, conforme prevê a teoria dos esquemas, descrita por Holland (1975), e, assim, apenas uma minoria dos indivíduos (os que sofrem mutação) estão sujeitos a perderem seus genes positivos em busca de mais genes vantajosos, enquanto que a maioria da população continuará apresentando as boas características obtidas pelos cruzamentos sucessivos.

A teoria dos esquemas, vista na seção 2.7.3, afirma que um determinado padrão de bits acaba prevalecendo na representação cromossômica dos melhores indivíduos após muitas gerações (HOLLAND, 1975). Este padrão, ou esquema, é representado por uma sequência de valores 0, 1 e *, em que * representa uma posição em que se pode ter 0 ou 1, indistintamente. Os padrões para a topologia do circuito são apresentados na coluna *Esquema* da Tabela V.

Na Figura 5.8, estão indicadas as máscaras utilizadas: m_1 (de 6 bits) para a topologia, m_2, m_3 e m_4 (de 2 bits cada) para os tipos de componente e m_5, m_6 e m_7 (de 31 bits cada) para os valores de componente.

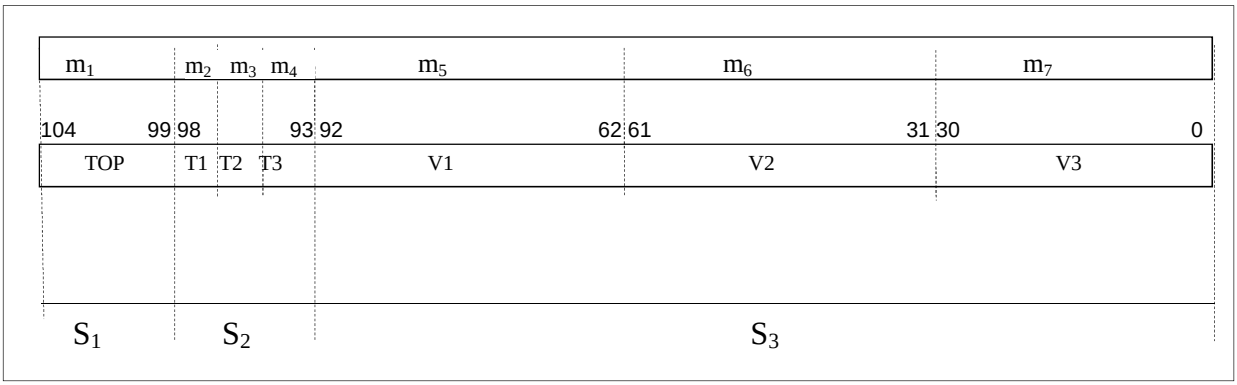


Figura 5.8 Segmentos (S_1 a S_3) e máscaras (m_1 a m_7) utilizados na mutação do cromossomo.

O indivíduo que sofre mutação recebe o resultado da operação OU-exclusivo (XOR) entre a máscara de mutação e o campo do cromossomo. Assim, por exemplo, um cromossomo com topologia de valor 0x1E, que sofre mutação com m_1=0x32, terá o novo valor de topologia igual a 0x2C (= 0x1E XOR 0x32). Analogamente, o processo se repete com os outros segmentos.

A Figura 5.9 apresenta a operação de mutação do exemplo descrito acima.

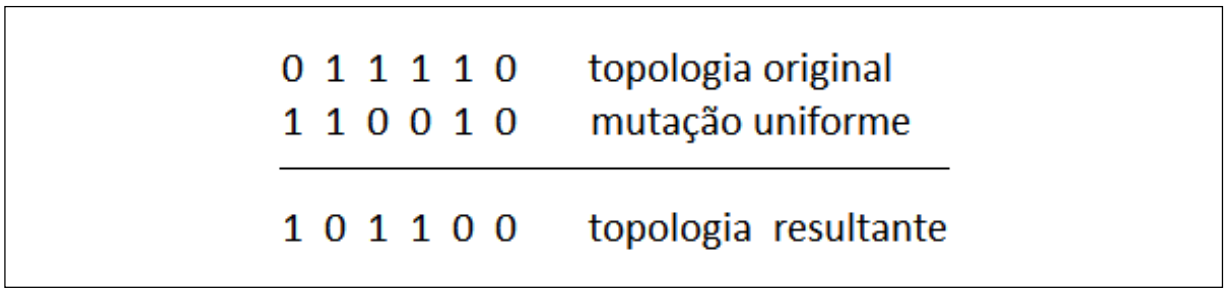


Figura 5.9 Exemplo da mutação uniforme aplicada ao cromossomo da topologia com o operador OU-exclusivo.

5.6.3 Elitismo

Com o intuito de não se correr o risco de perder uma boa solução com o passar das gerações, foi implementada a estratégia de elitismo de tamanho 1, ou seja, o grupo das melhores soluções tem apenas um único indivíduo, que visa garantir que a melhor solução da geração n permaneça na geração n+1. Esta opção pelo tamanho mínimo do grupo de elite se deve à simplicidade de sua execução e por ser suficiente para não haver retrocesso na busca pela melhor solução. Desta forma, o melhor pai de uma geração substitui o pior filho gerado no grupo seguinte. Esta técnica foi implementada diretamente no módulo principal do projeto,

simplesmente chamando os métodos *getPior* e *getMelhor* da classe População para se ter acesso à pior e à melhor solução.

5.7 Resultados e Análises

A seguir são apresentados, na Tabela VI, os resultados obtidos em três execuções do algoritmo genético para cada um dos quatro tipos de filtro com frequência de corte 1 kHz. A execução é finalizada quando o melhor circuito possui nota maior ou igual a 99,0, ou quando se atinge o número de 1.000.000 gerações. As simulações de verificação foram realizadas com o aplicativo 5SPICE Analysis v2.10.0 (ANDRESEN, 2013). Na Figura 5.10 são apresentadas as curvas de magnitude do ganho, em dB, em função da frequência do sinal de entrada, em Hz, correspondentes às execuções listadas na Tabela VI para os filtros passa-baixa, passa-alta, passa-faixa e rejeita-faixa, respectivamente.

Tabela VI. Resumo dos filtros obtidos com o programa proposto.

Tipo	*Exec*	*Tempo*	*Gerações*	*Top.* índice*	*re (Ω)*	*Componentes 1*	*2*	*3*	*Freq f_c*	*NOTA*
PB	1	0m39s	57 490	2	50	C=3,09µ	L=2,15H	C=7,16m	1kHz	99,29
PB	2	9m18s	1 000 000	4	50	L=132m	C=459µ	-	1kHz	96,92
PB	3	9m30s	1 000 000	11	50	-	L=133m	C=455µ	1kHz	96,92
PA	1	4m49s	475 151	9	50	C=4,77µ	L=5,96m	C=438µ	1kHz	99,00
PA	2	10m09s	1 000 000	9	50	C=5,90µ	L=7,40m	C=363µ	1kHz	98,53
PA	3	0m06s	429	11	50	C=4,76µ	L=0,58H	L=6,33m	1kHz	99,00
PF	1	0m35s	62 085	1	50	C=209µ	L=119µ	R=2,15M	1kHz	99,02
PF	2	5m09s	584 971	1	50	L=38,9µ	C=648µ	L=1,68H	1kHz	99,00
PF	3	6m44s	490 979	2	50	C=537µ	C=105n	L=0,24H	1kHz	99,20
RF	1	0m08s	4 289	15	50	L=2,67m	C=9,45µ	C=1,26m	1kHz	99,31
RF	2	0m07s	2 025	15	50	L=819µ	C=30,7µ	L=1,41H	1kHz	99,07
RF	3	0m19s	25 240	11	50	L=0,52H	C=48,7n	R=285K	1kHz	99,78

* Consultar na Tabela V a topologia correspondente ao índice indicado nesta tabela.

A execução do algoritmo gerou corretamente todos os circuitos solicitados, o que demonstra que o AG é uma ferramenta eficiente para o desenvolvimento de circuitos analógicos que pode ser estendida para outras aplicações, além de filtros, e com maior número de componentes. O uso de simuladores, em conjunto com a computação paralela, na obtenção das avaliações aumentaria muito o potencial do AG, pois reduziria o código, uma vez que não haveria métodos para análise de circuito, e o tempo de execução, visto que as avaliações seriam obtidas simultaneamente para um conjunto de indivíduos da população.

Observa-se da resposta em frequência do filtro passa-faixa (PF), visto na Figura 5.10, que para sinais de entrada com frequência menor do que 0,2 kHz e maior do que 5 kHz, o ganho está abaixo de -60 dB. Isso significa que menos de 0,1% da amplitude do sinal da entrada é detectado na saída do circuito. Em outras palavras, o sinal é praticamente filtrado nessas frequências. Nas proximidades de 1 kHz o ganho é aproximadamente 0 dB, o que significa que não há atenuação de amplitude do sinal nesta frequência de corte. A faixa estreita para a banda de passagem indica que o circuito tem nota alta.

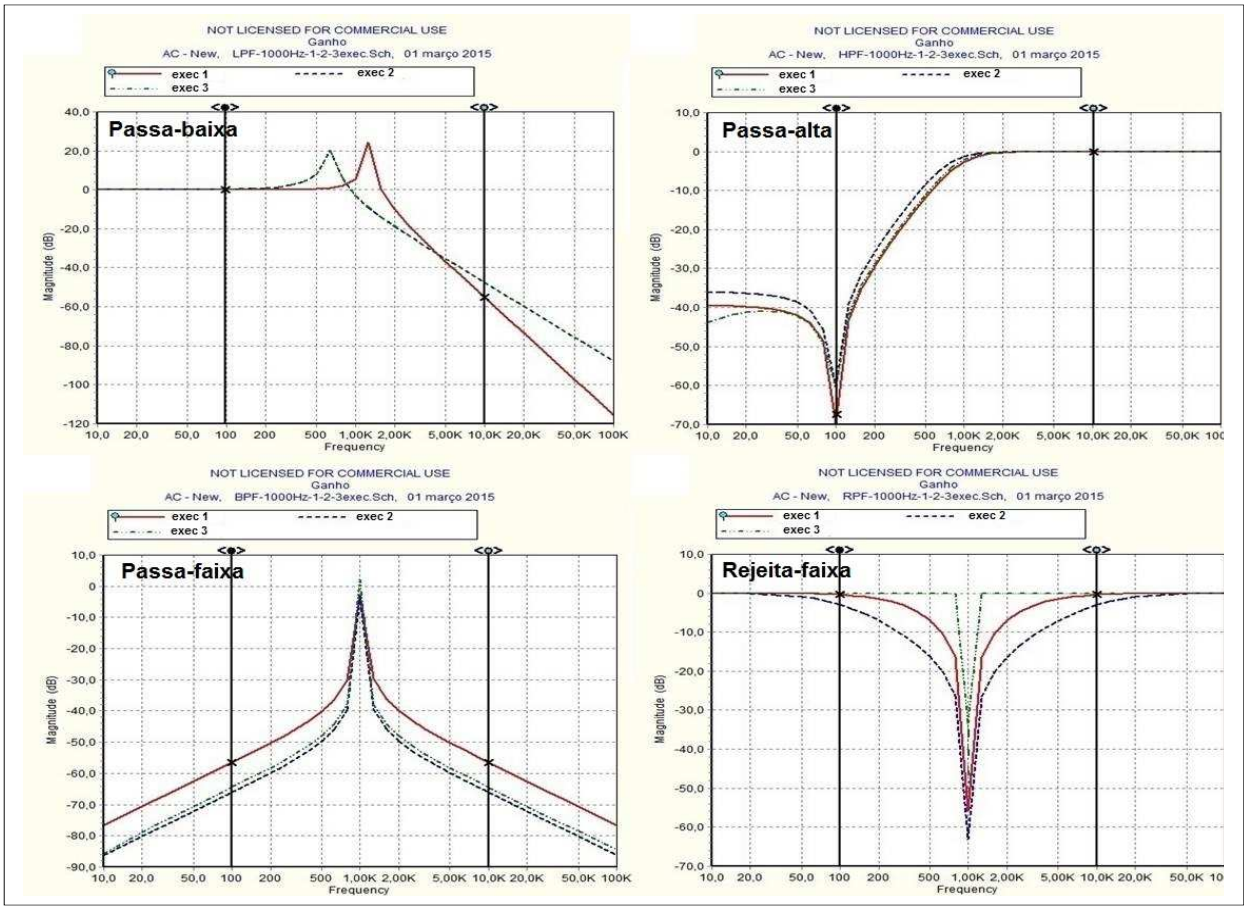


Figura 5.10 Resposta em frequência dos circuitos listados na Tabela VI.

Uma análise similar pode ser feita para as outras categorias de filtro. O filtro rejeita-faixa (RF) é complementar ao filtro PF, rejeitando sinais na entrada em torno de 1 kHz. O filtro passa-alta (PA) rejeita sinais de entrada com frequência menor do que 1 kHz, mais acentuadamente para frequências abaixo de 0,2 kHz. Pode-se observar que para 100 Hz, ou $0{,}1f_c$, o ganho do filtro PA está abaixo de -60 dB. O filtro passa-baixa (PB) rejeita sinais com frequência maior do que 1 kHz. Pode ser verificado que a inclinação negativa da curva do filtro PB poderia ser mais acentuada, o que resultaria em notas mais altas para o circuito, se a

resistência de entrada do circuito fosse menor do que 50 ohms, pois quanto menor r_e, menor a perda por inserção do filtro (ORSINI, 1991).

6. CONCLUSÃO E CONSIDERAÇÕES FINAIS

Este trabalho de pesquisa realizou o desenvolvimento de uma aplicação utilizando Algoritmo Genético (AG) com abordagem orientada a objetos que sintetiza automaticamente filtros com até três componentes passivos, a partir de requisitos de projeto inseridos pelo usuário, como a frequência de corte e a faixa de rejeição.

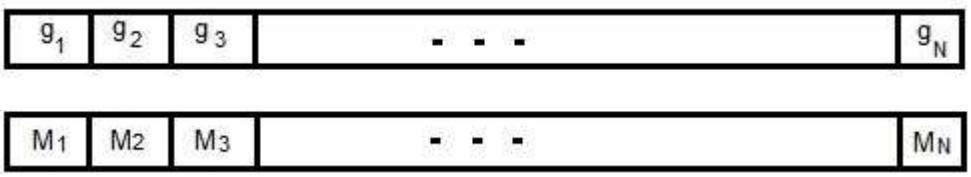
Para atingir esse objetivo, foi apresentado inicialmente o contexto em que os AGs estão inseridos, começando pela Teoria da Evolução e sua relação com os Algoritmos Evolucionários (AEs). No capítulo seguinte, as técnicas de algumas categorias de AE foram apresentadas, com destaque para o AG, que no capítulo 3 é apresentado com abordagem orientada a objetos, denominado Algoritmo Genético Orientado a Objetos (AGOO), como forma de modelar o problema real em termos de classes e diagramas UML (*Unified Modeling Language*), usando a linguagem Java™ para o desenvolvimento da aplicação. No capítulo 4, os conceitos e equações relacionados aos filtros passivos e o estudo da resposta em frequência dos circuitos foram discutidos como forma de análise do problema a ser tratado, tarefa imprescindível para as escolhas mais adequadas para a representação e a função de avaliação das soluções geradas pelo algoritmo. Finalmente, no quinto capítulo, o AGOO apresentado de forma genérica no capítulo 3, e considerando a teoria de filtros apresentada no capítulo 4, foi adaptado para sintetizar filtros passivos a partir de alguns requisitos de projeto fornecidos pelo usuário. Nesse capítulo, ainda, foram apresentados os resultados obtidos com o AGOO desenvolvido, que confirmaram a viabilidade do uso de um AG como sintetizador automático de circuitos.

Algumas observações e comentários são expostos a seguir como forma de destacar as possibilidades de desenvolvimento dos AGs a partir da aplicação que foi desenvolvida neste trabalho.

6.1 Representação com Tamanho Variável

A representação do circuito adotada pode ser adaptada para uma quantidade maior de componentes e já está compatível com a chamada VLR (*Variable Length Representation*), ou representação de tamanho variável, sem necessitar de uma máscara de ativação, como a que foi proposta por Zebulum (2001), que indica os componentes presentes e os ausentes no circuito final. A Figura 6.1a ilustra a VLR com máscara de ativação (ZEBULUM, 2001) e a Figura 6.1b, a VLR sem máscara auxiliar, como proposta neste trabalho. Os elementos g_i são os componentes do circuito e os elementos M_i são os ativadores, cujo valor é 0 ou 1, para os

componentes. Se $M_k = 0$, o componente g_k não é inserido no circuito. Já na representação usada neste trabalho, os próprios componentes g_i podem se excluir do circuito, caso seu tipo seja “nulo” (apenas os componentes cujo tipo for R, L ou C são inseridos no circuito).



a) VLR com máscara: o tamanho é a quantidade de $M_i \neq 0$ (ZEBULUM, 2001)

b) VLR sem máscara: o tamanho é a quantidade de $g_i \neq$ NULO

Figura 6.1 Representações com tamanho variável (VLR).

A representação proposta contempla o elemento nulo e, assim, a quantidade de elementos nulos define indiretamente o tamanho do circuito, uma vez que a quantidade de componentes no circuito (L) é dada pela diferença entre a quantidade máxima possível (N) e a quantidade de nulos (Q_{nulos}) (Zebulum, 2001), conforme a equação 6.1.

$$L = N - Q_{nulos} \tag{6.1}$$

6.2 Abordagem com Orientação a Objetos

As classes e subclasses propostas no modelo orientado a objetos foram definidas de forma que as etapas de execução do AG pudessem ser desenvolvidas de forma independente, como blocos funcionais intercambiáveis. Desta forma, a classe Seleção, por exemplo, pode executar o método de Roleta Viciada, ou Torneio, ou *Ranking*, bastando que o desenvolvedor faça a instância do objeto da classe escolhida no módulo principal da aplicação. Pode ser adotada também uma solução combinada entre as técnicas de seleção, instanciando dois ou mais objetos, cada um sendo um representante de cada classe filha de Seleção, para executar a técnica de seleção de cada uma, para então realizar a combinação. Analogamente, o mesmo

pode ser feito com a classe Cromossomo, que pode ter classes filhas que codificam representações distintas da que foi definida na classe mãe e, ainda, executar diferentes métodos de cruzamento e mutação. A classe Avaliação, é claro, também pode ter suas variações em classes filhas, que fariam com que o circuito fosse avaliado segundo outros critérios, e assim a aplicação se adaptaria para a síntese de outros circuitos, não apenas de filtros. Como cada etapa do algoritmo foi separada em classes distintas, com seus respectivos atributos e métodos, a flexibilidade para manutenção e adaptação do código para realizar outras tarefas é uma vantagem considerável para esse tipo de abordagem na construção do AG.

6.3 Aplicabilidade dos AGs

Os AGs devem ser entendidos como uma ferramenta extra para a resolução de problemas, para ser usada quando não se conhece uma solução analítica, ou um algoritmo dedicado, para o problema que se deseja solucionar. Não existe um AG universal que sirva para qualquer aplicação. Naturalmente, é possível, a partir de um AG genérico, já pronto, mudar sua função de avaliação e obter resultados interessantes, mas esta não é a melhor abordagem (LINDEN, 2012). A simplicidade e a flexibilidade para adaptação do AG não isenta o desenvolvedor de analisar profundamente o problema antes de começar a construir o algoritmo. É importante escolher uma representação para a solução e uma função de avaliação que tenham a capacidade de embutir o máximo de conhecimento do problema, sem as quais dificilmente o algoritmo fornecerá soluções satisfatórias. O AGOO apresentado no capítulo 3 deste trabalho de pesquisa propõe um modelo de estrutura para o *software*, que é válido para diferentes aplicações, mas o conteúdo, ou seja, o código escrito em cada classe, é um trabalho de desenvolvimento específico para cada aplicação.

Apesar de todas as suas limitações, inerentes a qualquer algoritmo de busca, o AG pode ser considerado uma técnica bastante viável para solução de problemas, uma vez que o avanço constante dos recursos computacionais disponíveis, como a multiplicidade de núcleos de processamento, as taxas de transmissão de dados na rede mundial em Gbps, a programação na nuvem, entre outros, tornam os algoritmos evolucionários muito mais competitivos atualmente do que eram há duas ou três décadas.

6.4 Possibilidades de Exploração dos AGs em Trabalhos Futuros

Desde a concepção dos primeiros algoritmos evolucionários, nos anos 1950, muitas técnicas computacionais de imitação da teoria da evolução das espécies foram desenvolvidas, dando origem às categorias de AEs, como os AGs, descritas no capítulo 2. Atualmente, as inúmeras possibilidades de exploração dos operadores genéticos e dos parâmetros de execução (tais como, tamanho da população, taxa de ocorrência de cruzamento e mutação, tamanho do grupo de elite, tamanho da representação cromossômica, fator de pressão seletiva, etc.) permitem a criação de algoritmos genéticos mais complexos e, em contrapartida, mais eficientes.

Michalewicz (2004) apresenta algumas técnicas a serem exploradas na execução do AG, ou de qualquer algoritmo evolucionário, visto que a combinação de técnicas entre os algoritmos evolucionários faz com a distinção entre suas categorias não sejam mais tão claras. Essas novas técnicas são descritas a seguir.

Considerando que o AG se inspira na ideia de evolução, por que não o próprio algoritmo se submeter à evolução de si mesmo, ou seja, do seu próprio desempenho? Assim seria natural imaginar que não apenas as soluções procuradas evoluíssem, mas também os parâmetros que controlam o desempenho do AG para cada problema específico. Durante a execução, o algoritmo levaria em conta o desempenho corrente de seu processo de busca para alterar seus parâmetros (MICHALEWICZ, 2004), ajustando o tamanho da população, a taxa de mutação, o método de seleção, etc., com o intuito de obter melhores resultados, ou seja, mais soluções bem avaliadas, mantendo a diversidade da população.

No modelo para o AGOO apresentado no capítulo 3, esta técnica poderia ser adotada com a inclusão da classe AG, cujos atributos seriam os parâmetros do algoritmo e os seus métodos executariam operações sobre esses atributos, com o objetivo, por exemplo, de aumentar a diversidade da população ou a velocidade de convergência das soluções. Esta classe poderia ter uma relação de agregação com a classe População, conforme é visto na representação UML da Figura 6.2, de modo que a classe População seja um atributo da classe AG. Notar que alguns atributos, que anteriormente eram da classe População, passariam a pertencer à classe AG, como o tamanho da população e as taxas de cruzamento e mutação. Todas as demais classes e seus relacionamentos permaneceriam como no diagrama original da Figura 3.9.

Outra possibilidade é criar um AG dinâmico, no sentido de que contemple mudanças nos requisitos do projeto durante a sua execução. Seria o equivalente na natureza às mudanças

repentinas no meio ambiente, impactando a sobrevivência das espécies que nele vivem. Isso significaria que a função de avaliação pode mudar a qualquer momento sem a necessidade de reiniciar o algoritmo, pois os AGs são essencialmente adaptativos: qualquer mudança na função de avaliação causaria certamente impacto nas avaliações da população, ou seja, algumas soluções anteriormente bem avaliadas teriam suas notas rebaixadas devido à mudança nos critérios de avaliação, mas depois de algumas gerações surgiriam outras soluções com boa avaliação segundo os novos critérios.

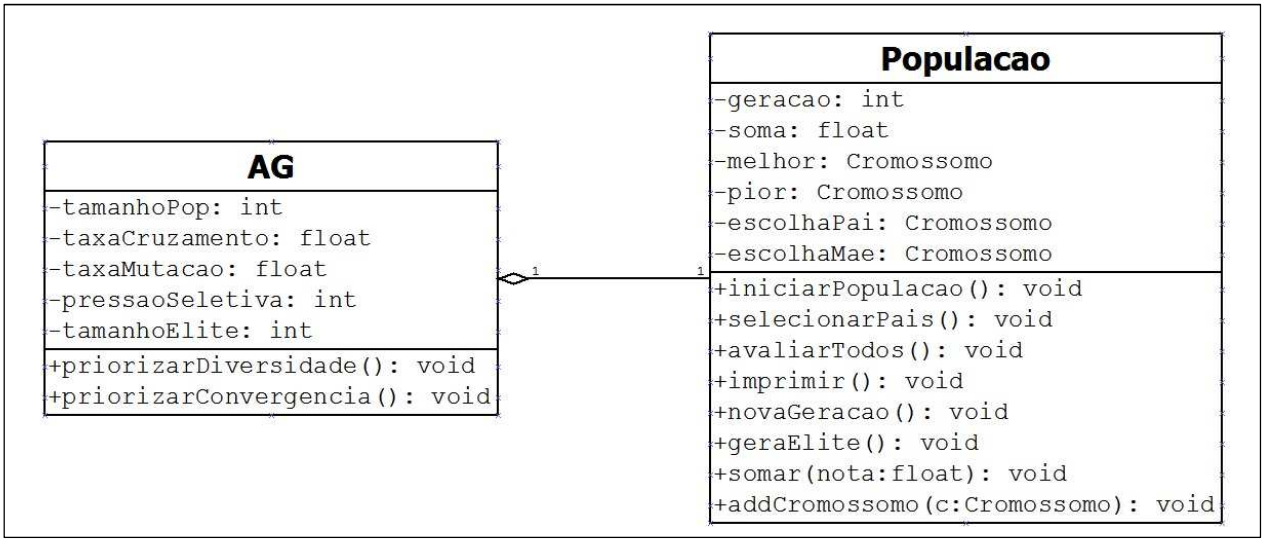


Figura 6.2 Relacionamento entre as classes AG e População para evolução de parâmetros.

De acordo com Michaeliwcz (2004), o desenvolvedor pode criar os mecanismos de evolução de seu algoritmo evolucionário como um criador da natureza e determinar as regras de simulação do algoritmo da mesma forma que faria com a física no mundo real. Assim, é possível violar as leis da natureza e introduzir, por exemplo, o Lamarckismo, permitindo que as soluções incorporem características adquiridas durante o seu ciclo de vida, com a possibilidade de transmiti-las aos seus descendentes. Em outras palavras, a representação cromossômica de uma solução pode receber parte do cromossomo de uma boa solução a partir de um novo operador genético - a aquisição – e ainda ser submetida ao cruzamento e à mutação.

Uma violação da concepção natural, mas que é perfeitamente possível no ambiente da simulação computacional, seria o operador de cruzamento com mais de dois genitores, gerando mais do que dois descendentes (MICHAELIWCZ, 2004). O objetivo desta técnica é elevar o grau de diversidade da população a partir dos cruzamentos.

O AG é, portanto, uma simulação da evolução natural que não tem compromisso em modelar apenas fenômenos observados na natureza, o que abre possibilidades ilimitadas para o desenvolvimento de algoritmos evolucionários mais eficientes.

APÊNDICE A – Artigos Publicados

A.1. Conferência IEEE ICM2014 Doha (Qatar) - dez/2014

Synthesis of Passive Filter using Object Oriented Genetic Algorithm

Orlando Verducci Jr, Paulo C Crepaldi, Leonardo B Zoccal, Tales C Pimenta
Universidade Federal de Itajubá
Itajubá – MG BRAZIL

***Abstract*— This paper describes the development of an evolutionary algorithm, the use of genetic algorithm to automatically synthesize analog circuits. The context of the project is the development of passive RLC filters of up to three components, by choosing cutoff frequency and type of filters (low pass, high pass, band pass or notch). The evaluation of each solution was performed by calculating the circuit voltages by nodal analysis for the various possible topologies without the use of simulators or programmable hardware. The genetic algorithm was fully developed on object-oriented language, Java, from a class diagram that shows the relationships between population, individual (candidate circuit), chromosome (genetic representation of the circuit), selection method, crossover, mutation, evaluation of the individual (quality of the circuit), among other classes.**

Keywords*— *Genetic Algorithm, Evolutionary Algorithm, Passive Filter, Java, Class, Object.

I. INTRODUCTION

Genetic Algorithm – GA is an evolutionary algorithm, inspired by nature, that is used to find solutions for a given problem [1]. More precisely, a GA models computationally, the phenomena involved in the reproduction of living beings (crossover and mutation) and the natural selection of the organisms most adapted to their environment, over many generations. Each generation of individuals is obtained after one GA cycle, as shown in Figure 1 [2].

The design of an electrical circuit is such a problem that can be automated with the use of a GA, whereas its implementation consists of finding circuits that meet the design requirements among a large number of combination of topologies, components and their possible values.

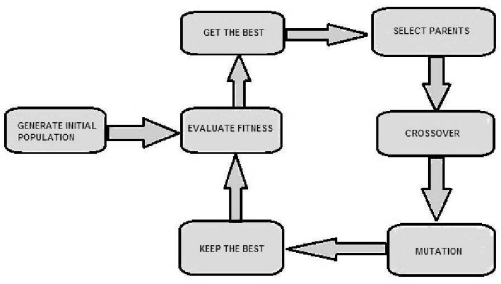


Fig. 1. GA execution cycle.

In a GA, each individual is a candidate solution to the proposed problem. The set of candidate solutions is a population. The environment in which the individuals must survive or adapt is represented by the criteria of assessment of the individuals as a solution of the problem [3].

At the same time, object oriented languages, such as Java, are very easy to use. Therefore it is quite simple to implement any circuit topology, such as filters. Additionally, based on nodal equations, the circuits can be evaluated without the use of simulators, such as SPICE. As a consequence, circuits modeled by nodal equations can be quickly evaluated over time or frequency, without the use of massive computational effort or time, and nevertheless, it is possible to achieve the proper circuit behavior.

The goal of this work is to obtain automatically, by using an object-oriented design of GA, the topology (components interconnection tree), the type of component (resistor, capacitor or inductor) and their values (in ohms, farads or henries) so that the circuit implements an electronic signal filter. The filter can have up to three components and must meet the desired frequency cutoff and the type of filter (low pass, band pass, high pass and notch).

This article is organized as follows. Section II describes the GA applied to the filter, Section III describes the results and finally Section IV presents the conclusions.

II. METHODOLOGY

Figure 2 presents the structure used to evaluate the evolutionary circuit. The structure is comprised of the sinusoidal input signal v_i = 1cos(ωt+0°) V, the input resistance r_i=50 Ω, the RLC circuit to be synthetized and the load resistance R_L=100 MΩ. The external components were chosen to provide an AC input signal and a high load impedance to the synthetized RLC circuit [4].

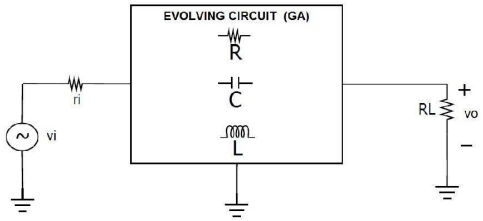


Fig. 2. Verification circuit, including the filter to be synthetized.

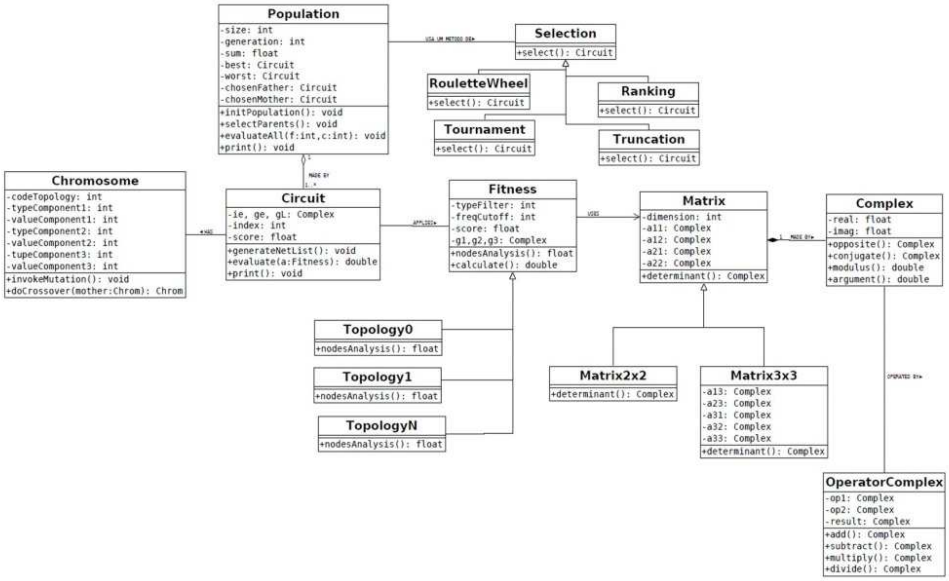


Fig. 3. Class diagram of an object-oriented GA for circuit synthesis.

Figure 3 presents the class diagram used as model to develop the GA to synthetize the passive filter. The GA was developed using the object-oriented Java language to allow flexibility for the users who wish to modify the parameters, such as parenting selection, chromosome representation of the circuit, and others, in order to compare the results. Each GA technique was built in separate classes as can be seen in the class diagram.

The circuit to be designed is represented by a chromosome containing the topology segments (6 bits), the types of components (6 bits) and value fields (31 bits for each component), as shown in Figure 4.

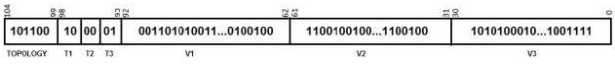


Fig. 4. Chromosome segments of a three component RCL circuit.

The topology encoding reserves two bits for each component of the circuit, as shown in Table I.

TABLE I. CODING OF TOPOLOGIES

Code	Meaning	
	Insertion	*Referred to*
00	Parallel	Output
01	Parallel	Previous component
10	Series	Output
11	Series	Previous component

As an example, similar to [5], the topology encoding 100100 corresponds to the sequence of inserts 10 (in series to the output), 01 (in parallel to previous one) and 00 (in parallel to the output), as shown in Figure 5. The component encoding (two bits each) is given by Table II.

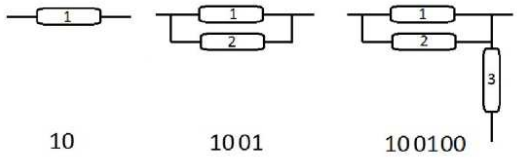


Fig. 5. Insertion sequence for the 100100 encoding sequence.

Thus, as an example, the component encoding 011000 corresponds to the insertion of a capacitor (01), followed by an inductor (10) and, finally a resistor (00) in the insertion sequence defined by the topology code. The null component (11) is considered a short-circuit when inserted in series and an open branch when inserted in parallel.

TABLE II. CODING OF COMPONENTS

Code	Meaning	
	Component	*Unit*
00	Resistor	<value> ($x10^{-3}$) ohm
01	Capacitor	<value> ($x10^{-12}$) farad
10	Indutor	<value> ($x10^{-9}$) henry
11	Null	-o-

From an initial population of 100 individuals (circuits) randomly generated, it is started the cycle of evaluations, crossovers, mutations and preservation of the fittest (elitism), until it is found the circuit with sufficiently high evaluation. The evaluation is performed by the *NodesAnalysis*(*int f*) method of the Fitness class, where *f* (the integer value of the input signal frequency) is the only one parameter needed in this method. The crossovers and the mutations are performed in each segment of chromosome, as indicated in Figure 6.

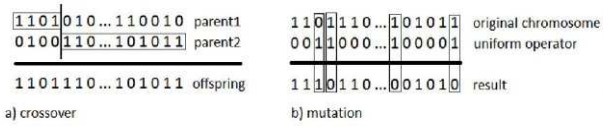


Fig. 6. Performing crossover and mutation.

In this work, it was used a modification of the expression given by [6] to calculate the circuit evaluation, so that the maximum score is 100:

$$E = \frac{100}{1 + \sum_{i=1}^{5} \omega_i \left| S(f_i) - O(f_i) \right|} \quad (1)$$

where E is the circuit evaluation score (from 0 to 100), ω_i is the gain weight at frequency f_i, i is the frequency order (multiple of cutoff frequency - f_c) used in the calculations, such as $f_1=f_c/100$; $f_2=f_c/10$; $f_3=f_c$; $f_4=10f_c$; $f_5=100f_c$. S(fi) is the ideal gain at f_i and O(f_i) is the actual gain at f_i. The original expression [6] takes only the weighted sum of deviations at several frequencies and, thus, the greater the calculation, the worse will be the circuit. In our modified expression (1), the result is directly proportional to the fitness of the circuit.

The nodal analysis, calculated for the cutoff frequency (fc) is just one step of the circuit evaluation procedure. For the full evaluation, the algorithm obtains the gain of the circuit, by nodal analysis, at five frequencies, multiples of the cutoff frequency: 0.01fc, 0.1fc, fc, 10fc and 100fc. Those frequencies were chosen in order to evaluate the transfer function, two decades bellow and two decades above the cutoff frequency.

Those few sampling points are spread enough to evaluate properly the filter response and they do not jeopardize the performance of algorithm, since a larger set of sampling points would require a larger processing time. The weights were chosen so that they are higher in the vicinity of the cutoff frequency, since that region defines the behavior of the filter. The weights $\omega_1 = \omega_5 = 1$ and $\omega_2 = \omega_3 = \omega_4 = 10$ proved to be suitable for the evaluation function.

III. Results

Table III presents the results obtained after few runs of the algorithm, considering the four types of filters, for a 1kHz cutoff frequency. The evaluation score was obtained from (1). The corresponding frequency responses are presented in Figure 7. They were obtained by using 5Spice Analysis 2.20.0 [7].

It can be observed from the band-pass filter, shown in Fig 7, that for input signal smaller than 0.2kHz and large than 5kHz, the gain in less than -60dB. It means that less than 0.1% of applied signal reaches the output. In other words, the signal was totally filtered at those frequencies. At the vicinity of 1kHz the gain is approximately 0db, which means there is no attenuation at that cutoff frequency. The narrow band-pass range of the circuit gives it its high score.

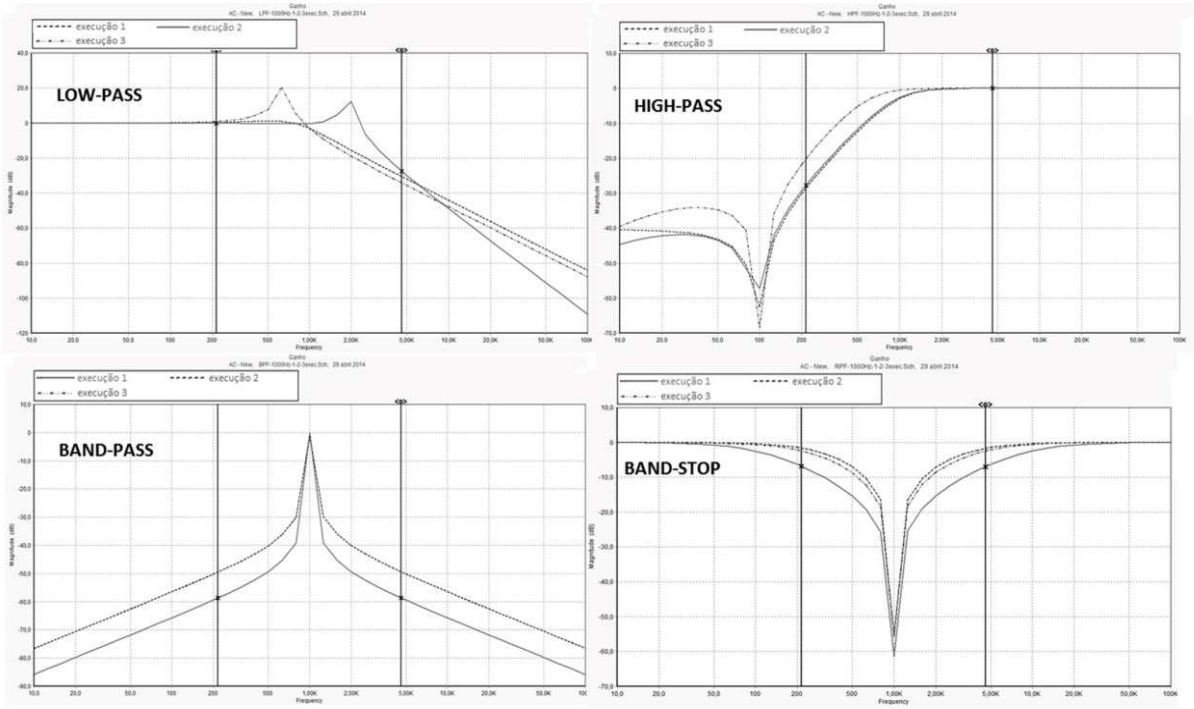


Fig. 7. Frequency response of the circuits given by Table III.

TABLE III. GENERATED CIRCUITS.

Function	*Generated Filters by GA*			
	Execution	***Topology***	***Components***	***Score***
Lowpass	1	100000	L=1.27H; R=6.3K; C=32nF	94.34
Lowpass	2	001000	C=3.3uF; L=2.15H; C=3.6nF	95.67
Lowpass	3	101000	L=148mH; C=17uF; C=414nF	94.81
Highpass	1	100011	C=4.6uF; L=5.8mH; C=474uF	99.13
Highpass	2	100100	L=610mH;C=4.7uF; L=5.9mH	99.12
Highpass	3	100100	C=7.3uF; L=350mH; L=9.3mH	99.01
Bandpass	1	100000	R=37.2K; L=30.4mH; C=834nF	99.05
Bandpass	2	000010	C=209uF; L=120uH; R=2.15M	99.01
Bandpass	3	000010	L=118uH; C=213uF; C=59.2uF	99.00
Notch	1	001110	C=27.8uF; L=918uH; L=1.7H	99.25
Notch	2	001110	L=2.6mH; C=9.6uF; L=1.03mH	99.02
Notch	3	100011	C=2mF; C=12.2uF; L=2.1mH	99.15

A similar analysis can be performed for the other filters. The band-stop filter, or notch, is complementary to the band-pass filter, rejecting the input signal around 1kHz. The high-pass filter (HPF) rejects the input signal less than 1kHz, mainly less than 0.2kHz. Note that at 100Hz (0.1 f_c).the gain of the HPF is less than -60db. The low-pass filter (LPF) rejects input signal frequencies greater than 1kHz. It can be verified that the negative slope of the curve for frequencies greater than 1kHz would be much more intense, and the LPF score would be higher, if the input resistance r_i of the circuit was less than 50 ohms.

IV. CONCLUSIONS

The algorithm properly generated the desired circuits, and thus proved to be a convenient tool for the development of analog circuits.

The use of object-oriented design to implement a GA, developed in this work, offers a wide range of advantages. If can be easily changed or upgraded. It also can be migrate to other circuit topologies, with more components, and even with other types of components, such as diodes and transistors, so that more complex circuits can be automatically designed. The classes can be adapted to include new attributes and to perform new methods.

ACKNOWLEDGMENT

The authors acknowledge CAPES, CNPq and FAPEMIG for their financial support.

REFERENCES

[1] I. Rechenberg, "Cybernetic solution path of an experimental problem". Royal Aircraft Establishment, Farnborough p. Library Translation 1122, 1965

[2] A. Das, R. Vemuri,"An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms", IEEE Computer Society Annual Symposium on VLSI, 2007.

[3] R. S. Zebulum et al., "Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms", CRC Press, Boca Raton, Florida, 2002

[4] LIU, Mingguo, HE, Jingsong Automated Analog Circuit Design Synthesis Using A Hybrid Genetic Algorithm with Hyper-Mutation and Elitist Strategies. Information Technology and Computer Science, 2009. 1, 23-32 Published Online October 2009 in MECS (http://www.mecs-press.org/

[5] E.T.Cuantle, et al."Automatic Synthesis of Electronic Circuits using Genetic Algorithms", Computación y Sistemas Vol. 10 No. 3, pp 217-229, 2007.

[6] M. Murakawa, et al. "Analogue EHW Chip for Intermediate Frequency Filters, Evolvable Systems: From Biology to Hardware", Procedings of the Second International Conference 1ed. Ed. M. Sipper, D. Mange, and A. Péres-Uribe, Lusanne. Switzerland: Springer, pp.134-143, 1998.

[7] R. P. Andresen, "5Spice Analysis" www.5spice.com, 2013

A.2. Publicação na Revista SODEBRAS n.110 – fev/2015

Revista SODEBRAS – Volume 10
N° 110- FEVEREIRO/2015

ALGORITMO GENÉTICO ORIENTADO A OBJETOS APLICADO EM HARDWARE EVOLUTIVO

ORLANDO VERDUCCI JR[1]; PAULO CÉSAR CREPALDI[1]; LEONARDO BRESEGHELLO ZOCCAL[1]
1 – UNIVERSIDADE FEDERAL DE ITAJUBÁ-MG
overducci@gmail.com crepaldi@unifei.edu.br lbzoccal@unifei.edu.br

***Resumo** - Este trabalho descreve o desenvolvimento de um algoritmo genético para sintetizar automaticamente circuitos elétricos analógicos. O contexto do trabalho é o desenvolvimento de filtros passivos RLC de até três componentes, com escolhas da frequência de corte e do tipo de filtro: passa-baixa, passa-alta, passa-faixa ou rejeita-faixa. A avaliação de cada solução gerada é realizada calculando-se as tensões no circuito por Análise das Tensões Nodais para as diversas topologias possíveis, sem utilizar simuladores ou hardware programável. O algoritmo genético foi totalmente desenvolvido com linguagem orientada a objetos, Java, a partir de um diagrama de classes que indica as relações entre população, indivíduo (circuito candidato), cromossomo (representação genética do circuito), método de seleção, cruzamento, mutação, avaliação do indivíduo (qualidade do circuito), entre outras classes pertinentes ao tratamento do problema.*

***Palavras-chave:** Algoritmo Evolucionário, Algoritmo Genético, Hardware Evolutivo, Filtros Passivos, Programação Orientada a Objetos.*

I. INTRODUÇÃO

O algoritmo genético (AG) pertence a uma categoria de algoritmos evolucionários que tem por princípio a inspiração na natureza para encontrar soluções para um determinado problema. Mais especificamente, os AGs modelam, de maneira computacional, os fenômenos envolvidos na reprodução dos seres vivos (cruzamento e mutação) e na seleção natural dos indivíduos mais adaptados ao meio, ao longo de várias gerações. Cada geração de indivíduos é obtida após um ciclo do AG, conforme mostra a Figura 1.

Figura 1 Ciclo de execução do Algoritmo Genético

Os algoritmos genéticos são aplicados em muitos problemas de otimização e em diversas áreas do conhecimento, como se vê, por exemplo, em (Pina et. al, 2011) e (Camargo et al, 2013), em que a solução analítica é de difícil implementação.

Projetar um circuito elétrico é um problema típico que pode ser automatizado com o uso de AG, considerando que a sua execução consiste em encontrar um circuito que atenda os requisitos de projeto dentre um grande número de combinação de topologias, componentes e seus valores possíveis.

Para um AG, cada indivíduo é uma solução candidata ao problema proposto. O conjunto dessas soluções candidatas representa uma população. O meio em que os indivíduos devem se adaptar, ou sobreviver, é representado pelo critério de avaliação dos indivíduos como solução do problema.

II. OBJETIVO

O objetivo deste trabalho é determinar de forma automática, com o uso de um AG proposto com abordagem orientada a objetos, a topologia (grafo de interconexão entre os componentes), o tipo de cada componente (resistor, capacitor ou indutor) e seus valores (em ohm, farad ou henry) de forma que o circuito se constitua em um filtro de sinais elétricos com até três componentes que atenda os seguintes requisitos: a frequência de corte desejada e o tipo de filtro, com relação à sua faixa de frequências passante.

A Figura 2 apresenta o circuito evolutivo inserido no circuito completo, que inclui o sinal de entrada v_e = 1 cos(ωt+0°) (V), a resistência de entrada r_e=50 Ω, o circuito RLC a ser sintetizado e a resistência de carga R_L=100 MΩ. Esses componentes e valores foram escolhidos deliberadamente para inserir um sinal de entrada AC e uma carga de alta impedância ao circuito sintetizado.

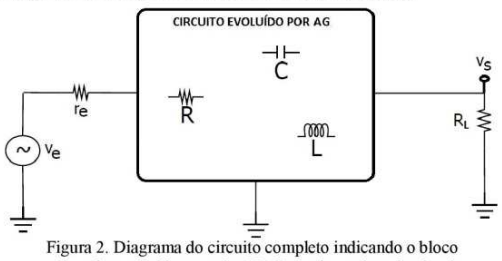


Figura 2. Diagrama do circuito completo indicando o bloco correspondente ao filtro passivo RLC que é gerado pelo algoritmo

III. CONCEITOS SOBRE FILTROS PASSIVOS

Um filtro é um circuito que permite a passagem de sinais que estejam inseridos dentro de uma certa faixa de

ISSN 1809-3957

frequências, enquanto rejeita os sinais fora desta faixa. As frequências limites destas faixas são denominadas de frequências de corte (f_c).

Denominam-se filtros passivos aqueles formados apenas por componentes passivos, ou seja, que não requerem alimentação elétrica para seu funcionamento. Os filtros passivos são formados por apenas três tipos de componentes: elétricos: resistores, capacitores e indutores; e, no que diz respeito à seletividade em frequência, podem ser divididos em: Passa-Baixa, Passa-Alta, Passa-Faixa e Rejeita-Faixa.

Por ser um circuito seletivo de frequências, os filtros devem ser avaliados no domínio da frequência. A função de transferência que será utilizada neste trabalho é a relação de tensão entre a saída e a entrada, conforme visto na Figura 3 (Edminister, 1985).

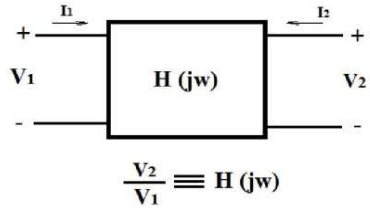


Figura 3 Relação de tensão entre saída e entrada do circuito, em função da frequência ω, usada na análise do filtro passivo.

A frequência de corte de um filtro está associada a presença do que se denomina de polos da sua função de transferência. O número de polos, por sua vez, depende da quantidade de elementos reativos (capacitores e indutores). Cada elemento reativo presente no filtro correspondendo a um polo de sua função de transferência provoca uma atenuação de 20dB/década na saída. Para um filtro de até três componentes, como o que será sintetizado pelo AG neste trabalho, a atenuação máxima que se pode obter é de 60 dB/década.

IV. METODOLOGIA

A Figura 4 apresenta o diagrama de classes em que foi baseado o modelo para o desenvolvimento do algoritmo genético que sintetiza o circuito de filtro passivo. O projeto do AG foi desenvolvido com orientação a objetos em linguagem Java para permitir flexibilidade ao desenvolvedor que deseje modificar as técnicas empregadas como, seleção de pais, avaliação, representação cromossômica do circuito, etc, e assim comparar os resultados. Cada uma das técnicas do AG foi construída em classes separadas como podem ser vistas no diagrama de classes. A classe Avaliação contém o método que calcula a relação $V_{saída}/V_{entrada}$ para múltiplos e submúltiplos de f_c, usando análise nodal, de modo a obter a resposta em frequência do circuito.

O circuito a ser projetado é representado por um cromossomo que contém os segmentos de topologia (6 bits) , tipos dos componentes (6 bits) e campos de valor (31 bits cada componente), conforme se vê na Figura 5.

A codificação da topologia reserva dois bits para cada componente que compõe o circuito, conforme apresentado na Tabela I.

Por exemplo, a codificação de topologia 100100 corresponde à sequência de inserções 10 (em série com a saída), 01(em paralelo com anterior), e 00(em paralelo com a saída), conforme apresentado na Figura 6.

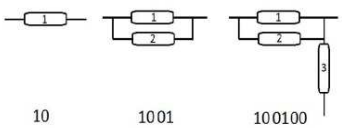


Figura 6. Sequência de inserções para a topologia codificada por 100100

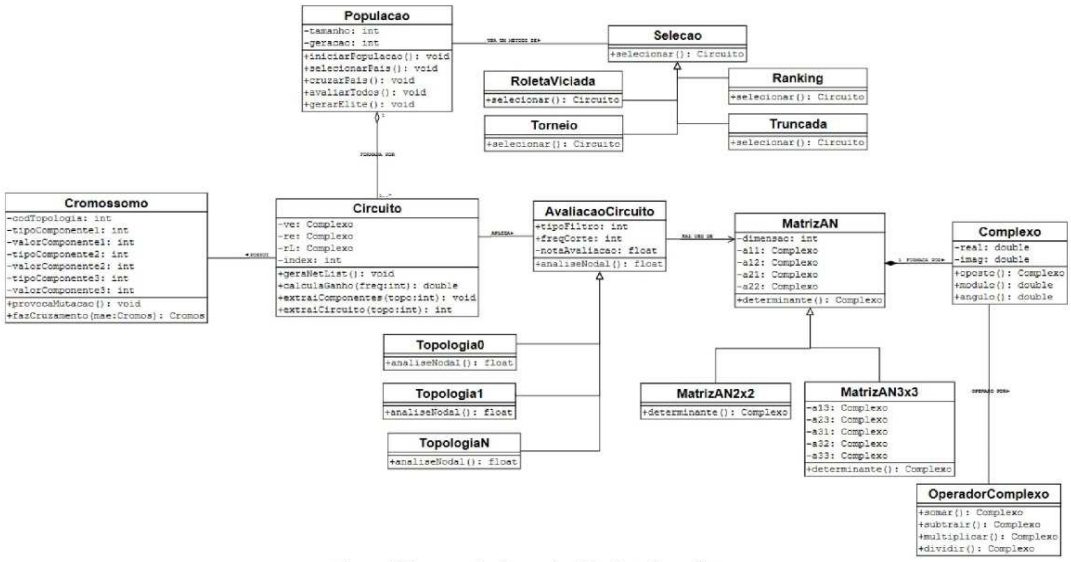


Figura 4.Diagrama de classes do AG orientado a objetos.

ISSN 1809-3957

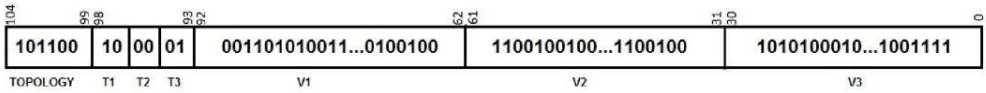


Figura 5. Segmentos do cromossomo do circuito RLC de até 3 componentes

TABELA I. CÓDIGOS DE TOPOLOGIA

Código	Significado	
	Modo de Inserção	Referente a
00	em paralelo	saída
01	em paralelo	componente anterior
10	em série	saída
11	em série	componente anterior

A codificação de componentes (2 bits cada um) é dada pela Tabela II.

Assim, por exemplo, a codificação de componentes 011000 corresponde às inserções de um capacitor (01), seguido de um indutor (10) e, por último, um resistor (00) na sequência de inserções definida pelo código da topologia. O componente nulo (11) é considerado um ramo em curto-circuito quando inserido em série e um ramo aberto quando inserido em paralelo.

TABELA II. CÓDIGOS DE COMPONENTE

Código	Significado	
	Compon.	unidade
00	resistor	<campo valor> ($x10^{-3}$) ohm
01	capacitor	<campo valor> ($x10^{-12}$) farad
10	indutor	<campo valor> ($x10^{-9}$) henry
11	nulo	-o-

A partir de uma população inicial de 100 indivíduos (circuitos) gerados aleatoriamente, se inicia o ciclo de avaliações, cruzamentos, mutações e preservação do mais apto (elitismo), até que se encontre um circuito com avaliação suficientemente alta. A avaliação é realizada pelo método NodesAnalysis( ) da classe Fitness. Os cruzamentos e as mutações são realizados em cada segmento do cromossomo de modo semelhante ao ilustrado na Figura 7. O ponto de corte para o cruzamento e o operador uniforme para a mutação são gerados aleatoriamente em cada operação genética realizada.

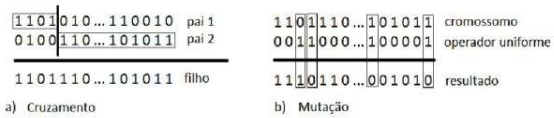


Figura 7. Operações de cruzamento e mutação utilizados no projeto

Para este trabalho foi usada uma adaptação da expressão apresentada em (Murakawa et al, 1998) para se calcular a avaliação do circuito, de forma que a nota máxima seja 100:

$$A = \frac{100}{1 + \sum_{i=1}^{5} \omega_i \left| S(f_i) - O(f_i) \right|} \qquad (1)$$

Onde:
A= nota de avaliação do circuito: 0 a 100;
ω_i = peso do ganho na frequência f_i;
i = ordem da frequência usada no cálculo, sendo:
f_1= $f_c/100$; $f_2=f_c/10$; $f_3=f_c$; $f_4=10f_c$; $f_5=100f_c$
$S(f_i)$= ganho calculado em f_i;
$O(f_i)$= ganho ideal em f_i.

Desta forma, nota-se que a análise nodal calculada para a frequência de corte (f_c) é apenas uma etapa do procedimento de avaliação do circuito. Para a avaliação completa, o algoritmo calcula o ganho do circuito, por Análise Nodal, para 5 frequências múltiplas da frequência de corte: $0{,}01f_c$, $0{,}1f_c$, f_c, $10f_c$ e $100f_c$. Essas frequências foram escolhidas de maneira a avaliar a função de transferência duas décadas antes e duas depois da frequência de corte, o que é razoável para o objetivo do trabalho, pois são poucos pontos de amostragem, e suficientemente abrangentes, que não comprometem o desempenho do algoritmo, pois quanto mais pontos de amostragem, maior o tempo de execução. Os pesos foram tomados de maneira que na vizinhança da frequência de corte os pesos sejam bem maiores, pois esta região é a que define o comportamento do filtro. Os valores de peso a seguir se mostraram adequados para a função de avaliação: $\omega_1=\omega_5=1$ e $\omega_2=\omega_3=\omega_4=10$.

A proposta de implementação para o elitismo consiste em manter na geração seguinte o circuito com melhor avaliação da geração corrente, descartando o pior filho gerado.

V. RESULTADOS

São apresentados na Tabela III os resultados obtidos com algumas execuções do algoritmo considerando os quatro tipos de filtros e para uma frequência de corte de 1kHz. A nota de avaliação foi calculada usando (1). As respectivas respostas em frequência são apresentadas na Figura 8, e foram geradas pelo aplicativo 5Spice Analysis versão 2.20.0.

VI. CONCLUSÃO

A execução do algoritmo gerou corretamente os circuitos solicitados, mostrando-se uma excelente ferramenta para desenvolvimento de circuitos analógicos que pode ser estendida para outras aplicações, com maior número de componentes, uma vez que a abordagem com

ISSN 1809-3957

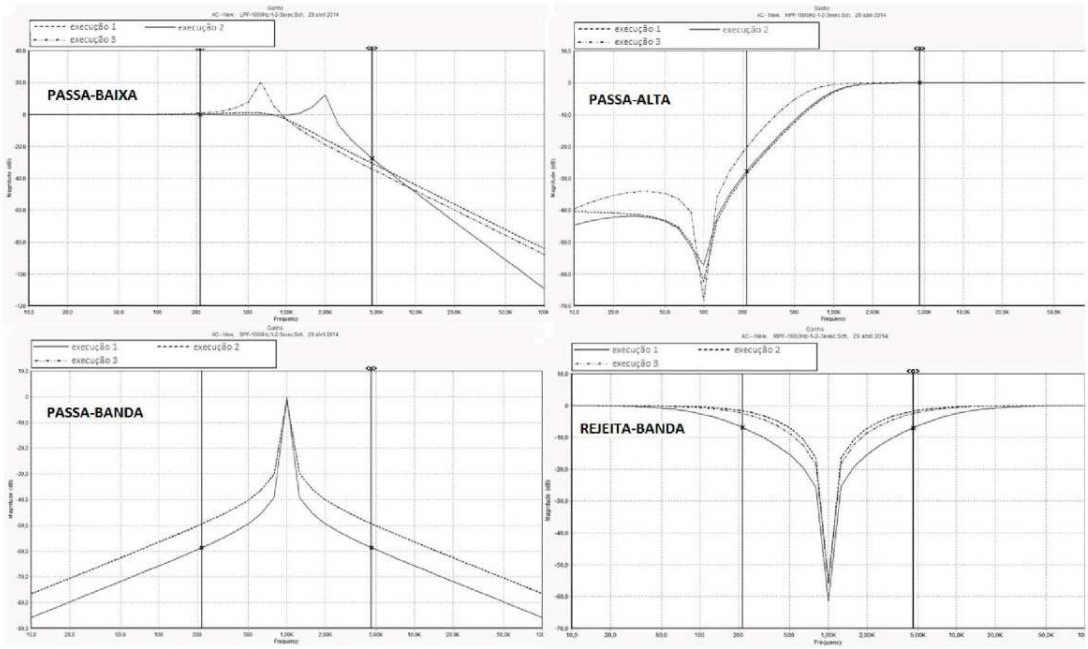


Figura 8. Resposta em frequência dos circuitos listados na Tabela III

orientação a objetos se caracteriza pela modularidade do código, facilitando a manutenção e a evolução do projeto. O uso de simuladores, em conjunto com a computação paralela, para obtenção das avaliações aumentaria muito o potencial do GA, pois seriam reduzidos o código, uma vez que seriam excluídos os métodos para cálculos da análise nodal, e o tempo de execução, visto que as avaliações seriam obtidas simultaneamente para um conjunto de indivíduos da população.

TABELA III. CIRCUITOS GERADOS

Tipo	Filtros Gerados pelo AG			
	Exec	Topol	Componentes	Nota
Passa-baixa	1	100000	L=1.27H; R=6.3K; C=32nF	94.34
Passa-baixa	2	001000	C=3.3uF; L=2.15H; C=3.6nF	95.67
Passa-baixa	3	101000	L=148mH; C=17uF; C=414nF	94.81
Passa-alta	1	100011	C=4.6uF; L=5.8mH; C=474uF	99.13
Passa-alta	2	100100	L=610mH;C=4.7uF; L=5.9mH	99.12
Passa-Alta	3	100100	C=7.3uF; L=350mH; L=9.3mH	99.01
Passa-banda	1	100000	R=37.2K; L=30.4mH; C=834nF	99.05
Passa-banda	2	000010	C=209uF; L=120uH; R=2.15M	99.01
Passa-banda	3	000010	L=118uH; C=213uF; C=59.2uF	99.00
Rejeita-banda	1	001110	C=27.8uF; L=918uH; L=1.7H	99.25
Rejeita-banda	2	001110	L=2.6mH; C=9.6uF; L=1.03mH	99.02
Rejeita-banda	3	100011	C=2mF; C=12.2uF; L=2.1mH	99.15

VII. REFRÊNCIAS BIBLIOGRÁFICAS

HOLLAND, John .H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, Mich., 1975.

LINDEN, Ricardo, "Algoritmos genéticos: uma importante ferramenta da inteligência computacional" Rio de Janeiro, Brasport, 2006.

CAMARGO, L. A. S, GUARNIER, E., RAMOS, D. S., "Análise da atratividade econômica de associações hidro eólicas como suporte a decisões estratégicas de comercialização de energia e novos investimentos", Revista Sodebras, vol.8, n.95, novembro 2013.

PINA, M. C., ABUD M. M., "Automação de horários escolar: um estudo utilizando uma abordagem sobre algoritmo genético", Revista Sodebras, vol.6, n.65, maio 2011.

ALMEIDA, C. P., GONÇALVES, R. A., "Hardware evolutivo – uma introdução", www.unicentro.br/pesquisa/anais/seminario/.../pdf/artigo_727.doc

EDMINISTER, J. A. "Circuitos elétricos", 2a. ed, McGraw-Hill do Brasil, Cap. 11, 1985.

VEMURI, Ranga, Das, Angan, "An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms", IEEE Computer Society Annual Symposium on VLSI, 2007.

ISSN 1809-3957

ZEBULUM, Ricardo Salem; PACHECO, Marco Aurélio; VELLASCO, Marley Maria; "Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms", CRC Press, Boca Raton, Florida, 2002

LIU, Mingguo, HE, Jingsong Automated Analog Circuit Design Synthesis Using A Hybrid Genetic Algorithm with Hyper-Mutation and Elitist Strategies. Information Technology and Computer Science, 2009. 1, 23-32 Published Online October 2009 in MECS (http://www.mecs-press.org/)

MURAKAWA, M, et al. "Analogue EHW Chip for Intermediate Frequency Filters, Evolvable Systems: From Biology to Hardware", Procedings of the Second International Conference 1ed. Ed. M. Sipper, D. Mange, and A. Péres-Uribe, Lusanne. Switzerland: Springer, pp.134-143, 1998.

ANDRESEN, R. P., "5Spice Analysis" www.5spice.com, 2013

VIII. COPYRIGHT

ISSN 1809-3957

REFERÊNCIAS

ANDRESEN, R. P., *5Spice Analysis*, www.5spice.com, 2013.

AVERY, O.T., et al., *Studies on the chemical nature of the substance inducing transformation of pneumococcal types*. Journal of Experimental Medicine 79, 137–157, 1944.

BÄCK, T., *Evolutionary algorithms and their standard instances-introduction*. Handbook of Evolutionary Computation, IOP Publishing Ltd and Oxford University Press, 1997.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.), *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing, 2000.

BAKER, J. E., *Adaptive selection methods for genetic algorithms*. In J.J. Grefenstettes, ed, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Erlbaum, 1985.

BARRICELLI, N.A., *Numerical testing of evolution theories: Part I Theoretical introduction and basic tests*. Acta Biotheoretica 16 (1-2): 69–98, 1962.

BLICKLE, T., THIELE, L., *A comparison of selection schemes used in evolutionary algorithms*, ETH Zurich, 1997.

BOOCH, G., JACOBSON, I., RUMBAUGH, J., *The Unified Modeling Language Reference Manual*, Addison Wesley Longman, Inc., 1999.

BOX, G.E.P., *Evolutionary Operation: a method for increasing industrial productivity*, Appl. Stat., 1957.

BREMERMANN, H.J., *Optimization through evolution and recombination. Self-organizing systems*, Yovits, M. C. et al, Eds, Washington, DC, 1962.

CUANTLE, E.T, et al. *Automatic Synthesis of Electronic Circuits using Genetic Algorithms*, Computación y Sistemas, Vol. 10 No. 3, pp 217-229, 2007.

DARWIN, C., *The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*; London: John Murray, Albemarle Street, 1859.

DE JONG, K., FOGEL, D.B. SCHWEFEL, H.-P., *A history of evolutionary computation, Handbook of evolutionary computation*, IOP Publishing Ltd and Oxford University Press, New York, 1997.

EDMINISTER, J. A., *Circuitos elétricos*, 2a. ed, McGraw-Hill do Brasil, Cap. 11, 1985.

FOGEL, L.J., *Autonomous Automata*, Industrial Res, 1962.

FRASER, A.S., *Monte Carlo analyses of genetic models*. Nature 181 (4603): 208–9, 1958.

FRASER, A.S., *Simulation of genetic systems by automatic digital computers*, Aust. J. Biol. Sci., 1957.

FRIEDBERG, R.M., *A learning machine: part I*, IBMJ, 1958.

HALDANE, J.B.S., *The causes of evolution*, Originally published in London; New York; Longmans, Green, 1932.

HOLLAND, J.H., *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

JACOB, J. M., *Advanced AC circuits and electronics: principles & applications*, pages 150-152, Thomson, Delmar Learning, 2004.

KOZA, J. R., *Genetic programming: on the programming of computers by means of natural selection*, A Bradford Book, The MIT Press, Massashusetts Institute of Technology, 1992.

LEVENE, P., *The structure of yeast nucleic acid*, J Biol Chem 40 (2): 415–24., 1919.

LINDEN, R., *Algoritmos Genéticos*, 3ª ed., Editora Ciência Moderna Ltda., Rio de Janeiro, Brasil, 2012.

LIU, M., HE, J., Automated Analog Circuit Design Synthesis Using A Hybrid Genetic Algorithm with Hyper-Mutation and Elitist Strategies. Information Technology and Computer Science, 1, 23-32 Published Online October 2009 in MECS (http://www.mecs-press.org/, 2009.

MENDEL, G., *Versuche über Pflanzen-Hybriden. Verh. Naturforsch*. Ver. Brünn 4: 3–47 (in English in 1901, J. R. Hortic. Soc. 26: 1–32), 1866.

MICHALEWICZ, Z., FOGEL, D.B., *How to Solve it: Modern Heuristics*, 2nd ed., Springer-Verlag, Berlin, 2004.

MIESCHER, F., *Ueber die chemische Zusammensetzung der Eiterzellen (on the chemical composition of pus cell)*, Medicinisch-chemische Untersuchungen, 4: 441–460, 1871.

MITCHELL, M., *An Introduction to Genetic Algorithms*, 1st ed, MIT Press, Cambridge, EUA, 1996.

MURAKAWA, M., et al., *Analogue EHW Chip for Intermediate Frequency Filters, Evolvable Systems: From Biology to Hardware*, Procedings of the Second International Conference, 1st ed. Ed. M. Sipper, D. Mange, and A. Péres-Uribe, Lusanne. Switzerland: Springer, 1998, pp.134-143.

ORSINI, L. Q., *Curso de Circuitos Elétricos*, Vol.2, EDUSP, São Paulo, Brasil, 1991.

RECHENBERG, I., *Cybernetic Solution Path of an Experimental Problem*, Roy. Aircr. Establ. Libr. Transl.,1122, Farnborough, Hants, UK, 1965.

RECHENBERG, I., *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog Verlag, 1973.

RIDLEY, M., *Evolution*, 2nd ed., Blackwell Science, Cambridge, 1996.

SCHWEFEL, H.-P., *Collective Phenomena in Evolutionary Systems*, In Preprints of the 31st Annual Meeting for General Systems Research, Budapeste, 2, pp. 1025-1033, 1987.

SCHWEFEL, H.-P., *Evolution and Optimum Seeking*, Wiley, New York, 1995.

SCHWEFEL, H.-P., *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik*, Diploma thesis, Technical University of Berlin, 1965.

SMITH, J.M., *The evolution of sex*, Cambridge University Press, 1978.

ZEBULUM, R.S., PACHECO, M.A C., VELLASCO, M.M.B.R., *Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms*, The CRC Press International Series on Computational Intelligence, 2001.