Fabricio Kassardjian

Algoritmos Genéticos
Um algoritmo evolucionário para otimização

Brasil

14 de novembro de 2019

Fabricio Kassardjian

Algoritmos Genéticos
Um algoritmo evolucionário para otimização

Monografia de final de curso, Instuto de Matemática e Estatística - USP, Matemática Aplicada

Universidade de São Paulo – USP

Insituto de Matemática e Estatística

Bacharelado em Matemática Aplicada e Computacional

Orientador: Anatoli Iambartsev

Brasil

14 de novembro de 2019

Fabricio Kassardjian
Algoritmos Genéticos
Um algoritmo evolucionário para otimização/ Fabricio Kassardjian. – Brasil, 14 de novembro de 2019-
54p. : il. (algumas color.) ; 30 cm.

Orientador: Anatoli Iambartsev

Monografia – Universidade de São Paulo – USP
Insituto de Matemática e Estatística
Bacharelado em Matemática Aplicada e Computacional, 14 de novembro de 2019.

1. Algoritmos Evolutivos. 2. Algoritmos Genéticos. 3. Otimização. I. Orientador. II. Universidade de São Paulo - USP. III. Instituto de Matemática e Estatística - IME. IV. Titulo

Fabricio Kassardjian

Algoritmos Genéticos
Um algoritmo evolucionário para otimização

Monografia de final de curso, Instuto de Matemática e Estatística - USP, Matemática Aplicada

Trabalho aprovado. Brasil, 24 de novembro de 2012:

Anatoli Iambartsev
Orientador

Professor
Convidado 1

Professor
Convidado 2

Brasil
14 de novembro de 2019

Resumo

Um dos problemas mais comuns em matemática é a busca de soluções que maximizem ou minimizem certa função e temos a disposição diversas formas de resolver que podem ser classificadas em técnicas baseadas em cálculo, aleatórias ou métodos que verificam todo o espaço solução pela melhor delas. Dentre as técnicas aleatórias temos aquelas puramente aleatória como o *random walk* e aquelas consideradas aleatórias guiadas como por exemplo o *simulated annealing* ou resfriamento simulado. No âmbito de sistemas evolutivos, os algoritmos genéticos podem ser classificado também como um método aleatório guiado, onde o estado corrente influencia a próxima escolha para a solução.

O objetivo desse trabalho é verificar e avaliar os algoritmos genéticos, explorando as técnicas e os mecanismos usados nesse método. Para tal será desenvolvido o algoritmo em uma linguagem de programação usando o paradigma de orientação a objetos e aplicá-lo em um problema de busca pela solução ótima, sendo que buscaremos a melhor solução para um modelo de Ising. Com isso poderemos avaliar o funcionamento e perfomance do método e como os parâmetros utilizados podem influenciar o resultado

Palavras-chave: Algoritmos Evolutivos. Algoritmos Genéticos. Otimização. Programação Orientada a Objetos.

Abstract

This is the english abstract.

Keywords: Evolutionary Algorithms. Genetic Algorithms. Optimization. Object Oriented Programming.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

SGA	Simple Genetic Algorithm
GA	Genetic Algorithm
NFL	No-Free-Lunch Theorem
EA	Evolutionary Algorithm
OO	Orientação a objetos

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

1.1 Algoritmos evolucionários

Os algoritmos evolucionários são um subconjunto da computação evolutiva, sendo essa um conjunto de algoritmos para busca de soluções ótimas globais. A computação evolutiva é baseada na evolução das espécies definida na biologia, e assim os algoritmos evolucionários se baseiam em processos encontrados na natureza de seleção, reprodução e mutação das espécies, com o propósito de otimizar a solução para um problema definido.

Um problema de otimização é definido como maximizar, ou minimizar, $f(x)$ sujeito a $g_i(x) \leq 0\, i = \{1, \ldots, m\}$, e $h_j(x) = 0, j = \{1, \ldots, n\}$ com $x \in \Omega$. A solução maximiza, ou minimiza, o escalar $f(x)$ onde x é um vetor com dimensão n, $x = \{x_1, x_2, \ldots, x_n\}$ do espaço de soluções Ω. (COELLO; LAMONT; VELDHUIZEN, 2007)

Dessa forma o problema de otimização tem os seguintes componentes:

- Função objetivo $f(x)$: função de avaliação que deve se minimizada ou maximizada;
- As restrições $g_i(x)$ e $h_j(x)$: definem limites para as soluções que são permitidas;
- Espaço de soluções $\Omega\ = \Omega(g_i, h_j)$: conjunto com todas as possíveis soluções para o problema;

De forma sintética pode ser escrito como, sendo a função

$$f : \Omega \to \mathbb{R}$$

e a solução o vetor $x \in \Omega$ tal que

$$\arg\min_{x \in \Omega} f(x)$$

que pode facilmente ser convertido para um problema de maximização usando $-g(x)$

Para atingir esse objetivo, os algoritmos evolutivos trabalham com uma população de indivíduos que indicaremos como soluções candidatas para o problema. Baseado na avaliação de cada individuo com relação ao seu ambiente, em nosso contexto a função de avaliação, e usando operadores definidos para seleção e evolução, a população vai sendo modificada até que se atinja um resultado satisfatório para o problema. Assim como no processo de evolução das espécies definidos por Darwin, que introduziu o conceito de seleção natural através da sobrevivência do mais apto, os indivíduos da população que tem melhores avaliações, ou seja, que melhor se adaptam ao ambiente, possuem mais probabilidade de sobrevivência e assim se reproduzirem.

Existem vários métodos para se otimizar um problema conforme pode ser visto na Figura 1. Existem os métodos enumerativos, que se tornam impraticáveis quando o Ω é muito amplo, pois o algoritmo deve testar todas as soluções possíveis. Os algoritmos determinísticos são os mais eficientes em determinadas condições, definidas pelo comportamento de $f(x)$. Se a função objetivo possui múltiplos máximos (ou mínimos) alguns algoritmos determinísticos, como o *hill-climbing* por exemplo, podem ficar 'presos' em soluções locais e não encontrar a solução global. A última categoria, onde se encontram também os algoritmos evolucionários, são os estocásticos (ou randômicos). Possuem a vantagem de não ficar presos em soluções locais, porém nem sempre obterão a melhor solução.

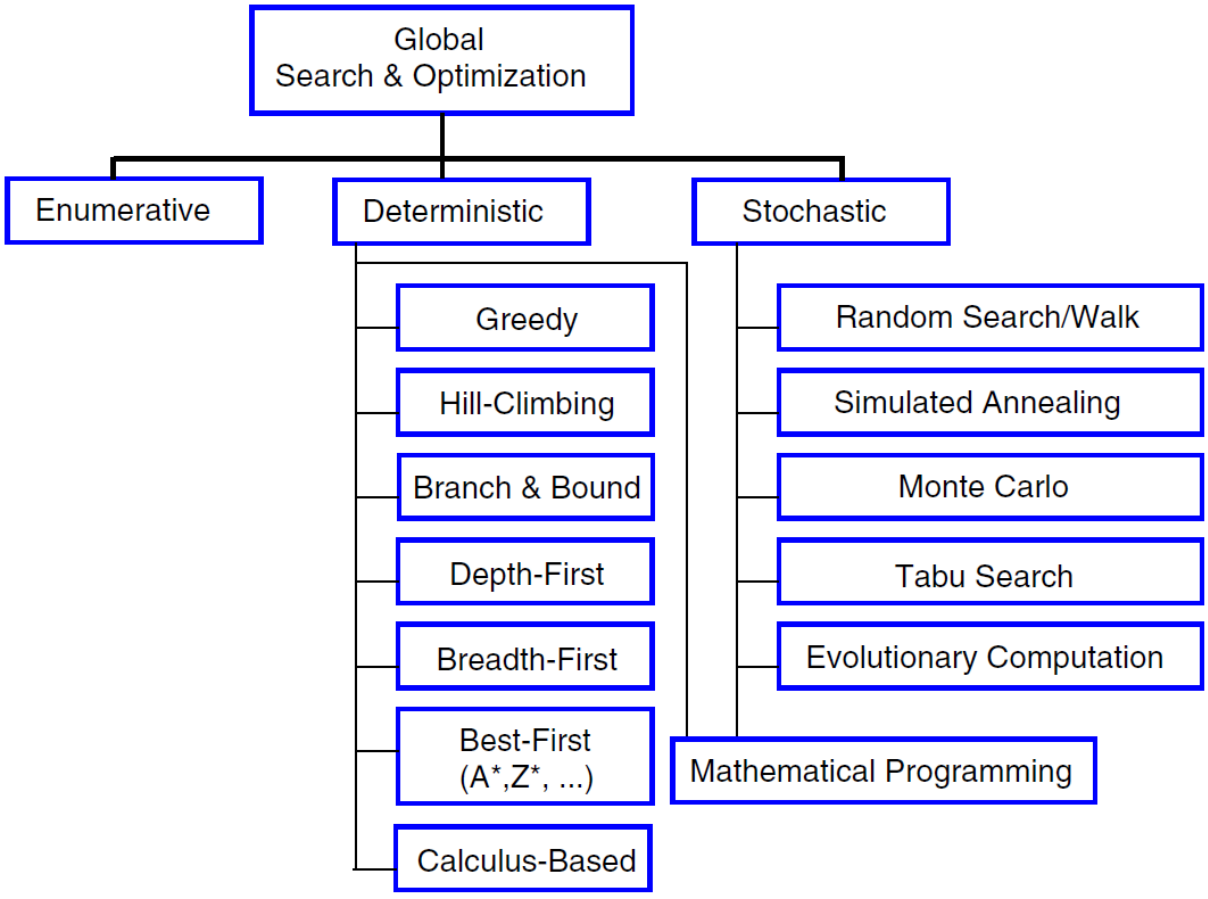

Figura 1 – Técnicas de otimização global - (COELLO; LAMONT; VELDHUIZEN, 2007)

Dentre a categoria dos métodos estocásticos existem os que são completamente aleatórios, como a busca randômica (ou passeio aleatório – *random walk*), e os que são de alguma forma guiado como o método de resfriamento simulado (*simulated annealling*) por exemplo. O algoritmo evolucionário se encaixa nessa ultima definição, onde existem componentes aleatórios atuando na seleção e reprodução dos indivíduos mas de forma guiada pelos resultados da função de avaliação.

De acordo com LINDEN, os AE são **heurísticas**[1] que não asseguram obter o melhor resultado possível, e além disso o resultado pode diferir entre as execuções do algoritmo.

Para SIVANANDAM; DEEPA, um algoritmo evolucionário são processos estocásticos e iterativos que operam em um conjunto de indivíduos (população). Cada individuo representa uma possível solução para o problema de otimização ou busca, sendo que os parâmetros estão de alguma forma codificados nesses indivíduos. A população inicial é gerada aleatoriamente e são avaliados usando alguma função, que determina o quão bem o indivíduo responde ao problema. Esse valor determina a direção de busca do algoritmo.

1.2 Biologia

A ideia por trás dos algoritmos evolucionários e por consequência do algoritmo genético, é a teoria de evolução das espécies na natureza de **Darwin**, onde a sobrevivência de cada indivíduo é determinada por como ele se adapta ao seu meio. Assim aqueles que conseguiam vantagens sobre os demais por ter uma maior sobrevivência se reproduziam mais, e assim, passavam para as próximas gerações essas características que os diferenciavam. Darwin chamou de seleção natural esse mecanismo da sobrevivência dos mais aptos.

Contudo Darwin não sabia explicar como essa informação era passada dos ancestrais para os descendentes, onde então entram as descobertas de **Mendel**, que através dos conceitos de genética determinava como características eram compartilhadas entre pais e filhos.

A unidade básica de informação é o gene, que é um bloco de sequências de DNA e o conjunto de genes formam o cromossomo. Cada gene tem um *locus*, que define a região dentro do cromossomo onde está localizado, e possui um conjunto de valores possíveis chamados de alelos. Os genes controlam as características do indivíduo, sendo que a expressão dessas no individuo é denominada de fenótipo. Assim um conjunto específicos de genes define o genótipo do indivíduo que está associado a um fenótipo, que apresenta as características codificadas no genótipo e que podem ser modificadas pelo ambiente.

Organismos com cromossomos combinados em pares são chamados de diplóides em contraste com os que não possuem pares, chamados de haplóides. Na natureza a maioria dos seres mais complexos que se reproduzem sexualmente são normalmente diplóides e possuem um ou mais pares de cromossomos sendo que sua quantidade e tamanho dependem de cada ser vivo.

Durante a reprodução, que pode ser de dois tipos, assexuada ou sexuada, ocorre a transmissão da informação. Na reprodução assexuada, o organismo replica a si mesmo,

[1] Heurísticas são algoritmos polinomiais que usualmente tendem a encontrar soluções ótimas ou próximas delas, mas sem garantias

e está mais presente em seres simples. Não apresenta tanta diversidade por não existir combinação de material genético entre dois seres, e fica sujeita apenas a alterações por mutação na cópia.

Na reprodução sexuada de seres diplóides, há presença de dois indivíduos que compartilham seu material genético para formar um novo organismo. No inicio da reprodução, existe a cópia do material genético e recombinação, também chamada de *crossover*(Figura 2). Esse processo é feito com os cromossomos se cruzando um sobre o outro, por isso o termo *crossover*, em um ou mais pontos havendo assim a troca nas sequências de genes. Após feita a recombinação, o material genético é divido em gametas que então são combinados com os gametas do outro pai para formar novamente um cromossomo diplódie completo, gerando assim os novos indivíduos. Para organismos haplóides é feita apenas a combinação das sequencias de cada pai.

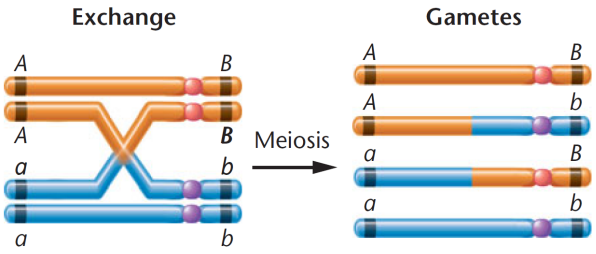

Figura 2 – Exemplo de reprodução com crossover - (KLUG et al., 2011)

Dentro da etapa de replicação do DNA podem ocorrer erros ou alterações influenciadas por fatores externos, gerando assim as mutações. Isso pode ser positivo, negativo ou não influenciar o resultado final, mas é um outro mecanismo que pode determinar a evolução das espécies.

Os genes definem as características dos indivíduos, mas também existe interação entre os genes, chamada de epistasia (*epistasis*), podendo um par de genes mascarar ou modificar a característica final de outro par de genes. (KLUG et al., 2011). Isso é importante do ponto de vista do algoritmo genético pois nem sempre a melhor avaliação estará associada a um único parâmetro, podendo estar associado a uma combinação de parâmetros.

Mendel ainda definiu o conceito de dominância-recessividade em organismos diplóides, assumindo que cada característica é controlada por pares de genes, sendo recebidos um de cada pai. Assim um dos alelos do gene é dominante sobre o outro apresentando a característica final no individuo, e outro alelo será considerado recessivo. Alguns algoritmos genéticos também implementam a lógica para genes dominantes-recessivos.

O processo de seleção acontece então combinando indivíduos que melhor se adaptaram as condições de sobrevivência. Esse seres selecionados devem gerar novos indivíduos

que tenha características ainda melhores. O outro caso pode acontecer também e o novo indivíduo ter piores condições de viver, e assim o processo de seleção natural levará esse espécime a extinção favorecendo outros que se sobressaíram na próxima geração.

A evolução então é um processo adaptativo onde através de mutações e recombinação entre os indivíduos, vão surgindo novas gerações que devem apresentar cada vez mais seres que se adaptam cada vez melhor ao ambiente.

1.3 Surgimento do algoritmo genético

Os algoritmos evolucionários vem sendo estudados a um longo tempo, desde da década de 40 que os cientistas se inspiram na natureza para criar os primeiros passos para a inteligência artificial, passando pela década de 50 onde começam os estudos sobre sistemas adaptativos para gerar soluções candidatas para problemas de difícil solução. Na década de 60 Rechenberg desenvolveu as estratégias evolucionárias (*evolutionary strategies*) usando cromossomos compostos de números reais para estudos com aerofólios. Também apareceu a programação evolucionária, usando estruturas de pequenas máquinas de estados para resolver determinadas tarefas, que evolui para programação genética onde pequenos programas passam a ser as soluções candidatas. Existiram outros trabalhos sobre algoritmos evolucionários, programação evolucionária e algoritmos genéticos nas áreas de otimização e aprendizado de máquina, porém foram os trabalhos de Holland nas décadas de 60 e 70 que consolidou os algoritmos genéticos. (MITCHELL, 1996; LINDEN, 2008)

Alguns autores como MITCHELL, JACOBSON, KWONG; MAN; TANG entre outros citam Holland como criador do algoritmo genético(*Genethic Algorithm* ou GA) em 1975 no livro *"Adaptation in Natural and Artificial Systems"*. Nele Holland formaliza uma estrutura para sistemas adaptativos, e enquadra o GA nessa estrutura como uma abstração para a evolução biológica.

O objetivo de Holland não era fazer algo especifico mas sim analisar os sistemas adaptativos e criar uma solução computacional que simulasse os processos encontrados nos sistemas de adaptação natural. Para isso cria uma abstração da evolução na biologia usando uma estrutural formal teórica que poderia atender diversos sistemas evolutivos e não somente o GA. Na formulação do GA por ele, os cromossomos usam representações binárias e os operadores utilizados são o de crossover, inversão e mutação inspirados na genética. O fato de ter usado cromossomos binários serviu de influência para vários outros estudos que se seguiram, mas será exposto no Capítulo 2 que existem outras formas de representar os cromossomos que dependem exclusivamente do problema a ser estudado.

O algoritmo genético geral proposto por Holland se baseia em uma população de cromossomos com alelos ‘0’ e ‘1’ e com os operadores de crossover, mutação e inversão. As evoluções seguiam os seguintes passos:

1. Seleção de um cromossomo na população de forma estocástica baseada nas avaliações de todos os cromossomos

2. Aplicações dos operadores genéticos sobre uma cópia do individuo selecionado em 1.

3. Seleção de outro cromossomo de forma aleatória com probabilidade igual para todos a ser substituído pelo novo cromossomo gerado em 2

4. Avaliar o novo cromossomo

5. Retonar ao 1

Essa é uma descrição bem resumida sobre o algoritmo e serve de base para as derivações. Por exemplo no caso do uso de um operador de crossover no Item 2, há a necessidade de seleção de outro cromossomo para formar um par e assim gerar uma nova estrutura.

Avaliando o algoritmo, temos alguns itens básicos que estarão presentes. A codificação do cromossomo para representar os parâmetros das soluções para os problemas. Uma população de cromossomos que serão avaliados simultaneamente durante as iterações do processo. A função de avaliação, que baseada nos parâmetros de cada estrutura irá fornecer uma forma de comparar os diversos elementos presentes na população. Os operadores genéticos que serão usados, sendo que podem ser combinados de diversas formas.

De acordo com MITCHELL existem três operadores para o algoritmo mais simples:

Seleção – seleciona cromossomos favorecendo os que tem melhor função de avaliação

Crossover – também chamado de recombinação, combina dois cromossomos na geração de um novo

Mutação – Altera de forma aleatória alguns alelos dos cromossomos, por exemplo em uma codificação binária seria alterar um dos bits do parâmetro.

No Capítulo 2 será explorado com mais detalhes cada um dos operadores. HOLLAND também descreve um operador de inversão, que apesar de garantir uma melhor diversidade genética impõe uma carga computacional grande para os ganhos efetivos, e nos trabalhos mais recentes se tornou um operador quase nunca usado.

Outro teoria que HOLLAND formula sobre os GAs é em relação ao processo de paralelismo intrínseco, sobre a forma do algoritmo trabalhar com esquemas (*schemata*) em vez de testar indivíduos específicos da população. GOLDBERG aborda esse tema com a hipótese dos blocos de construção (*Building block hypothesis*).

1.4 Teorema da inexistência do almoço grátis

Existe um compromisso entre a eficiência de um algoritmo e sua robustez, isto é, ele pode ser muito eficiente para determinados problemas mas serem ineficientes em outros. Isso implica que não existe um algoritmo que seja eficiente para todos os problemas, e então é razoável supor que algoritmo genético tenha melhor eficiência em determinadas situações. Essa falta de um algoritmo universal é comparado em (SPALL, 2003) a busca de uma agulha no palheiro sem nenhum dado sobre sua posição. Para essa situação a busca aleatória seria um dos melhores métodos de busca e em média nenhum outro seria mais eficiente.

Essa falta de um algoritmo universal para solução dos problemas de otimização parte do teorema da inexistência do almoço grátis (NFL - *No-Free-Lunch Theorem*) proposto por **Wolpert**. O NFL ainda afirma que todos os algoritmos de busca tem em média o mesmo desempenho (LINDEN, 2008). O que de fato deve ser chamado a atenção é que apesar do apelo do algoritmo comparado com a evolução da espécies, se houver um método específico para determinado problema definitivamente ele será mais eficiente que o GA.

Na Figura 3 GOLDBERG mostra que no espectro de problemas de busca e otimização, um método especializado em determinado problema sempre terá a maior eficiência e que os métodos aleatórios e enumerativos apesar de serem considerados menos eficientes podem ter desempenho melhor que o método específico em um espectro mais amplo. O método robusto representa um algoritmo idealizado para solução dos problemas em todas as variedades de problemas. Seria interessante ter um algoritmo robusto, mesmo abandonando o pico de eficiência pois assim atenderia uma variedade de problemas. Como tal algoritmo não é conhecido o que pode ser feito é usar métodos híbridos combinando mais de uma técnica.

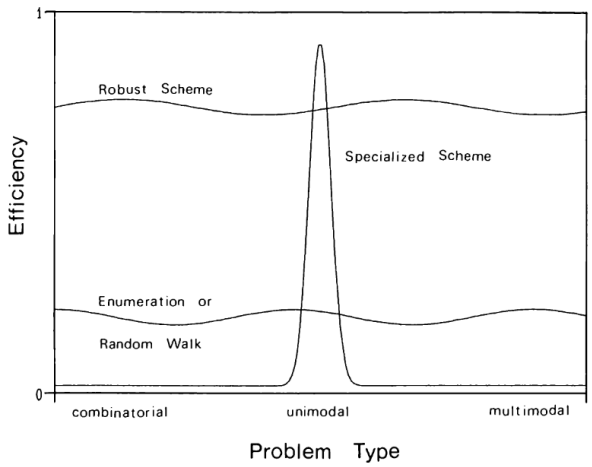

Figura 3 – Exemplo para NFL - (GOLDBERG, 1989)

2 Algoritmos Genéticos

O algoritmo genético (GA) parte de uma população de *cromossomos* que representam as soluções candidatas para o problema de otimização. Cada uma das soluções é avaliada de acordo com critérios inerentes do problema, para posteriormente serem selecionados e combinados de forma a criar novas soluções candidatas.

Já é possível perceber uma das vantagens do algoritmo, ao fazer uma avaliação direta das soluções de forma paralela. Por exemplo em problemas NP-Completos onde são difíceis de se obter soluções numéricas eficientes, mas é possível testar soluções em tempo polinomial, o algoritmo se torna atrativo. O teste de múltiplas soluções facilita o processamento paralelo aumentando a eficiência do algoritmo.

O algoritmo genético mais simples deverá conter pelo menos uma forma de avaliar os elementos da população, uma forma de seleção baseada nas avaliações e operadores genéticos para gerar a nova população.

LINDEN afirma que quanto mais conhecimento especifico sobre o problema for incorporado ao GA, melhor eficiência ele terá. Assim ao definir os operadores que serão utilizados é importante ter em mente que eles devem ser adequados ao problema e não ao contrário.

2.1 Codificação dos genes

Cada cromossomo da população possui uma sequência de genes, que representam os parâmetros para solução do problema. Cada gene terá um possível conjunto de valores, os alelos no equivalente biológico, que para o GA será um alfabeto de valores $\mathcal{A}$, ou um intervalo caso trabalhando com genes de valores reais. O alfabeto mais simples é o binário com $\mathcal{A} = \{0, 1\}$, e é muito utilizado pela simplicidade e por ter sido originalmente a escolha de HOLLAND para explicar os esquemas, além disso para ele cromossomos de maior comprimento e com menos alelos geravam mais paralelismo intrínseco. Porém essa questão dos esquemas e paralelismo vem levantado questionamentos sobre sua real contribuição para o algoritmo.

Com o alfabeto binário, os parâmetros são codificados usando sequências binárias, que podem ser representadas por *strings*. Isso envolve conversões dos parâmetros para binário e vice versa. Os operadores de crossover e mutação ficam simplificados ao utilizar esse tipo de codificação. Uma variação da codificação binária é usar o código gray, que evita o chamado abismo de Hamming[1]

[1] O abismo de Hamming é o efeito que para mudar o valor inteiro em uma unidade na representação

A representação dos genes em termos gerais deve ser o mais simples e o mais natural possível ao problema. Então é comum usar outras formas de representar os genes, por exemplo, em um problema de escolha da melhor rota é interessante representar os grafos no cromossomo, com isso uma sequência de números inteiros representando cada vértice seria o mais apropriado. Também tem sido muito utilizada a codificação dos genes como números reais, representando assim diretamente os parâmetros do problema, e adaptando então as técnicas de mutação e crossover para se adequar a essa nova codificação. A representação com números reais é muito presente em problemas de otimização para redes neurais, onde o cromossomo é o conjunto de parâmetros de ajuste da rede.

Quando possível é interessante impor na codificação as restrições que são impostas ao conjunto de soluções. Por exemplo, ao utilizar um alfabeto binário para representar valores inteiros, ou reais, pode ser feita a conversão de maneira que os valores máximos e mínimos da representação binária coincidam com os valores permitidos dos parâmetros após a decodificação dos genes. Com isso mais conhecimento é sobre o problema é agregado ao algoritmo melhorando sua resposta. Nem sempre esses limites serão possíveis e será vista outra maneira de impor as condições ao algoritmo através da avaliação.

2.2 População, Avaliação e Seleção

2.2.1 População

O conjunto de soluções candidatas é definido como a população do algoritmo, que será modificada a cada iteração através dos operadores genéticos. A cada uma dessas populações geradas no instante t será dado o nome de geração. O tamanho da população é um parâmetro para o algoritmo, e quanto maior, mais soluções serão testadas em paralelo, porém mais recursos computacionais serão usados para cada geração. No algoritmo mais simples, a população é de tamanho fixo e toda substituída na próxima geração, e em outras formas do algoritmo ela pode ter o tamanho variável.

Não existe um número idealizado para o tamanho da população sendo que boa parte dos trabalhos usam 100 como um parâmetro inicial. LINDEN indica que um bom começo seria usar $40 * p$ onde p é a quantidade de parâmetros codificados em nosso cromossomo, porém para codificações de grafos ou ordenamentos já não seria prático usar esse tipo de escolha para o tamanho da população.

É importante também definir dois conceitos presentes na literatura sobre os GA, que são o *exploitation* e o *exploration*. O *exploitation* (aproveitamento) é o fato de explorar cada um dos indivíduos da população pelos potenciais resultados que podem gerar através dos descendentes. Já *exploration* (exploração) é o conceito de se manter a diversidade

binária é necessário alterar todos os bits, por exemplo para ir do número 7 (0111) para o 8 (1000)

genética da população de forma a explorar o espaço de soluções de forma completa. HOLLAND indica que deve existir um equilíbrio entre os dois conceitos na população, pois deve-se explorar ao máximo cada solução encontrada que poderá gerar novas soluções melhores, e da mesma maneira manter os horizontes sobre o espaço de soluções aberto para a descoberta de novas soluções, evitando a convergência genética.

Defini-se convergência genética como sendo o fato dos cromossomos presentes na população começarem a ficar todos com a codificação similares, levando a entender que o algoritmo atingiu o objetivo, ou pelo menos um máximo ou minimo local. Algumas GAs criam operadores de comparação entre os cromossomos de forma a quantificar essa proximidade como parâmetro para encerramento do algoritmo, mas o mais comum é executar o processo de iteração por T gerações. A dificuldade de usar a comparação entre os cromossomos está no custo computacional, que dependendo da forma de codificação usada pode ficar muito complexo. LINDEN menciona que uma forma de verificar essa convergência é usando o algoritmo de agrupamento K-Means e e depois comparar a distância entre os centróides gerados pelo algoritmo.

Sobre a população podem existir pequenas variações sobre como é evoluída durante o algoritmo. Uma dessas alterações é o **elitismo**, que separa os k melhores indivíduos para sobreviverem na próxima geração. Isso é usado devido ao fato de que os operadores de crossover e mutação poderem destruir esses indivíduos na geração da próxima geração, perdendo assim os bons resultados alcançados. Essa mudança melhora o equilíbrio entre os conceitos de *exploitation* e *exploration* e consequentemente o desempenho do algoritmo, pois mantemos o *exploitation* sobre os k indivíduos preservados entre as gerações, e mantemos o *exploration* dando liberdade de escolha arbitrária para os demais $n - k$ indivíduos da população, onde normalmente $n \gg k$.

Outra estratégia para as populações é chamada de $\mu + \lambda$, onde são criados λ cromossomos gerados por μ indivíduos da população atual (geralmente $\mu < \lambda$). Feito isso os $\mu + \lambda$ cromossomos pais e filhos competem para serem selecionados apenas os μ melhores indivíduos para formar a nova população. Essa técnica também pode acelerar a convergência genética, devido ao fato da seleção de apenas os melhores que podem apresentar pouca variação genética, e que para ser compensado deveria ser empregado alguma técnica para manter a diversidade.

Outra alteração possível é denominada *Steady state*, onde em vez de haver a substituição de todos os n indivíduos da população anterior para a nova (ou dos $n - k$ no caso do elitismo) vão sendo criados poucos indivíduos a cada iteração substituindo de forma aleatória os piores pais, isto é, seleciona-se com mais probabilidade os pais com pior avaliação para serem substituídos. Dessa forma existirá uma interação entre cromossomos da geração t com as da geração $t + 1$. Um porém de usar essa modificação é de acelerar a convergência genética, pois os indivíduos com pior avaliação sempre serão substituídos

de forma rápida, mesmo os recém criados, e também o fato de um cromossomo poder reproduzir com o cromossomo que o gerou, deverá gerar um cromossomo muito parecido com os dois, limitando o *exploitation* desses indivíduos.

Para populações com tamanha variável, basicamente duas técnicas podem ser empregadas, uma para considerar um tempo de vida para cada indivíduo levando em conta su avaliação sobre a média da população, e a outra considerando a variabilidade genética. O método que considera a idade dos indivíduos, mantém eles na população por quanto tempo foi determinado na avaliação, e podem levar a condições que a população cresça indefinidamente ou que seja extinta, sendo necessário um controle extra sobre o tamanho da população.

A técnica levando em conta a diversidade, aumenta a população caso perceba-se, através de algum tipo de medição, que os cromossomos da população são semelhantes. Nesse caso alguns cromossomos gerados aleatoriamente podem ser acrescentados a população de modo a manter a exploração do espaço de soluções. Porém como todas as técnicas que devem comparar semelhança entre cromossomos pode ser custosas computacionalmente.

A população inicial geralmente é iniciada criando cromossomos de forma aleatória, porém pode ser feito de forma a subdividir o espaço de soluções em n partes, sendo n o tamanho da população, e gerando um cromossomo aleatório para cada uma dessas partes. (LINDEN, 2008).

2.2.2 Avaliação

Para executar uma seleção sobre os cromossomos da população é necessário primeiro existir uma função de avaliação, que deverá fornecer um escalar, de forma a atribuir para cada cromossomo presente na população uma espécie de nota para sua adaptação. O comum nos algoritmos é usar funções de avaliação que são sempre positivas, para o uso no método de seleção da roleta viciada. Alguns cuidados devem ser tomados ao definir a função de avaliação, pois se ela não apresentar variações suficientes para separar os cromossomos que melhor se adaptam dos demais, o algoritmo irá convergir mais lentamente, e se o contrário ocorrer e a função de avaliação tiver valores muito elevados, teremos uma convergência genética rápida demais impedindo a exploração do espaço de soluções podendo ficar restrito a um máximo local.

De uma forma geral consideremos uma função de avaliação f e que a probabilidade de selecionar o individuo i para reprodução seja definida por

$$p = \frac{f_i}{\sum f}$$

Assim quanto melhor a avaliação de um individuo sobre a média da população, maior é o número esperado de vezes que ele será selecionado. Para evitar que alguns cromossomos,

considerados como **superindivíduos** por LINDEN, dominem a próxima geração, devido a uma avaliação muito superior a média dos demais cromossomos, alguns métodos de transformação sobre a função de avaliação podem ser empregados.

O primeira técnica é a de normalização linear onde pode transformar a função de avaliação na forma $f' = a * f + b$, onde a é normalmente um valor escolhido entre 1 e 2. GOLDBERG indica selecionar esses parâmetros de forma que se mantenha o valor médio de f e o valor máximo de f' seja duas vezes o valor médio. Essa transformação pode gerar problemas pois dependendo dos valores médio, máximo e mínimo e as escolhas dos coeficientes, a função f' pode assumir valores negativos que prejudicam a seleção pelo método da roleta viciada que será visto na subseção 2.2.3 sobre seleções. Esse problema pode ser contornado usando outros coeficientes para evitar tal condição passando a se balizar por manter o mínimo de f' em zero enquanto se mantém o valor médio de f' igual ao valor médio de f.

Outra técnica é o escalonamento Sigma, que procura manter a pressão na seleção constante através das gerações e ao mesmo tempo evitando a convergência genética prematura da população. Assim usando esse método, o valor esperado de vezes que o individuo será selecionado é definido por

$$E[i,t] = \begin{cases} 1 + \frac{f(i) - \overline{f}(t)}{2\sigma(t)} & \text{, se } \sigma(t) \neq 0 \\ 1.0 & \text{, se } \sigma(t) = 0 \end{cases}$$

onde $E[i,t]$ é o valor esperado do individuo i no instante t, *f(i)* é a função de avaliação de i, $\overline{f}(t)$ é a média da função de avaliação da população no instante t e $\sigma(t)$ é o desvio padrão da função de avaliação da população no instante t. Com essa configuração, mesmo cromossomos que se destaquem demais nas gerações iniciais, devido ao desvio padrão da população ser maior nesse momento, eles não irão dominar a próxima geração, e conforme vai ocorrendo um equilíbrio entre os cromossomos nas gerações futuras e perto da convergência, os que ainda tiverem uma melhor avaliação, serão destacados dos demais para terem maior probabilidade de seleção. Caso o valor esperado se torne negativo para algum individuo, adota-se um valor pequeno como 0.1 permitindo um chance relativamente muito pequena para o individuo se reproduzir. (MITCHELL, 1996)

Em alguns problemas as restrições poderão ser definidas limitando os valores dos parâmetros presentes no cromossomo. Porém em alguns casos isso não é possível, principalmente onde, por exemplo, o espaço de soluções não seja convexo. Assim nesses casos é interessante deixar que as soluções sejam avaliadas também fora do espaço de soluções mas aplicar uma penalidade na função de avaliação. Essa penalidade pode ser proporcional ao desvio da solução do espaço permitido, ou uma penalidade constante, o importante é que com a penalidade esse cromossomo seja menos provável de ser selecionado para a próxima geração. O atrativo de permitir esses cromossomos na população, é que mesmo estando fora do domínio do problema, ele pode apresentar um caminho para

evolução de soluções melhores, assim se mesmo com a penalidade ele continua com uma boa avaliação, ele deve conter componentes que podem ser aproveitados nos descendentes (*exploitation*).

2.2.3 Seleção

Na etapa de seleção do algoritmo, serão separados os cromossomos para reprodução e consequente geração da próxima população a ser testada. Para tanto o comum é se usar uma componente estocástica para escolha dos pais, e uma determinística que define a probabilidade de escolha de cada pai. A determinística deriva da função de avaliação, que irá definir qual o valor esperado, ou probabilidade de seleção de cada cromossomo presente na população atual. Deve-se levar em consideração de que os métodos apresentados a seguir, podem incorporar o elitismo descrito anteriormente, e selecionar uma quantidade de pais que gere apenas os cromossomos restantes para completar a população.

- **Roleta Viciada**

  O método mais comum e mais simples encontrados no GAs é o da roleta viciada. Nesse método cada cromossomo recebe a probabilidade de ser escolhido definido como $p(i) = f(i)/\sum_{j=0}^{n} f(j)$, com $i = 1, 2, \ldots, n$, sendo n o tamanho da população e f a função de avaliação. Nesse método que percebe-se a necessidade de $f > 0$ e a preocupação com cromossomos que tem valores de avaliação destacados perante a média levarem a uma convergência genética precoce. Imagina-se uma roleta que para cada cromossomo é separado uma área proporcional a sua probabilidade de escolha definida pela avaliação, e gira-se a roleta para a seleção de um dos cromossomos. Por exemplo, sendo os valores de avaliação para uma população de 6 cromossomos presentes na Tabela 1, obtêm-se a roleta mostrada na Figura 4.

	f	p
Cromossomo 1	152	28,84%
Cromossomo 2	38	7,21%
Cromossomo 3	42	7,97%
Cromossomo 4	5	0,95%
Cromossomo 5	234	44,40%
Cromossomo 6	56	10,63%
Total	527	100,00%

Tabela 1 – Exemplo avaliação

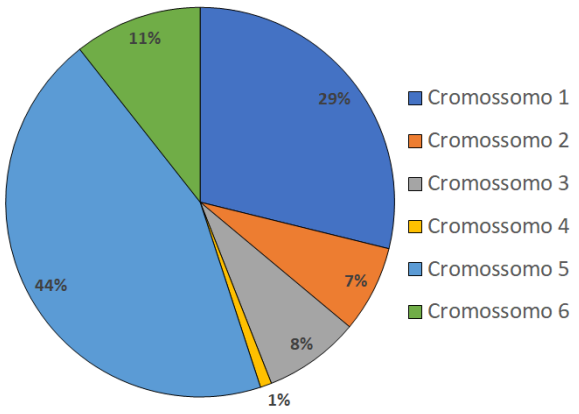

Figura 4 – Exemplo distribuição em uma roleta viciada

A implementação se torna simples, pois não há necessidade de ordenação dos cromossomos, e um algoritmo simples para a seleção pode ser visto em Algoritmo 1, considerando a função *random()* como um gerador de números aleatórios entre 0 e 1. Como pode ser visto, sorteia-se um número entre 0 e o valor da soma das avaliações da população, depois é feito um *loop* iterando pelos valores das avaliações de cada cromossomo e somando a uma variável auxiliar. Ao atingir o valor sorteado, o ultimo individuo que teve seu valor somado a variável auxiliar é retornado.

Entrada: População de cromossomos: populacao
Saída: Cromossomo selecionado: cromossomo

```
1 valor = random() * somaAvaliacao(populacao);
2 aux = 0;
3 i = 1;
4 enquanto aux < valor e i <= populacao.tamanho() faça
5 |   aux = aux + calculaAvaliacao(populacao[i]);
6 |   i = i + 1;
7 fim
8 retorna populacao[i-1]
```

Algoritmo 1: Roleta viciada

- **Amostragem Estocástica Uniforme**

  Uma variação do método da roleta viciada é o Amostragem Estocástica Uniforme (SUS - *Stochastic Universal Sampling*), onde em vez de girar a roleta N vezes, é feito apenas um sorteio. O método consiste em alinhar os valores de avaliação dos indivíduos em uma reta contínua, com segmentos proporcionais as avaliações dos cromossomos e normalizados para que a reta tenha tamanho 1. Sorteia-se um número i entre 0 e $1/n$, onde n é a quantidade de pais que serão selecionados, e depois são gerados n ponteiros para reta na forma $i + j/n$ com $j = 0, 2, \ldots, n-1$. Os cromossomos são selecionados conforme esses ponteiros caiam sobre seu segmento correspondente na reta e colocados em uma lista. Os cromossomos da lista devem ser embaralhados ou sorteados de forma uniforme sem reposição para dar sequência com os operadores genéticos. Isso é necessário pois como os segmentos são contínuos, os cromossomos que forem selecionados mais de uma vez estarão na sequência na lista. Esse método, diferente da roleta, garante que cada individuo será selecionado para reprodução um número de vezes no intervalo $[\lfloor E[i,t] \rfloor, \lceil E[i,t] \rceil]$, onde $E[i,t] = \frac{f_i}{\sum f} \cdot n$ é o valor esperado de vezes que o cromossomo i será escolhido na geração t, com f sendo a função de avaliação e $\sum f$ a soma das avaliações da população. Importante ressaltar que esse método não impede a questão dos superindivíduos dominarem o processo de seleção podendo levar a convergência genética prematura.

- **Seleção por Torneio**

Nessa forma de seleção é através do método do torneio. Nesse modelo de seleção, são sorteados dois oi mais cromossomos da população com probabilidade uniforme, depois é feito um confronto direto entre os valores de avaliação de cada um, sendo que o que tiver maior valor será o selecionado. Esse mecanismo de seleção evita o favorecimento de indivíduos dominantes da população evitando a convergência genética prematura. Seja k a quantidade de cromossomos selecionados para cada rodada do torneio, e N o tamanho da população, a probabilidade de seleção de cada um é dada por $1/N$. Sendo assim a probabilidade de selecionar o pior indivíduo passa a ser $1/N^k$), pois para ele ser selecionado para a próxima geração deve ser o único na rodada do torneio. Em testes empíricos esse método de seleção apresentou resultados melhores que o da roleta quando $k = 2$, sendo que não é sensível a questões de escalas da função de avaliação e também ao fato da função de avaliação ser negativa. (LINDEN, 2008)

- ***Ranking***

  Outro método é a seleção por *ranking*, que também busca prevenir o convergência genética muito rápida, e consiste em ordenar os indivíduos por suas avaliações e depois usar este ranking para definir o valor esperado de cada um, em vez do valor da avaliação. Assim como no método de torneio, não é necessário se preocupar com a escala da função de avaliação. A relação entre os indivíduos i e o $i+1$ será a mesma independente dos valores avaliados para cada um.

  Para cada cromossomo é classificado com os valores de 1 a N indo do pior avaliado para o melhor. O valor esperado de cada cromossomo será dado por

  $$E[i,t] = \min + (\max - \min)\frac{rank(i,t) - 1}{N - 1}$$

  onde max é o valor esperado para o melhor indivíduo e min para o pior. Como há as restrições de $\max \geq 0$ e $\sum_i E[i,t] = N$, pois deve-se manter a população, é necessário então que $1 \leq \max \leq 2$ e $\min = 2 - \max$.

  O valor proposto por Baker é $\max = 1,1$, e esse método tem a desvantagem de diminuir a pressão seletiva, o que pode levar ao GA uma convergência mais lenta, porém mantém a diversidade garantindo uma melhor exploração do espaço de soluções. Uma forma de manter a pressão seria usar uma função exponencial para definir os valores esperados de cada cromossomo. Seja qual for a forma de mapeamento dos valores esperados, depois pode ser utilizado o método SUS ou da roleta viciada para selecionar os pais da próxima geração.(MITCHELL, 1996)

- **Seleção Boltzmann**

  Esse método é inspirado no *simulated annealing*, onde variando a temperatura de forma continua, pode-se controlar a taxa de seleção. O algoritmo inicia com uma

temperatura alta, garantindo que todos na população tem maior probabilidade de reprodução, reduzindo a pressão seletiva, e conforme as iterações do algoritmo são executadas, a temperatura é reduzida gradativamente, aumentando assim a pressão seletiva, e restringindo para apenas os melhores continuarem a se reproduzirem. Uma implementação tipica do modelo é

$$E[i,t] = \frac{e^{f_i/T}}{\frac{\sum_j e^{f_j/T}}{N}}$$

onde T representa a temperatura, e N o tamanho da população, assim no denominador da expressão é representada a média da população no instante t. (MITCHELL, 1996)

2.3 Operadores genéticos

Definidos como serão a população e a codificação dos cromossomos, além dos métodos de avaliação e seleção do GA, deve-se agora selecionar quais operadores genéticos serão usados. Para cada operador há um parâmetro associado que define uma probabilidade de uso do operador a cada etapa do algoritmo, e os operadores podem ser combinados. Os parâmetros podem ser fixos ou variarem conforme a evolução do algoritmo. Lembrando que com o uso do elitismo, teremos indivíduos que passarão para a próxima geração passar pelos operadores genéticos.

2.3.1 Crossover

O operador mais característico e diferencial do algoritmo genético é o de crossover ou recombinação. É com esse operador que dois cromossomos tem suas características combinadas gerando um novo indivíduo que potencialmente terá avaliação melhor que seus pais. Na literatura temos exemplos de implementação do operador onde há um parâmetro que determina a probabilidade de ser executado o crossover sobre um par de cromossomos selecionados, porém há também exemplos em que sempre é executado o operador. No caso de ser usado esse parâmetro, ele normalmente é alto entre 0,8 e 1, e se após o sorteio ficar definido que não será usado o operador, um dos pais é passado sem alterações para as próximas etapas.

- **Crossover de um ponto**

  Esse é o operador mais simples de crossover, onde é sorteado um ponto de corte dentro do cromossomo para ocorrer a "quebra" da sequência dos genes. O cromossomo é formado por uma sequência n de genes, e entre esses genes há $n-1$ pontos de corte que definem as posições onde o cromossomo poderá ser interrompido para ser

combinado com outra parte. O processo é feito após a seleção de dois pais, é feito um sorteio para o ponto de corte. Pode ser gerados dois novos indivíduos a partir desse ponto, o primeiro combinando o material genético do pai A a esquerda do ponto de corte e o material genético do pai B a direita do ponto, e o segundo com o inverso. Algumas implementações aproveitam os dois novos indivíduos para a nova geração, outros ainda realizam um sorteio com igual probabilidade de escolha para um dos dois novos cromossomos.

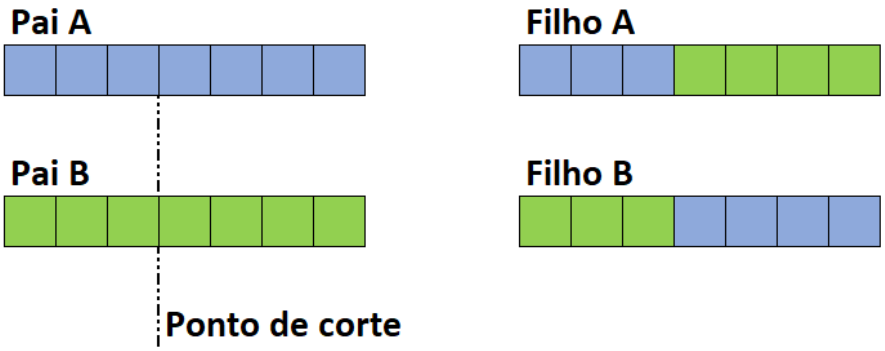

Figura 5 – Crossover com um ponto

Existem dois detalhes sobre o crossover de um ponto, primeiro se uma boa solução do problema estiver codificado nos genes extremos do cromossomo, a chance de manter esse dois genes no novo indivíduo reduz muito, pois o novo elemento da população terá a parte inicial do cromossomo ou a parte final somente. Isso também leva ao segundo problema, pois existem uma certa "preferência" no método para determinadas posições do cromossomo, já que as partes trocadas entre os pais sempre contém os extremos. (MITCHELL, 1996).

- **Crossover de dois ou mais pontos**

  Para melhorar as questões levantadas pelo crossover de um ponto, surgiram as derivações para terem mais pontos de corte no operador. O processo se mantém o mesmo, no entanto em vez de sortear um ponto de corte para executar as trocas de material genético, são sorteados dois ou mais pontos, e os cromossomos pais tem então suas sequências de genes intercaladas entre esses pontos de corte. Um exemplo para dois pontos é apresentado na Figura 6.

  Fazendo o crossover em mais pontos diminuí os problemas encontrados quando utilizado apenas um ponto de corte. Aumenta-se a chance de manter os genes que estejam em locus distantes na sequência do cromossomo que contribuem para boa avaliação do indivíduo, e diminui-se um pouco o detalhe de uso dos extremos, pois pode-se combinar apenas a parte central de um dos pais.

- **Crossover uniforme**

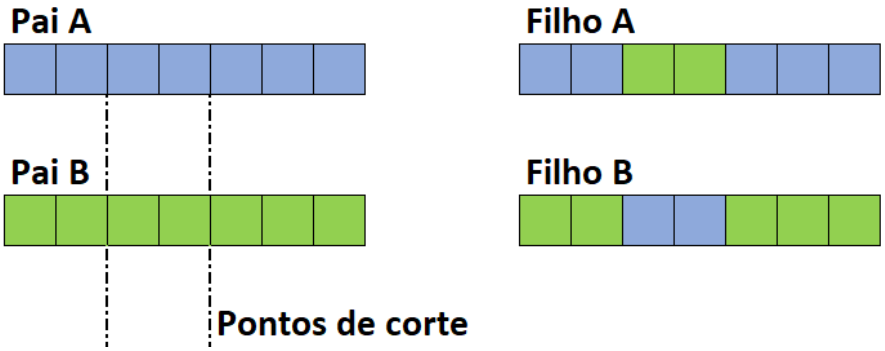

Figura 6 – Crossover com dois ponto

Esse operador é capaz de combinar qualquer esquema presente nos cromossomos. O conceito de esquemas será melhor explorado na seção 2.4, mas para o entendimento do crossover, o esquema pode ser considerado como uma máscara para os genes do cromossomo, fixando alguns desses genes e deixando outros livres. Considerando que os genes fixos no esquema são os responsáveis bela boa avaliação do cromossomo, dependendo da distância das posições desses genes na sequência foi visto que os operadores de crossover usando pontos de cortes podem interromper mais facilmente esses esquemas. Com o crossover uniforme esse efeito é minimizado, permitindo que todos os esquemas tenha probabilidade iguais de serem transmitidos aos filhos. O exemplo simplificado do operador pode ser visto em Figura 7

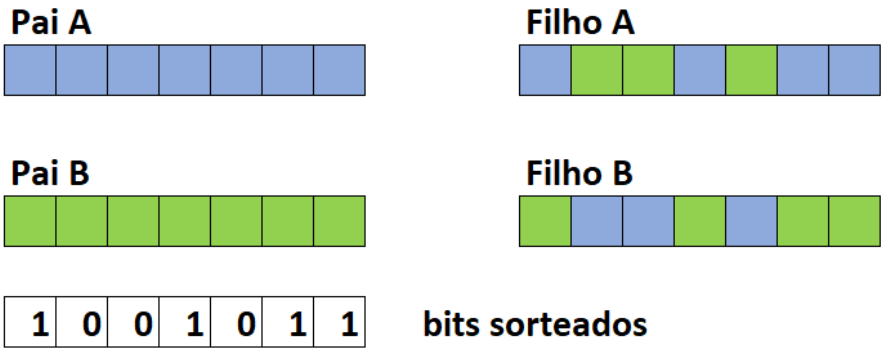

Figura 7 – Crossover uniforme

A diferença com relação aos operadores de combinação anteriores se deve ao fato de que é feito um sorteio dos valores 1 ou 0 com iguais probabilidades para cada posição do cromossomo criando uma sequência binária. Utiliza-se então o resultado do sorteio para definir se será usado o gene do pai A ou do B. Como todas as sequências tem iguais probabilidades de ocorrerem, justifica-se o fato de que os esquemas passam a ter a mesma probabilidade de se manterem durante a reprodução.

MITCHELL menciona uma outra forma de realizar o operador de crossover uniforme, onde seguindo o mesmo raciocínio dos pontos de cortes, onde se realiza um primeiro sorteio para decidir por qual pai começa a construção do filho, e para cada posição

do gene é feito um novo sorteio com igual probabilidade para decidir se continua copiando a sequência do pai atual ou se passa a copiar do outro pai, alcançando os mesmos resultados do procedimento descrito anteriormente.

- **Crossover baseado em maioria**

  O ultimo operador de crossover apresentado será um que combina vários pais simultaneamente, e não é muito utilizado pois pode levar a convergência genética muito rápido. O conceito por trás desse operador é selecionar z pais com $3 \leq z \leq N$, sendo N tamanho da população, e depois definir para cada gene o valor correspondente a maioria. Alternativamente pode ser feito de forma a calcular uma distribuição para cada gene baseado nos valores encontrados no z cromossomos, e depois realizando uma seleção nos moldes da roleta viciada para decidir qual gene utilizar. Esse crossover acaba gerando mais um parâmetro para o algoritmo, pois é necessário definir qual o valor de z

Os operadores descritos podem em geral serem usados com qualquer tipo de codificação, em especial com codificações binárias seu uso são imediatos. Para algumas outras codificações ajustes são necessários. Para representações com reais paras os genes, ou de grafos como na programação genética, os operadores de crossover que usam pontos de corte ou o uniforme funcionam normalmente, pois definido os pontos de corte basta executar os intercâmbios dos valores situados entre eles.

Para cromossomos baseados em ordem, isto é, a ordem da sequência apresentada no cromossomo que representa o fenótipo do indivíduo e consequentemente sua avaliação, usar o crossover de um ponto ou mais pode gerar um problema. Como o operador se baseia na posição dos genes para fazer a concatenação, e nesse tipo de cromossomo as posições dos genes mudam, é necessário adaptar o operador. A forma mais simples de realizar essa adaptação é, em vez de intercambiar diretamente os valores entre os pontos de corte, mantém-se os valores de um dos pais selecionado, mas utiliza-se a ordem para esses valores encontradas no segundo pai. Por exemplo na Figura 8, podemos ver que o pai A possui na região entre os dois pontos de corte a ordem 7 - 1, e esses mesmos valores estão na ordem 1 - 7 no pai B, assim o operador muda a ordem dos valores nesse pedaço para ficar com a mesma ordem do pai B e o resultado é visto no filho A. No filho B o inverso ocorre, sendo que na região de corte é usada a ordem que os valores aparecem no pai A. Pode ser feita uma adaptação análoga para o crossover uniforme. Para o operador baseado em maioria, pode ser feito inciando o primeiro item da sequência pela maioria presente nos pais selecionados, e nas demais posições ir pela maioria encontrada nos pais que seguem o gene escolhido anteriormente.

Quando a codificação for baseada em números reais, o operador de crossover pode ser modificado para em vez de escolher um dos dois valores do gene presentes nos pais,

Figura 8 – Crossover com codificação por ordem

considerar um deles como mínimo e o outro como máximo e realizar um sorteio uniforme no intervalo para o valor do gene. Isso irá gerar filhos bem distintos dos pais, aumentando o aspecto de *exploration* do algoritmo.

2.3.2 Mutação

O operador de mutação é responsável em aumentar o aspecto de *exploration* do GA, e através dele que é possível explorar novas soluções ainda não testadas. Existe um parâmetro associado a esse operador referente a taxa de mutações presentes no algoritmo, sendo que essa taxa usada é um valor baixo na ordem de 0,5%. A taxa deve ser baixa, pois quanto maior for, mais o algoritmo irá se assemelhar a uma busca randômica. Portanto com probabilidades de mutação nessa ordem garante apenas que novas soluções serão apresentadas quando o algoritmo já estiver convergindo, e assim representa uma alternativa caso o GA estiver convergindo para um máximo local.

O procedimento é simples, para cada gene é feito um sorteio com a probabilidade determinada para a taxa de mutação, caso o sorteio tenha um resultado positivo, o gene em questão será modificado. Essa modificação depende do tipo de codificação que foi utilizado. Para codificação binária há duas opções, inverte-se o estado do bit que deve sofrer a mutação, ou pode-se sortear um novo bit com probabilidade igual para 0 ou 1. Usando essa segunda forma, a taxa de mutação resultante será $p_m \cdot 0,5$, onde p_m é o parâmetro de mutação escolhido.

Sobre a codificação binária, existe um detalhe sobre o operador mutação. Como cada bit tem uma significância para o parâmetro que representa do problema, dependendo do bit que for modificado pode representar "saltos" maiores ou menores no espaço de soluções e consequentemente nos valores da avaliação. Algumas formas foram propostas para contornar esse dilema, uma seria definir que o p_m seja proporcional à significância do bit que será modificado. Outra forma seria primeiro transformar a representação binária na variável do problema, realizar uma mutação sobre esse valor, e depois transformar de volta a variável para binário.

Para codificações em inteiro ou em reais, a mutação será na forma de um sorteio sobre o valor do gene. Esse pode usar alguma distribuição conhecida para toda a faixa dos valores que a variável pode assumir, ou usar uma distribuição normal para definir um valor de desvio para a variável e condicionando o resultado para ficar nos limites, ou como visto anteriormente na seção 2.2, permitir esse valores fora das restrições que serão penalizados pela função de avaliação.

As mutações para codificações que usam ordem podem ser feita de duas formas. A primeira se um gene deverá ser modificado, sorteia-se outro gene aleatoriamente e é feita a permutação dos dois. Na segunda forma defini-se dois pontos no cromossomo e é feita ou uma mistura dos elementos entre esses dois pontos, ou inverte-se a ordem entre esses dois pontos.

Para cromossomos baseados em grafos, ou árvores, a sugestão é remover o ramo selecionada para mutação, e gerar um novo aleatório, seguindo o mesmo processos usado para gerar a população inicial.

É comum usar para o parâmetro de taxa de mutação, um valor variável, que pode ir aumentando a cada iteração do algoritmo, permitindo assim que conforme vai se atingindo a convergência genética, o GA não perca sua característica de *exploration*, buscando soluções totalmente novas.

2.3.3 Outros operadores

Existem outros operadores menos usados no algoritmo como por exemplo o de inversão proposto por Holland e alguns que operam sobre como são formados os pares para reprodução.

HOLLAND propôs um operador que inverte parte da sequência do cromossomo, isso pois percebeu a questão do crossover de um ponto que têm uma maior probabilidade em interromper esquemas com genes mais afastados na sequência. Esse operador é inspirado pelo que acontece na genética real, onde a função do gene não depende da sua posição, portanto invertendo parte do cromossomo, sua essência seria mantida. Para adaptar esse operador no GA, além do valor do gene, o cromossomo deve armazenar qual a sua posição original também, o que acarreta uma maior custo computacional. O funcionamento básico do operador é selecionar dois pontos aleatórios no cromossomo, e inverter a ordem dos genes encontrados nesse intervalo. Quando foi realizar o crossover, como os pais selecionados podem ter a ordem dos genes alteradas, utiliza-se um deles como referência e ordena-se o segundo da mesma forma, assim o crossover não correrá o risco de repetir genes na concatenação das partes. Testes com inversão foram feitos mas nenhum obteve resultados indicando que o uso desse operador melhora o desempenho do GA. (MITCHELL, 1996).

Além desse operador, foram feitos alguns experimentos com operadores de seleção

que procuram determinar outros fatores na seleção dos pais, como por exemplo não selecionar dois indivíduos que tenham o genótipo parecido, evitando assim convergência genética prematura e buscando que a combinação entre pais geram filhos diferentes dos encontrados na população até o momento.

Também existem as versões dos operadores apresentados para cromossomos diplóides, onde o genótipo possui um par de cromossomos, sendo que deve ser utilizado o aspecto de dominância dos alelos para definir o fenótipo. A maior diferença está no operador de crossover, pois por serem indivíduos diplóides, ocorre primeiro a separação do par de cromossomos em gametas, feito o crossover e depois recombinação entre os gametas dos pais selecionados. Em (GOLDBERG, 1989) é possível verificar uma análise mais detalhada sobre esse assunto.

2.4 Esquemas

O conceito de esquema foi primeiro definido por HOLLAND, em uma tentativa de especificar por qual razão o algoritmo genético funciona e pode ser eficiente apresentado por GOLDBERG como teorema fundamental do GA e o chamou de Teorema dos esquemas (*Schema Theorem*). Por trás desse conceito, Holland ainda explica a teoria do paralelismo implícito presente no algoritmo razão da qual ele apresentar certa eficiência se comparado a outros algoritmos.

Para explicar o conceito primeiro define-se que o cromossomo é uma sequência de tamanho l de genes que podem ter os alelos presentes no alfabeto

2.5 Fundamentos teóricos

3 Implementação do algoritmo genético

Para implementação do algoritmo foi utilizado a linguagem de programação JAVA de forma a aproveitar o paradigma de orientação a objetos e reaproveitamento de código. A estrutura do programa permite com classes básicas formar a lógica do algoritmo e com classes derivadas especializar o algoritmo para especificações de cada problema. Dessa forma com pouco código extra sendo feito é possível reaproveitar as classes já definidas, usando o polimorfismo inerente a linguagens OO, é possível reescrever e acrescentar as partes necessárias para embutir conhecimento específico no problema.

3.1 Modelo de Ising

— Inserir uma explicação simples sobre o modelo de Ising

O algoritmo genético simples será usado para apresentar uma solução para modelo de Ising em uma matriz 10×10, com cada nó (spin) representado pelos possíveis valores -1 e 1.

Sendo $\Lambda = [-5, 5]^2$ as posições da matriz, e $\sigma_t \in \{-1, 1\}$ representando os valores de cada spin em determinada posição da matriz, e seja $(\sigma_t)_{t\in\Lambda} = \sigma$ um conjunto de valores da matriz.

A função de avaliação que deve ser minimizada é dada por $H(\sigma) = -\sum_{<t,t'>} \sigma_t \sigma_{t'}$ onde t e t' representam dois spins vizinhos.

3.2 Codificação

A classe cromossomo será usada para representar cada solução para o problema e conterá um vetor de valores inteiros de tamanho 100, representando assim a matriz 10×10, alinhando uma linha da matriz na sequencia da outra no vetor. Assim o nosso genótipo é uma cadeia de inteiros, onde o fenótipo será a matriz correspondente quebrando o vetor em linhas de tamanho 10.

Exemplo da relação genótipo x fenótipo para uma matriz 3×3:

$$\text{cromossomo} = [-1, 1, 1, 1, -1, 1, 1, 1, -1]$$

$$\text{fenótipo} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

3.3 Seleção e avaliação

3.4 Crossover e mutação

3.5 Resultados

4 Conclusão

Referências

COELLO, C. C.; LAMONT, G. B.; VELDHUIZEN, D. A. van. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. New York, NY: Springer, 2007. ISBN 978-0-387-36797-2. Disponível em: <https://www.amazon.com/Evolutionary-Algorithms-Multi-Objective-Problems-Computation-ebook/dp/B00DZ75DNE?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00DZ75DNE>. Citado 3 vezes nas páginas 9, 21 e 22.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1. ed. Reading, MA: Addison-Wesley Professional, 1989. ISBN 0201157675. Disponível em: <https://www.amazon.com/Genetic-Algorithms-Optimization-Machine-Learning/dp/0201157675?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0201157675>. Citado 4 vezes nas páginas 9, 26, 27 e 33.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. 2. ed. Cambridge, MA: MIT Press Ltd, 1992. ISBN 0262581116. Disponível em: <https://www.ebook.de/de/product/2864942/john_h_holland_adaptation_in_natural_and_artificial_systems.html>. Citado 4 vezes nas páginas 26, 29, 31 e 42.

JACOBSON, B. K. L. *Genetic Algorithms in Java Basics*. New York, NY: Springer-Verlag GmbH, 2015. Disponível em: <https://www.ebook.de/de/product/25481443/lee_jacobson_burak_kanber_genetic_algorithms_in_java_basics.html>. Citado na página 25.

KLUG, W. S. et al. *Concepts of Genetics*. São Francisco, CA: Benjamin Cummings, 2011. ISBN 978-0-321-72412-0. Disponível em: <https://www.amazon.com/Concepts-Genetics-William-S-Klug/dp/0321724127?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321724127>. Citado 2 vezes nas páginas 9 e 24.

KWONG, S.; MAN, K.-F.; TANG, K.-S. *Genetic Algorithms*. London: Springer London, 2001. ISBN 1852330724. Disponível em: <https://www.ebook.de/de/product/3753808/sam_kwong_kim_fung_man_kit_sang_tang_genetic_algorithms.html>. Citado na página 25.

LINDEN, R. *Algoritmos Geneticos*. 2. ed. Rio de Janeiro: Brasport, 2008. ISBN 978-8574523736. Citado 9 vezes nas páginas 23, 25, 27, 29, 30, 31, 32, 33 e 36.

MITCHELL, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. 1. ed. Cambridge, MA: The MIT Press, 1996. ISBN 0262133164. Disponível em: <https://www.amazon.com/Introduction-Genetic-Algorithms-Complex-Adaptive/dp/0262133164?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0262133164>. Citado 8 vezes nas páginas 25, 26, 33, 36, 37, 38, 39 e 42.

SIVANANDAM, S.; DEEPA, S. N. *Introduction to Genetic Algorithms*. New York, NY: Springer, 2007. ISBN 978-3-540-73189-4. Disponível em: <https://www.amazon.com/Introduction-Genetic-Algorithms-S-N-Sivanandam/dp/354073189X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=354073189X>. Citado na página 23.

SPALL, J. C. *Introduction to Stochastic Search and Optimization*. Hoboken, NJ: John Wiley & Sons, 2003. ISBN 0471330523. Disponível em: <https://www.ebook.de/de/product/2876128/spall_stochastic_optimization.html>. Citado na página 27.

Apêndices

Anexos