

Fabricio Kassardjian

Algoritmos Genéticos

Um algoritmo evolucionário para otimização

Brasil

26 de novembro de 2019

Fabricio Kassardjian

Algoritmos Genéticos
Um algoritmo evolucionário para otimização

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

Universidade de São Paulo – USP
Instituto de Matemática e Estatística
Bacharelado em Matemática Aplicada e Computacional

Orientador: Anatoli Iambartsev

Brasil
26 de novembro de 2019

Fabricio Kassardjian

Algoritmos Genéticos

Um algoritmo evolucionário para otimização/ Fabricio Kassardjian. – Brasil, 26 de novembro de 2019-

73p. : il. (algumas color.) ; 30 cm.

Orientador: Anatoli Iambartsev

Monografia – Universidade de São Paulo – USP

Instituto de Matemática e Estatística

Bacharelado em Matemática Aplicada e Computacional, 26 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. Título

Fabricio Kassardjian

Algoritmos Genéticos

Um algoritmo evolucionário para otimização

Monografia de final de curso, Instituto 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
26 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 performance 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

Figura 1 – Técnicas de otimização global - (COELLO; LAMONT; VELDHUIZEN, 2007)	22
Figura 2 – Exemplo de reprodução com crossover - (KLUG et al., 2011)	24
Figura 3 – Exemplo para NFL - (GOLDBERG, 1989)	27
Figura 4 – Exemplo distribuição em uma roleta viciada	35
Figura 5 – Crossover com um ponto	37
Figura 6 – Crossover com dois ponto	38
Figura 7 – Crossover uniforme	39
Figura 8 – Crossover com codificação por ordem	40
Figura 9 – Representação dos esquemas como hiperplanos em um espaço tridimensional. (GOLDBERG, 1989)	44
Figura 10 – Gráfico da função de avaliação	54
Figura 11 – Distribuições das soluções encontradas nos algoritmos de busca aleatória e GA	57
Figura 12 – Exemplo de uma região (vermelha) com baixa energia, próxima de outras com energias superiores	61
Figura 13 – Distribuições das populações para gerações 0, 10, 20 e 40, com $p_m = 0,02$	62
Figura 14 – Distribuições das populações para gerações 0, 10, 20 e 40, com $p_m = 0,$	63
Figura 15 – Valores das médias e variâncias da população durante as gerações com $p_m = 0,005$. Os valores apresentados são da média de 500 testes	64
Figura 16 – Evolução do algoritmo genético durante gerações	65
Figura 17 – Evolução do algoritmo genético durante gerações	73

Lista de quadros

Quadro 1 – Resultado de um ciclo de seleção, crossover e mutação do teste preliminar 55

Lista de tabelas

Tabela 1 – Exemplo avaliação	35
Tabela 2 – Resultados teste simples com alguns parâmetros	56
Tabela 3 – Resultados da média de energia para 1000 testes com modelo Ising alterando tipo de crossover	60
Tabela 4 – Resultados da média de energia para 1000 testes com modelo Ising alterando outros parâmetros	61
Tabela 5 – Resultados da média de energia para 1000 testes com modelo Ising e usando elitismo mantendo 3 cromossomos	62
Tabela 6 – Resultados da média de energia para 1000 testes das gerações 0, 10, 20, 30 e 40 do modelo Ising e usando elitismo com 3 cromossomos	62
Tabela 7 – Resultados da média de energia para 1000 testes com modelo Ising usando elitismo mantendo 3 cromossomos	63
Tabela 8 – Resumo da geração 9 com média de avaliação 185,86 mostrando a taxa de amostragem da seleção e o que era esperado para a proporção usando a função de avaliação da Equação 3.1	71
Tabela 9 – Resumo da geração 5 com média de avaliação 3.516 mostrando a taxa de amostragem da seleção e o que era esperado para a proporção usando a função de avaliação da Equação 3.2	72

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	21
1.1	Algoritmos evolucionários	21
1.2	Biologia	23
1.3	Surgimento do algoritmo genético	25
1.4	Teorema da inexistência do almoço grátis	27
2	ALGORITMOS GENÉTICOS	29
2.1	Codificação dos genes	29
2.2	População, Avaliação e Seleção	30
2.2.1	População	30
2.2.2	Avaliação	32
2.2.3	Seleção	34
2.3	Operadores genéticos	36
2.3.1	Crossover	37
2.3.2	Mutação	40
2.3.3	Outros operadores	42
2.4	Fundamentos teóricos	43
2.4.1	Esquemas	43
2.4.2	Teorema fundamental do GA	45
2.4.3	Two-armed bandit	46
2.4.4	O GA como uma cadeia de Markov	48
2.4.5	Mecânica estatística	50
2.5	Problemas associados ao GA	50
3	IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO	53
3.1	Implementação inicial	53
3.2	Modelo de Ising	56
3.3	GA para o modelo de Ising	58
4	CONCLUSÃO	67
	REFERÊNCIAS	69
	ANEXO A – RESULTADOS	71

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 $g_i(x) \leq 0, i = \{1, \dots, m\}$, e $h_j(x) = 0, j = \{1, \dots, n\}$ com $x \in \Omega$. A solução maximiza, ou minimiza, o escalar $f(x)$ onde x é um vetor com dimensão n , $x = \{x_1, x_2, \dots, x_n\}$ 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

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

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.

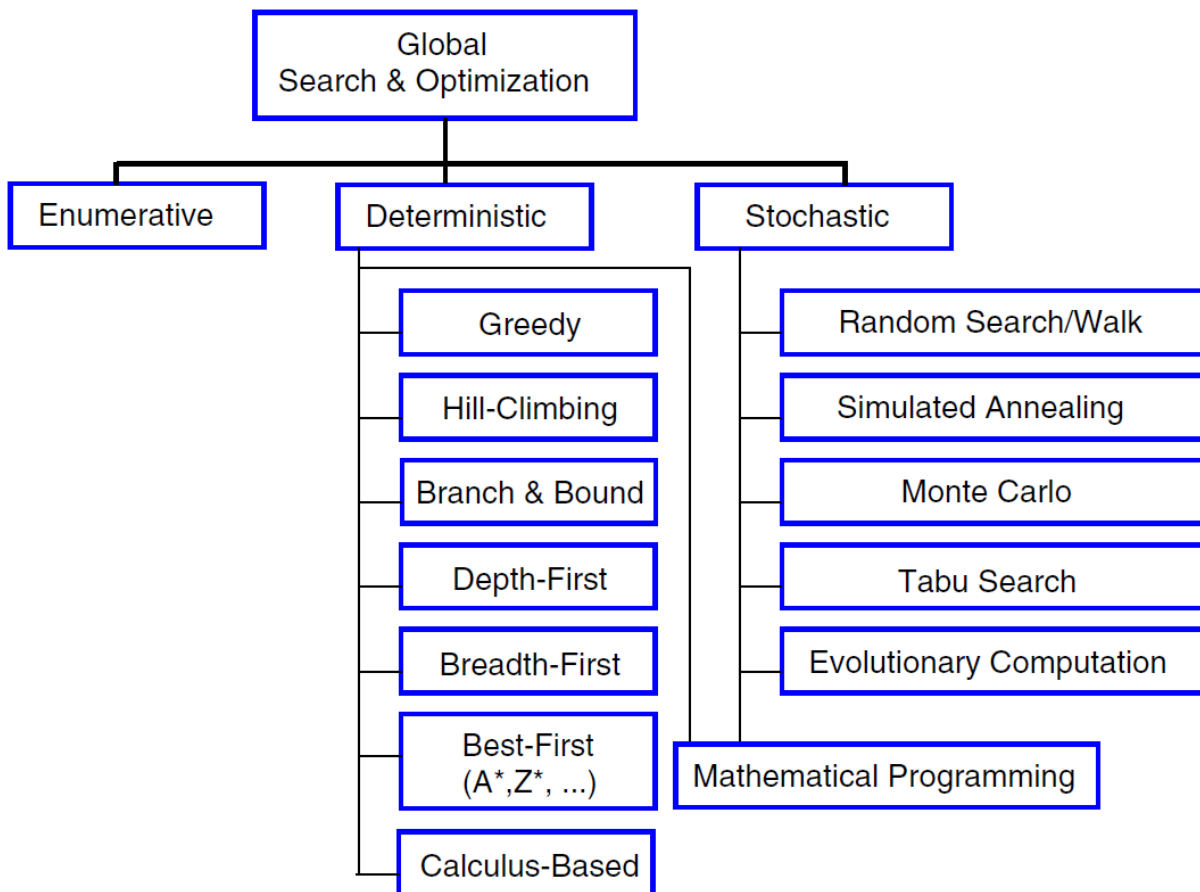


Figura 1 – Técnicas de otimização global - (COELLO; LAMONT; VELDHUIZEN, 2007)

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 indivíduo 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 indivíduo 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 início 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 dividido em gametas que então são combinados com os gametas do outro pai para formar novamente um cromossomo diplóide completo, gerando assim os novos indivíduos. Para organismos haplóides é feita apenas a combinação das sequências de cada pai.

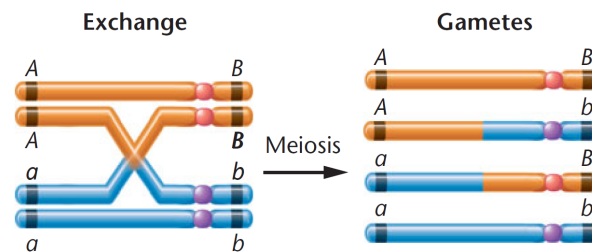


Figura 2 – Exemplo de reprodução com crossover - (KLUG et al., 2011)

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 associada 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 indivíduo, 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. Esses 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 específico 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 estrutura 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 seguem 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 indivíduo 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.

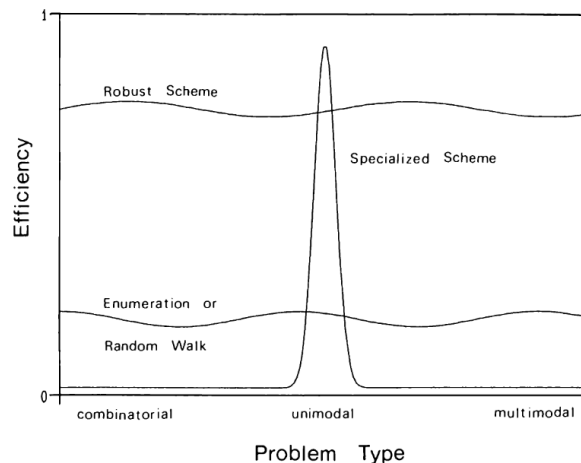


Figura 3 – Exemplo para NFL - (GOLDBERG, 1989)

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-Completo 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 específico 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 levantando 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 Hamming¹

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*² e o *exploration*³. O *exploitation* (aproveitamento) é o fato de

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

² Exploitation consiste em testar uma região limitada mas promissora do espaço de soluções com a expectativa de melhorar uma solução já conhecida dentro dessa região

³ Exploration consiste em testar uma região muito mais ampla do espaço de soluções, com a expectativa

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 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 aproveitar 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 mínimo 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 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 conseqüentemente 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 tamanho variável, basicamente duas técnicas podem ser empregadas, uma para considerar um tempo de vida para cada indivíduo levando em conta sua 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 indivíduo i para reprodução seja definida por

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

Assim quanto melhor a avaliação de um indivíduo 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 indivíduo será selecionado é definido por

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

onde $E[i, t]$ é o valor esperado do indivíduo i no instante t , $f(i)$ é a função de avaliação de i , $\bar{f}(t)$ 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 indivíduo, adota-se um valor pequeno como 0.1 permitindo um chance relativamente muito pequena para o indivíduo 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

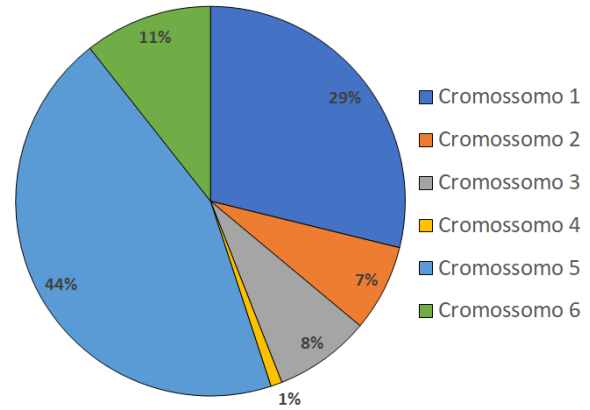
O método mais comum e mais simples encontrados no GAs é o da roleta viciada. Nesse método cada cromossomo recebe a probabilidade de ser escolhido definido como $p(i) = f(i) / \sum_{j=0}^n f(j)$, com $i = 1, 2, \dots, n$, sendo n o tamanho da população e f a função de avaliação. Nesse método que percebe-se 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](#).

A implementação se torna simples, pois não há necessidade de ordenação dos cromossomos.

• 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 $j = 0, 2, \dots, n - 1$. Os cromossomos são selecionados conforme esses ponteiros caíam sobre seu segmento

	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

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 indivíduo será selecionado para reprodução um número de vezes no intervalo $[E[i, t], \lceil E[i, t] \rceil]$, 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 ou 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 $i + 1$ 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

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

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 contínua, 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 típica do modelo é

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

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.

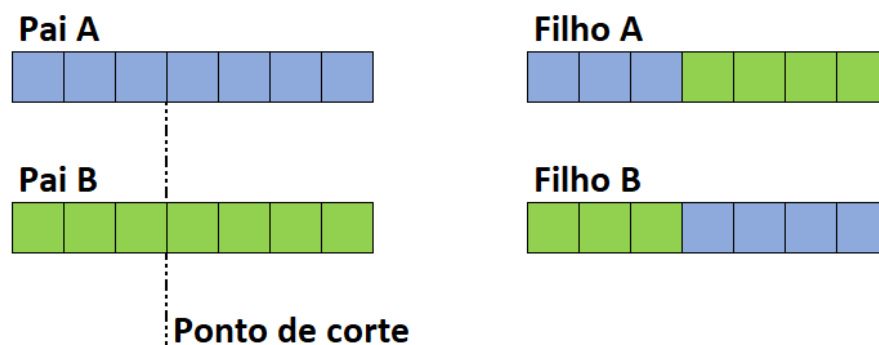


Figura 5 – Crossover com um ponto

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.

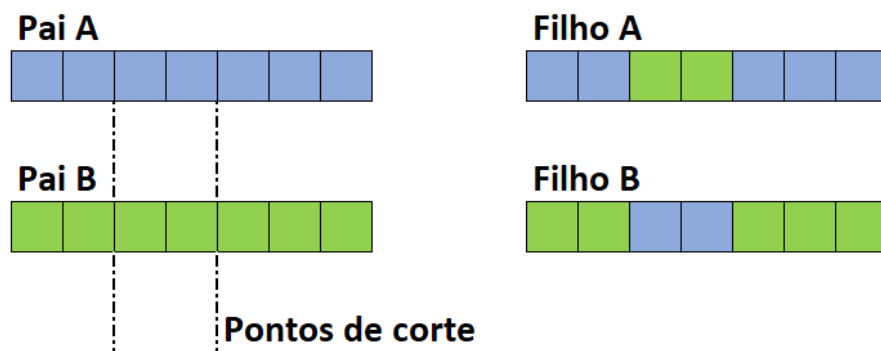


Figura 6 – Crossover com dois ponto

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**

Esse operador é capaz de combinar qualquer esquema presente nos cromossomos. O conceito de esquemas será melhor explorado na subseção 2.4.1, 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 pela 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](#)

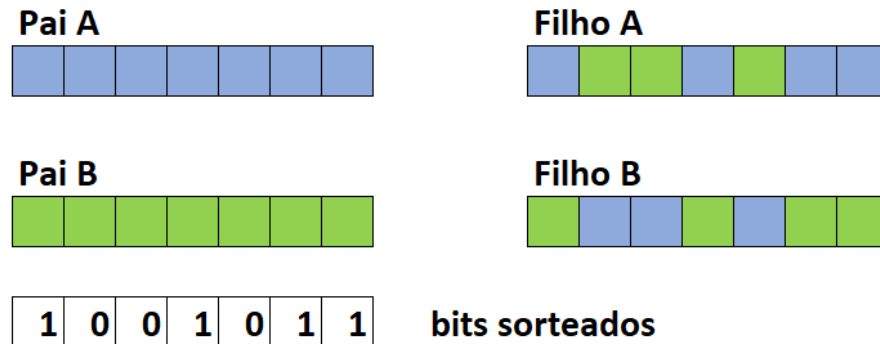


Figura 7 – Crossover uniforme

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 para 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 iniciando 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.



Figura 8 – Crossover com codificação por ordem

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, 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 conseqüentemente 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 Fundamentos teóricos

2.4.1 Esquemas

O conceito de esquema (*schema* ou *schemata*) foi primeiro definido por [HOLLAND](#), em uma tentativa de especificar por qual razão o algoritmo genético funciona e pode ser eficiente. Foi apresentado também por [GOLDBERG](#) como teorema fundamental do GA, e ficou conhecido 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 comprimento l , de genes que podem ter os alelos presentes no alfabeto V . Para simplificar as definições e cálculos e sem perda de generalidade, foi escolhido um alfabeto binário $V = \{0, 1\}$ e os cromossomos são portanto sequências binárias. O esquema funciona como um gabarito ou *template* definindo assim subconjuntos de todos os possíveis indivíduos. Para realizar tal gabarito o esquema usa um outro elemento considerado um coringa, que representa uma posição no cromossomo que não importa (*don't care*). Assim para o alfabeto binário utilizado o alfabeto para os esquemas pode ser definido como $V+ = 0, 1, *$, sendo $*$ nosso valor coringa. Exemplo, para uma população de cromossomos com $l = 8$, o esquema $H = **11*0*1$ tem os indivíduos com cromossomos $A_1 = 00110001$ e $A_2 = 10111011$ como representantes desse esquema e são chamados de instâncias de H . Assim se tivermos cromossomos binários com comprimento l existirão 2^l combinações possíveis de cromossomos e 3^l combinações de esquemas.

Existem duas características importantes sobre os esquemas, uma delas é sua **ordem**, denotada por $o(H)$, que define quantas posições possuem valor fixo no esquemas, isto é, quantas posições no esquema são diferentes de $*$. No exemplo anterior $H = **11*0*1$ possui ordem $4(o(**11*0*1) = 4)$. Outra característica é o **tamanho** do esquema (*defining length*), denotado por $\delta(H)$, que é calculado como a diferença entre as posições do ultimo valor definido no esquema e do primeiro definido. Por exemplo para o esquema $011*1***$ seu tamanho é 4, pois o ultimo valor definido está na posição 5 e o primeiro na posição 1 ($\delta(011*1***) = 4$).

Goldberg usou o simbolo H pois o esquema define um hiperplano do espaço de dimensão l que representa as sequências binárias de comprimento l . Considerando sequências binárias e esquemas de comprimento $l = 3$, é possível desenhar a representação dos hiperplanos de forma geométrica, que pode ser visto na [Figura 9](#). Os vértices são esquemas de ordem 3, as linhas de ordem 2 e os planos de ordem 1.

Avaliando os novos conceitos, pode ser verificado que quanto maior for o tamanho de um esquema ($\delta(H)$), maior será sua probabilidade de ser desfeito com os operadores genéticos de crossover, e quanto maior sua ordem, maior será a probabilidade do operador

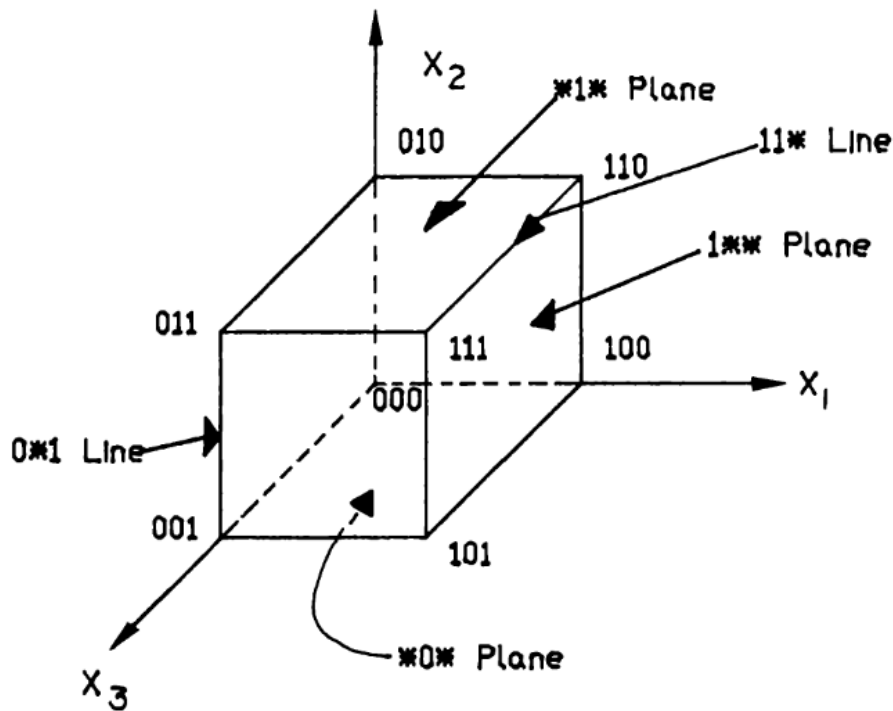


Figura 9 – Representação dos esquemas como hiperplanos em um espaço tridimensional. (GOLDBERG, 1989)

de mutação também destruí-lo. Apesar da probabilidade de modificar alguns dos esquemas presentes e que foram bem avaliados, pois os operadores serão aplicados aos cromossomos selecionados e portanto que possuem melhor avaliação média, existe a chance de com isso encontrar esquemas que responderão melhor ainda para o problema em questão.

Como esquemas de ordem e tamanho maiores estão mais sujeitos a serem destruídos, os esquemas mais curtos e que mais contribuem para a avaliação dos cromossomos, tendem a ser preservados, e esses constituem os blocos de construção (*building blocks*) citados por GOLDBERG.

O que HOLLAND indica como paralelismo intrínseco se atribui ao fato, de que apesar do algoritmo estar avaliando n indivíduos contido na população na geração presente, ele está também avaliando de forma implícita um número bem maior de esquemas. Qualquer sequência binária de comprimento l é uma instância de 2^l esquemas. Por exemplo a sequência 11, é uma instância dos seguintes esquemas: $**$, $1*$, $*1$ e 11 . Estendendo para a população de tamanho n existem portanto um total de até $n \cdot 2^l$ esquemas, assim o total de esquemas em determinada população será um valor entre 2^l , caso todos os cromossomos sejam iguais, e $n \cdot 2^l$, caso todos diferentes. Por essa razão o GA avalia uma quantidade de esquemas maior do que a quantidade de indivíduos presentes na população.

2.4.2 Teorema fundamental do GA

A definição do teorema dos esquemas de acordo com (LINDEN, 2008) é "o GA tende a preservar com o decorrer do tempo aqueles esquemas com maior avaliação média e com menores ordem e tamanho, combinando-os como blocos de montar de forma a buscar a melhor solução".

O teorema pode ser verificado através da estimação do aumento ou diminuição das instâncias dos esquemas presentes na população. Consideremos uma população Ω de tamanho N de cromossomos com codificação binária e comprimento l . Seja H um esquema que tenha ao menos uma instância na população na geração t , e $m(H, t)$ o número de instâncias de H no instante t . Definimos

$$\hat{u}(H, t) = \frac{\sum_{x \in H} f(x)}{m(H, t)}$$

onde x é um indivíduo na população, e $f(x)$ é a função de avaliação. Assim $\hat{u}(H, t)$ é a média das avaliações das instâncias de H no instante t .

Relembrando que a probabilidade de um indivíduo ser escolhido para seleção com o método da roleta viciada é definido por $p(i) = f(i) / \sum_{j=0}^N f(j)$, com $i = 1, 2, \dots, n$. Sendo $n(i, t)$ a quantidade de vezes que o cromossomo i é selecionado na população na geração t , o valor esperado de n será $E[n(i, t)] = N \cdot p(i)$. Se $\bar{f}(t) = \sum_{i \in \Omega} f(i) / N$ é a média das avaliações da população no instante t , o valor esperado de $n(i, t)$ pode ser expresso como $E[n(i, t)] = f(i) / \bar{f}(t)$

O valor esperado da quantidade de instâncias de H em $t + 1$ considerando apenas a seleção descrita e sem usar os operadores de crossover e mutação será dado por:

$$\begin{aligned} E[m(H, t + 1)] &= \frac{\sum_{x \in H} f(x)}{\bar{f}(t)} \\ &= \frac{\hat{u}(H, t)}{\bar{f}(t)} m(H, t) \end{aligned} \quad (2.1)$$

Assim é verificado que a quantidade de instâncias do esquema H na próxima geração depende diretamente da média das avaliações das instâncias do esquema na geração atual e pode ser concluído que esquemas com avaliação maior que a média irão aumentar ao passo que os avaliação menor diminuirão.

Como visto antes, os operadores de crossover e mutação podem tanto construir como destruir as instâncias do esquema H , e que a probabilidade de destruir é proporcional as características do esquema dado pela ordem ($o(H)$) e pelo tamanho ($\delta(H)$). Considerando apenas o fato destrutivo dos operadores, para o caso do crossover de ponto único, com p_c sendo a probabilidade de ser aplicado, o limite inferior da probabilidade de sobrevivência

do esquema pode ser estimado como:

$$S_c(H) \geq 1 - p_c \frac{\delta(H)}{l-1}$$

e pela equação é verificado que quanto menor o tamanho do esquema, maior sua probabilidade de sobrevivência.

Já para o operador de mutação, considerando p_m como a probabilidade de um bit sofrer mutação, a probabilidade de sobrevivência do esquema é dada por:

$$S_m(H) = (1 - p_m)^{o(H)}$$

e novamente quanto menor a ordem do esquema, maior probabilidade de sobrevivência.

Por fim, a Equação 2.1 pode ser alterada para definir um limite inferior dado por:

$$E[n(H, t+1)] \geq \frac{\hat{u}(H, t)}{\bar{f}(t)} m(H, t) \left(1 - p_c \frac{\delta(H)}{l-1}\right) [(1 - p_m)^{o(H)}] \quad (2.2)$$

A Equação 2.2 define o Teorema dos Esquemas proposto por HOLLAND e que pode ser visto também em Goldberg (1989).

2.4.3 Two-armed bandit

HOLLAND menciona que a adaptação possui uma tensão entre os componentes de *exploitation* e *exploration*, e que a análise de esquemas demonstrava, assumindo certas condições, que o GA atingi um certo equilíbrio próximo do ótimo. A permutação entre os dois podem ser comparados com o “*Two-armed bandit problem*”⁴. (MITCHELL, 1996)

O problema é descrito como um jogador com N moedas e uma máquina caça-níquel com duas alavancas, sendo que uma delas (A_1) paga com média μ_1 e desvio-padrão σ_1 e a outra (A_2) paga com média μ_2 e desvio-padrão σ_2 . Esses valores não variam, são independentes e são desconhecidos ao jogador. O objetivo do jogador é maximizar os lucros, porém como não tem informações sobre qual das alavancas paga mais, só pode estimar realizando jogadas nas duas.

MITCHELL descreve uma solução simples para o problema seguindo o que foi proposto por HOLLAND e que pode ser visto em detalhes em Holland (1992). Considerando sem perda de generalidade que $\mu_1 > \mu_2$, ou seja, a alavanca A_1 paga mais que a A_2 , e sendo $A_h(N, N-n)$ e $A_l(N, n)$ as alavancas com maior e menor valores pagos observados respectivamente, após N tentativas, sendo $N-n$ tentativas usadas em A_h e n tentativas na A_l . O que se busca é o valor $n = n^*$ que maximize o lucro. A melhor estratégia seria colocar as N apostas na alavanca que paga mais, porém como ela não é conhecida isso não é possível.

⁴ Uma máquina caça-níquel é também conhecida como *One-armed bandit*

Duas possibilidades podem ocorrer, ou A_l corresponde a pior alavanca e nesse caso houve prejuízo somente nas n apostas feitas em A_l (as perdas dadas por $(N - n)(\mu_1 - \mu_2)$), ou A_l corresponde a melhor alavanca e então houve prejuízo nas $N - n$ apostas colocadas em A_h (perdas dadas por $n(\mu_1 - \mu_2)$). Seja q a probabilidade de A_l ser na verdade a melhor alavanca A_1 , isto é, $q = P(A_l(N, n) = A_1)$, assim as perdas $L(N - n, n)$ após N tentativas é dado por:

$$L(N - n, n) = q(N - n)(\mu_1 - \mu_2) + (1 - q)n(\mu_1 - \mu_2)$$

Como o desejado é $n = n^*$ que minimiza essa função (ou seja maximiza o lucro), tomando a derivada com relação a n e igualando a zero.

$$\frac{dL}{dn} = (\mu_1 - \mu_2) \left(1 - 2q + (N - 2n) \frac{dq}{dn} \right) = 0$$

Que para ter uma solução é necessário expressar q em função de n . Sejam S_1^n a soma recebida pelas apostas em A_1 e S_2^{N-n} a soma recebida pelas apostas em A_2 . Então q pode ser expreso na forma:

$$\begin{aligned} q &= P \left(\frac{S_2^{N-n}}{N - n} > \frac{S_1^n}{n} \right) \\ &= P \left(\left(\frac{S_2^{N-n}}{N - n} - \frac{S_1^n}{n} \right) < 0 \right) \end{aligned} \quad (2.3)$$

que indica de fato a probabilidade de observar que A_2 pagou mais que A_1 . Como S_1^n e S_2^{N-n} são variáveis aleatórias, sua diferença também é uma v.a. e a probabilidade descrita em [Equação 2.3](#) é a área abaixo da curva de distribuição na parte menor que zero. [HOLLAND](#) aproximou essa probabilidade usando o teorema central do limite assumindo a distribuição normal. Na segunda edição, Holland atribui a correção dessa aproximação para Dan Frantz, que usou a teoria dos grandes desvios (*theory of large deviations*), para aproximar essa probabilidade e chegar ao resultado:

$$n^* \approx c_1 \ln \left(\frac{c_2 N^2}{\ln(c_3 N^2)} \right)$$

onde c_1 , c_2 e c_3 são constantes positivas. Rearranjando os termos obtêm-se:

$$N - n^* \approx e^{n^*/2c_1} \sqrt{\frac{\ln(c_3 N^2)}{c_2}} - n^*$$

Conforme n^* aumenta, o termo exponencial domina a equação, e pode-se aproximar mais ainda obtendo:

$$N - n^* \approx e^{cn^*}$$

que indica em resumo que o valor ótimo de apostas $N - n^*$ feitas à melhor alavanca observada deve aumentar exponencialmente conforme o número de apostas feitas na pior alavanca observada. (MITCHELL, 1996)

O que o Teorema dos Esquemas sugere é que da mesma forma que o problema descrito “*two-armed bandit*” que pode ser estendido a um problema com k alavancas (“*k-armed bandit*”), o GA aumenta de forma exponencial as tentativas feitas com os melhores esquemas observados com relação as tentativas feitas aos piores esquemas. GOLDBERG indica que na verdade o GA deve ser comparado a uma composição de *k-armed bandits*.

MITCHELL faz uma ressalva com a comparação com relação ao fato de que no caso do *two-armed bandit*, as alavancas são independentes, que não é o caso do GA, pois os esquemas interagem entre eles, pois por exemplo, o valor observado de retorno do esquema $111 * \dots *$ tem influência sobre o valor observado do retorno de $1 * * * \dots *$.

2.4.4 O GA como uma cadeia de Markov

O teorema dos esquemas versa sobre a evolução dos esquemas presentes no algoritmo, porém para tentar determinar exatamente fatores como a distribuição da população, das avaliações e da convergência do algoritmo conforme as iterações, foram procurados outros modelos matemáticos para determinar o comportamento do GA. Um desses modelos descritos por NIX; VOSE, modela o GA na forma de uma cadeia de Markov, para uma população finita. Assim o estado atual é definido pela presente população, que através de uma matriz de transição Q define a probabilidade de ir para uma nova população dada a atual.

Seja o espaço de soluções definido por Ω formado por cromossomos codificados como sequências binárias de comprimento l , assim $r = 2^l$ define a quantidade de soluções possíveis, e seja P uma população de elementos de Ω de tamanho n . N é a quantidade de possíveis populações que podem ser formadas e pode ser calculado como:

$$N = \binom{n + 2^l - 1}{2^l - 1}$$

que de fato é a combinação com repetição de n elementos de um conjunto com 2^l elementos.

Define-se a matriz Z com tamanho $r \times N$ cuja as colunas representam todas as possíveis populações de tamanho n . Cada coluna é um vetor coluna das incidências de cada elemento y de Ω na população P_i , isto é, cada elemento $z_{y,i}$ representa o número de ocorrências do indivíduo y na população P_i , com y sendo o indexador do indivíduo começando com 0. A matriz Z define assim os possíveis estados da cadeia de Markov.

Agora seja Q a matriz de transição de tamanho $N \times N$, onde $Q_{i,j}$ é a probabilidade de a partir da população P_i passar para a população P_j . Para determinar as probabilidades de transição, defini-se $p_i(y)$ como a probabilidade de o indivíduo y ser gerado a partir de P_i através dos operadores genéticos de seleção e recombinação (crossover e mutação). Na próxima população P_j , y está presente o número de vezes determinado em $z_{y,j}$, e para determinar a probabilidade, Nix e Vose enumeraram as formas possíveis de gerar as sequências presentes em P_j .

A quantidade de formas de selecionar $z_{0,j}$ vezes o indivíduo 0 dentro da população de tamanho n é dado por $\binom{n}{z_{0,j}}$, e sobram $n - z_{0,j}$ posições na população para serem preenchidas. Seguindo para o indivíduo 1, a quantidade de formas de selecionar ele $z_{1,j}$ vezes é definido por $\binom{n-z_{0,j}}{z_{1,j}}$. Continuando da mesma maneira, o total de combinações para todas os indivíduos é dado por:

$$\binom{n}{z_{0,j}} \binom{n-z_{0,j}}{z_{1,j}} \dots \binom{n-z_{0,j}-\dots-z_{2^{l-2},j}}{z_{2^{l-1},j}} = \frac{n!}{z_{0,j}! z_{1,j}! \dots z_{2^{l-1},j}!}$$

Como a probabilidade de produzir a quantidade certa do indivíduo y na população P_j a partir da população P_i é dado por:

$$\prod_{y=0}^{2^l-1} (p_i(y))^{z_{y,j}}$$

A probabilidade de produzir P_j a partir de P_i fica definido como a distribuição multinomial dada por:

$$Q_{i,j} = \frac{n!}{z_{0,j}! z_{1,j}! \dots z_{2^{l-1},j}!} \prod_{y=0}^{2^l-1} (p_i(y))^{z_{y,j}} = n! \prod_{y=0}^{2^l-1} \frac{(p_i(y))^{z_{y,j}}}{z_{y,j}!}$$

Para calcular a probabilidade $p_i(y)$ no artigo de [Vose e Liepins \(1991\)](#) são derivados duas matrizes, F que define a avaliação dos indivíduos em sua diagonal principal e M que define a recombinação do algoritmo genético simples. Seja $\vec{\phi}_i$ o vetor coluna que representa a população P_i e portanto é uma das colunas de Z , e define-se $|\vec{\phi}_i|$ como a soma dos elementos do vetor. Dessa forma a proporção dos indivíduos y na população P_i é dada por $(\vec{\phi}_i / |\vec{\phi}_i|)_y$. A probabilidade de y ser selecionado é dado por

$$\left(\frac{F \vec{\phi}_i}{|\vec{\phi}_i|} \right)_y$$

e o valor esperado da proporção de y na próxima geração dado por

$$\left[M \left(\frac{F \vec{\phi}_i}{|\vec{\phi}_i|} \right) \right]_y$$

e como $p_i(y)$ é equivalente a proporção esperada de y , a matriz de transição então pode ser escrita como

$$Q_{i,j} = n! \prod_{y=0}^{2^l-1} \frac{\left[M \left(\frac{F\vec{\phi}_i}{|F\vec{\phi}_i|} \right)_y \right]^{z_{y,j}}}{z_{y,j}!}$$

Assim a cadeia de Markov fica definida com Z sendo a matriz de estados, e Q a matriz de transições. Com base nas propriedades das cadeias de Markov, [NIX](#); [VOSE](#) derivaram alguns resultados importantes, como por exemplo, quando $n \rightarrow \infty$, a cadeia converge para iterações de G , com probabilidade próxima de 1, onde $G = F \circ M$. Também mostraram que para n suficientemente grande, e se G_p tem um ponto fixo, o GA tende a ficar o tempo todo com a população correspondente a este ponto. Se G_p tem mais de um ponto, o GA gasta um tempo fora desses pontos com tendência a 0. $G_p(\vec{x}) = M(F\vec{x}/|F\vec{x}|)$

Apesar de lucidar alguns pontos sobre o GA, esses modelos são impraticáveis para prever modelos reais por serem muitos detalhados e necessitarem de matrizes muito grandes conforme l e n aumentam.

2.4.5 Mecânica estatística

Uma outra forma de avaliar o GA indicado por [MITCHELL](#) é o uso da mecânica estatística, que em vez de procurar determinar como cada parte define o comportamento do algoritmo, procura ver o GA de uma forma mais macroscópica. Um estudo feito sobre esse tema pode ser encontrado no [Prügel-Bennett e Shapiro \(1994\)](#), onde usaram o modelo de Ising de uma dimensão e o GA para minimizar a energia na grade. O objetivo era verificar e tentar prever o comportamento da distribuição de energia na população conforme as gerações do algoritmo. Para o modelo de uma dimensão, [PRÜGEL-BENNETT; SHAPIRO](#), conseguiram gerar uma previsão sobre os dois primeiros momentos, média e variância, da distribuição de energia na população.

Como será visto no [Capítulo 3](#) foi escolhido o modelo de Ising com duas dimensões, e será feito uma simulação para verificar o comportamento da distribuição da população durante as gerações do algoritmo, variando alguns dos parâmetros do GA para averiguar quais mudanças causam nos resultados.

2.5 Problemas associados ao GA

Existem duas situações conhecidas que levam o GA a ter uma piora de desempenho podendo se tornar ineficiente. Uma delas é chamado de carona (*hitchhiking*), e consiste no fato de que genes vizinhos aos que formam esquemas com alta avaliação, tendem a se manter nas próximas gerações como caronistas. Isso levanta questões sobre o teorema dos

esquemas, dando argumentos aos que se opõe a ele. No exemplo de codificação binária, a ocorrência desses bits que pegam caronas se deve ao operador de crossover, que deveria ter seus pontos de corte sorteados de tal forma a precisamente manter somente os bits que geram a alta avaliação no esquema e remover os caronas, mas que dependendo do comprimento das cadeias de bits e dos esquemas passam a ter probabilidade pequena de ocorrerem.

Outra questão levantada sobre os GAs é com relação aos problemas que foram definidos como enganadores (*deceptives*), que são problemas que apresentam esquemas que não contêm o máximo global com média de avaliação superior aos esquemas que o contêm. [LINDEN](#) descreve esses problemas como tendo o máximo global em forma de pico cercados por valores de avaliação muito baixos, porém da mesma forma que impõem uma dificuldade ao GA, também será difícil para outros métodos.

[SPALL](#) também menciona a questão sobre epistasia, onde genes interagem entre si e combinados produzem um resultado melhor do que avaliados individualmente, o que vai contra a teoria dos blocos de construção de [GOLDBERG](#), onde pequenos blocos com alta avaliação garantem ao GA sua eficiência. Na forma do algoritmo genético simples apoiado no teorema dos esquemas, ficou definido que o GA tende a preservar esquemas com menor ordem e tamanho, mas no caso deveria-se preservar esquemas que podem ter o tamanho grande se os genes que interagem ocupam posições distantes no cromossomo.

Mesmo com o GA apresentando alguns problemas, ele pode ser uma opção interessante, principalmente se o conhecimento do problema permitir que esses efeitos sejam reduzidos, como o posicionamento dos genes no cromossomo, ou o uso de outros operadores, como o de inversão que explora novos posicionamentos dos genes, ou usar operadores modificados de crossover, ou de mutação direcionados. Além disso ainda existe a opção de combinar o GA com outros métodos de otimização locais, por exemplo, usar o método de *hill-climbing* na fase de mutação.

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 com a característica de 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, e usando o polimorfismo inerente a linguagens OO, é possível reescrever e acrescentar as partes necessárias para embutir conhecimento específico no problema.

3.1 Implementação inicial

Os testes começam sobre um modelo simples usando codificação binária para representar duas variáveis reais x e y com valores no intervalo $[-100, 100]$, e 15 bits de comprimento. A conversão é feita usando um fator C definido por $C = \frac{200}{2^{15} - 1}$ que multiplicado pela representação binária transformada para o inteiro i e somado ao valor mínimo, apresenta o valor real da variável, com resolução definida pelo fator, $x = C \cdot i_x - 100$. O cromossomo é formado por uma sequência de 30 bits, 15 para cada variável, concatenados na forma de um vetor (Em Java será utilizado os conceitos de Listas por serem mais versáteis com relação aos elementos que podem ser guardados e manipulados). A função que deve ser maximizada, que será a mesma de avaliação, é dada pela equação seguinte e apresenta o gráfico visto na [Figura 10](#):

$$f(x, y) = \left| x \cdot y \cdot \sin\left(\frac{\pi \cdot y}{4}\right) \right|$$

É possível ver que a função possui 4 pontos máximos definidos dentro do domínio, e utilizando o R para encontrar o máximo da função em um desses pontos, foi obtido os seguintes resultados: $x = 100$, $y = 98,01654$ e $f(x, y) = 9800,827$. O ponto com é crítico de ser encontrado pelo fato de possuir vários máximos locais dentro do domínio. Para comparação dos resultados do GA, sempre será executado uma busca aleatório com a quantidade de pontos gerados equivalentes a $P * T$, onde P é o tamanho da população definido como parâmetro do GA e T o total de gerações que o algoritmo será executado.

Para esse GA foi utilizado o método de seleção da roleta viciada, com um algoritmo simples que 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

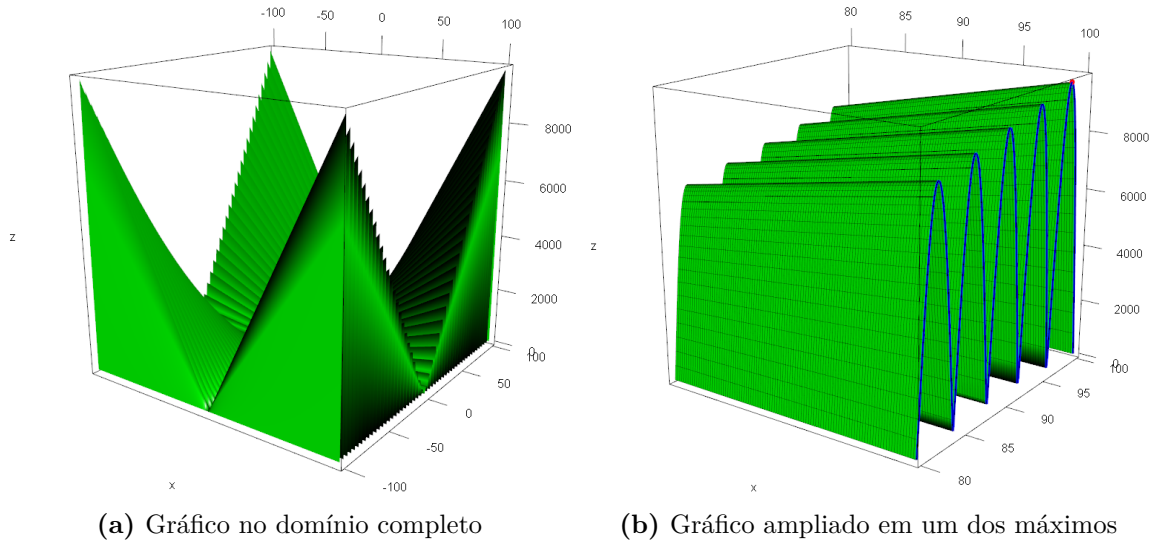


Figura 10 – Gráfico da função de avaliação

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

O operador de crossover testado foi o com único ponto de corte, e seu algoritmo pode ser visto em [Algoritmo 2](#). O operador de mutação é feito apenas invertendo o estado do bit caso o sorteio seja positivo para aquele bit, respeitando o parâmetro de probabilidade de mutação. O critério de parada ficou definido apenas pela quantidade de iterações, isto é, quantas gerações da população foram produzidas. Para a população inicial são gerados cromossomos com bits aleatórios, sendo que é feito um sorteio individual para cada bit. Essa mesma operação é usada para gerar os valores para a busca randômica. Nesse primeiro teste a população é toda substituída por uma nova a cada geração, mas o melhor cromossomos de todas as gerações é guardado e apresentado como resultado final do algoritmo genético.

Seguem alguns resultados para execução do GA, com os parâmetros: $P = 30$, $T = 10$, $p_c = 0,8$, $p_m = 0,01$, onde P é o tamanho da população, T é por quantas gerações o GA irá evoluir, p_c e p_m as probabilidades de crossover e mutação. No [Quadro 1](#)


```

Entrada: Cromossomos: PaiA e PaiB; probabilidade de crossover:  $p_c$ 
Saída: Cromossomo Filho
1 se random() <=  $p_c$  então
2   | corte = random() * PaiA.tamanho();
3   | se random() <= 0.5 então
4   |   | Filho = concatenar(PaiA[0:corte], PaiB[corte:PaiB.tamanho()]);
5   |   | senão
6   |   |   | Filho = concatenar(PaiB[0:corte], PaiA[corte:PaiA.tamanho()]);
7   |   | fim
8   | fim
9   | Filho = PaiA;
10 fim
11 retorna Filho

```

Algoritmo 2: Crossover com um ponto de corte

temos o resultado dos operadores de seleção, crossover e mutação para um descendente. O “*HERITAGE MAP*” mostra qual porção do cromossomo herdou do pai 1 ou 2 e o “*MUTATE MAP*” representa os bits que sofreram mutação. Além disso demonstra quantas vezes cada pai foi selecionado para produzir a nova geração, e com esses dados foi possível verificar os funcionamentos básicos do algoritmo, correspondendo ao esperado.

Quadro 1 – Resultado de um ciclo de seleção, crossover e mutação do teste preliminar

```

Geração: 1 | Média Avaliação: 2847,450535
#### CHILD #####
ID: 31
Fenotipo: x = -13,91949 | y = -94,32356
Avaliação: 1270,770525
#### PARENT 1 #####
ID: 11
Fenotipo: x = -13,91949 | y = -19,02829
Avaliação: 183,079928
Selecionado 1 vezes, 0,03

#### PARENT 2 #####
ID: 13
Fenotipo: x = 38,99960 | y = -94,28694
Avaliação: 3584,173001
Selecionado 6 vezes, 0,20

### HERITAGE MAP ###
11111111111111111111222222222222
### MUTATE MAP
000000000000000000000000000000
#### CHILD #####

```

Na [Tabela 2](#) é feito um comparativo entre o GA e a busca randômica (RS), executados em 1000 testes e calculado médias com intervalo de confiança e qual proporção o GA teve melhor resultado que o RS. Pode ser verificado que com poucas gerações, o GA apresentou resultados melhores que o RS, aumentando a população para 50 $P = 50$ o GA teve ainda uma pequena melhora com relação ao RS, mas aumentando para 100 gerações ($T = 100$) os resultados praticamente são similares considerando os intervalos de confiança, porém percebe-se que ambos começaram a apresentar média de resultados maiores, e o GA supera o RS em menor quantidade de testes. Isso deve ocorrer, primeiro pelo fato do GA já não conseguir pelos operadores normais, refinar a solução, e segundo pelo RS ter cada vez mais amostras e com isso melhorar seu resultado.

Os testes com os outros parâmetros do GA se mostraram significativos também, com $P = 30$, $T = 10$, $p_c = 0,8$, $p_m = 0,05$, o GA apresentou resultados ainda melhores, particularmente com $T = 100$, mostrou expressiva melhora com relação ao teste anterior e ao RS. Isso mostra a necessidade do operador de mutação para garantir a exploração (*exploration*) do espaço de soluções. Os testes alterando $p_c = 0,9$ apresentaram nenhum efeito prático para esse problema, e o destaque ficou na ultima linha da tabela, onde ao usar em conjunto os parâmetros que individualmente melhoraram os resultados, obteve-se soluções muito próximas da ótima e um desempenho bem superior à busca aleatória.

Parâmetros				Resultados GA		Resultados RS		#
P	T	p_c	p_m	Média \pm IC	D.P.	Média \pm IC	D.P.	GA > RS
30	10	0,8	0,01	8693,10 \pm 47,04	758,98	8419,40 \pm 39,28	633,80	659
50	10	0,8	0,01	9084,39 \pm 32,39	522,60	8736,45 \pm 32,43	523,22	716
30	100	0,8	0,01	9369,35 \pm 26,09	421,03	9352,98 \pm 15,41	248,67	578
30	10	0,8	0,05	8995,23 \pm 35,40	571,19	8481,60 \pm 38,72	624,75	741
30	100	0,8	0,05	9670,58 \pm 12,67	204,53	9367,58 \pm 15,16	244,68	852
30	10	0,9	0,01	8729,25 \pm 45,65	736,59	8453,47 \pm 39,33	634,68	623
30	10	0,9	0,05	8997,06 \pm 35,47	572,39	8504,56 \pm 38,22	616,72	735
50	100	0,8	0,05	9739,77 \pm 8,45	136,34	9454,56 \pm 12,07	194,75	927

Tabela 2 – Resultados teste simples com alguns parâmetros

Além disso, pode ser vista a distribuição de um dos testes de busca aleatória, da ultima geração no GA e um comparativo da distribuição em algumas gerações [Figura 11](#). O ponto verde indica o melhor resultado para cada uma das distribuições.

3.2 Modelo de Ising

O modelo de Ising¹ é um modelo da física para estudos de fenômenos magnéticos em materiais. Ising resolveu analiticamente o modelo para uma dimensão e depois

¹ O modelo também é conhecido como Lenz-Ising, pois Lenz introduziu o modelo, porém nunca calculou nenhuma de suas propriedades

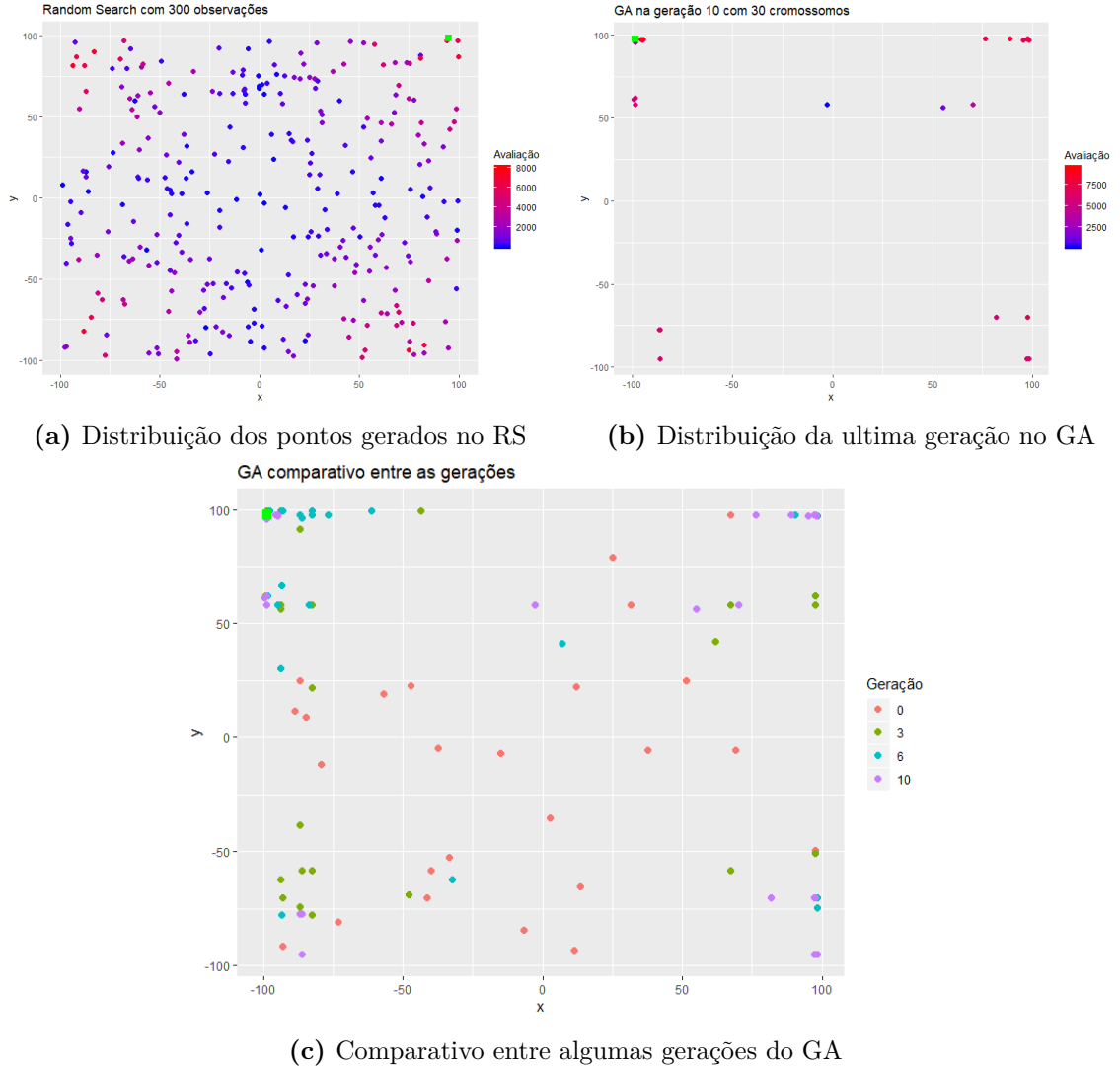


Figura 11 – Distribuições das soluções encontradas nos algoritmos de busca aleatória e GA

Onsanger resolveu para uma rede quadrada (2D). Apesar do modelo ter sido elaborado para compreender melhor as propriedades magnéticas de certos materiais, ele se mostrou de grande utilidade para modelar problemas de outras áreas de estudos de fenômenos cooperativos, como por exemplo a dissertação de [Junior \(2014\)](#) onde usa o modelo de Ising aplicado a um estudo de criminalidade. É considerado um dos mais simples e estudado dos modelos de mecânica estatística.

O modelo define uma grade de *spins* σ_t que podem conter os valores -1 e $+1$ somente. Esses spins possuem uma interação de energia dada por $-J(t, t')\sigma_i\sigma_{t'}$, ou seja, o valor de energia de dois spins será $-J(t, t')$ se os spins forem diferentes, e $+J(t, t')$ se forem iguais, e adicionalmente podem interagir com um campo magnético externo H com energia $-H\sigma_t$. ([MCCOY; WU, 1973](#)).

Seja então $\sigma_t \in \{-1, +1\}$ cada vértice do modelo, Λ representando a grade do modelo, e $\Omega_\Lambda = \{-1, +1\}^\Lambda$ o espaço de possíveis estados do modelo, define-se $\sigma \in \Omega_\Lambda$,

$\sigma = (\sigma_t)_{t \in \Lambda}$ como um estado possível do modelo de Ising para essas configurações. O Hamiltoniano do sistema para o estado σ ($\mathcal{H}(\sigma)$) será dado por:

$$\mathcal{H}(\sigma) = - \sum_{t,t' \in \Lambda} J(t,t') \sigma_t \sigma_{t'} - \sum_{t \in \Lambda} H \sigma_t$$

Sendo $\beta = 1/kT$, com T a temperatura em Kelvin e k a constante de Boltzmann, a função de partição é definida por

$$Z(\beta) = \sum_{\sigma \in \Omega_\Lambda} e^{-\beta \mathcal{H}(\sigma)}$$

e a probabilidade de encontrar o estado σ

$$P(\sigma) = \frac{e^{-\beta \mathcal{H}(\sigma)}}{Z(\beta)}$$

Temperaturas altas deixam β baixo, o que torna a distribuição mais plana, e todos os estados tem probabilidades mais próximas de serem encontrados, conforme a temperatura baixa, os estados de menor energia passam a ter a maior probabilidade de ser encontrado.

O modelo de duas dimensões mais simples usa uma estrutura formada por quadrados com formato retangular de dimensões $\Lambda = n \times m$, facilmente representada por uma matriz. Além disso $J(t,t') = J_{tt'}$ é um valor fixo independente dos spins e que se t e t' não forem vizinhos será igual 0. Por fim será sem campo magnético externo, ou seja, $H = 0$, e assim o Hamiltoniano pode ser reescrito na forma:

$$\mathcal{H}(\sigma) = - \sum_{i=1}^{n-1} \sum_{j=1}^{m-1} J_{tt'} \sigma_{i,j} \sigma_{i,j+1} + J_{tt'} \sigma_{i,j} \sigma_{i+1,j} \quad (3.1)$$

O algoritmo genético simples será usado para apresentar uma solução, isto é σ , para modelo de Ising em uma matriz 10×10 , com $J_{tt'} = 1$ para os spins vizinhos, sendo $\Lambda = [1, 10]^2$ as posições da matriz, e a função de avaliação utilizada será baseada na [Equação 3.1](#).

3.3 GA para o modelo de Ising

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 :

$$\begin{aligned} \text{cromossomo} &= [-1, 1, 1, 1, -1, 1, 1, 1, -1] \\ \text{fenótipo} &= \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \end{aligned}$$

Usando os conceitos de OO, os cromossomos são definidos como classes, encapsulando os genes, os métodos dos operadores genéticos de crossover e mutação, e os métodos para avaliação e classificação dos cromossomos quando necessário. Para outros tipos de verificação também guardou informações sobre os cromossomos pais.

Para o método de seleção foi usado o método da roleta viciada, definindo a proporção de seleção de cada estado presente na população relacionado diretamente a sua função de avaliação. A primeira função testada foi usando o resultado da [Equação 3.1](#) somando o valor de $\min_{\sigma \in \Omega_\sigma}(\mathcal{H}(\sigma))$ e multiplicando por -1 para conter somente valores positivos. Essa função não mostrou bons resultados pois não houve muita pressão seletiva, pois cromossomos com melhores valores recebiam pouco ou nenhum destaque na fase de seleção, assim o algoritmo precisava de muitas gerações para começar um avanço em encontrar melhores soluções. Na literatura sobre GA é mencionado o fato da função de avaliação ser muito plana o algoritmo passa a ser menos eficiente, isso também ocorre com outros métodos de otimização, como por exemplo os que usam gradientes por aproximação numérica, pois em funções que mostram pouco crescimento, a aproximação pode falhar em indicar a direção de maior crescimento. Os resultados para a seleção de uma das gerações pode ser visto na [Tabela 8](#) do [Anexo A](#). Para corrigir isso foi adotada a função de avaliação $e^{-\beta\mathcal{H}(\sigma)}$ o que leva a uma probabilidade de seleção do indivíduo definida por:

$$Pr(\sigma) = \frac{e^{-\beta\mathcal{H}(\sigma)}}{\sum_{\sigma \in \Phi_\Lambda} e^{-\beta\mathcal{H}(\sigma)}} \quad (3.2)$$

Onde $\Phi_\Lambda \in \Omega_\Lambda$ representa a população de estados daquela geração do algoritmo e o método de seleção é parecido com o apresentado como seleção de Boltzmann na [subseção 2.2.3](#). O problema que deve ser definido um novo parâmetro para o algoritmo dado por β , que como valor inicial será usado o 0,05. Os resultados foram muito melhores e podem ser verificados na [Tabela 9](#) do [Anexo A](#).

O operador de crossover será testado com o método de ponto único, dois pontos e uniforme, e para o operador de mutação cada posição do gene será sorteado com a probabilidade p_m para ser eleito para alteração e depois cada estado possível, no caso dois $\{-1, +1\}$, será sorteado com probabilidades iguais para definir o novo estado do gene.

A população inicial é definida sorteando os σ_t de cada posição das matrizes de cada um dos indivíduos. O programa terá definido uma classe para a população que é

responsável por criar a geração inicial aleatória, controlar aspectos sobre a geração como melhor e pior cromossomo e algumas estatísticas sobre a população como quantidade de indivíduos que sofreram mutação e crossover, entre outros.

Da mesma forma que o teste inicial, serão testados algumas variações dos parâmetros do algoritmo e comparado com os resultados da busca aleatória. Além disso será verificado, assim como no mencionado em [Mitchell \(1996\)](#) sobre o trabalho de [Prügel-Bennett e Shapiro \(1994\)](#), as distribuições dos valores de $\mathcal{H}(\sigma)$ para as gerações 0, 10, 20, 30 e 40 do GA, e calculando as estatísticas para média e variância da amostra de 1000 testes. De forma ampliar as comparações dos resultados finais, os dois algoritmos foram modificado para ordenar as populações, que no caso do GA é feito a cada geração.

O começo da [Tabela 3](#) demonstra os resultados mantendo-se os parâmetros fixo e alterando apenas o método usado no crossover, indicado pelo índice nos valores de p_c , com 1 para ponto único, 2 para crossover em dois pontos e u para uniforme. Todos são muitos superiores ao método RS, provavelmente em virtude do espaço de soluções ser dado por $2^{10 \times 10}$ e a quantidade de observações geradas para o método estar cobrir uma parte pequena do espaço. Comparando o resultado entre os 3 métodos de crossover, é possível pela média dos resultados verificar pequena melhora usando o método para 2 pontos e mais uma pequena melhora para o uniforme, mais visível nos testes com mais gerações. Isso pode ser explicado pela construção do genótipo do cromossomos, como cada linha da matriz está na sequência da outra, podemos ter *clusters* com bom resultados de forma quadrada, como mostrado na [Figura 12](#). A região vermelha possui uma baixa energia contribuindo de forma positiva para a avaliação do indivíduo, porém essa área não consegue ser isolada pelos operadores de crossover de 1 ou 2 pontos, mas tem boa probabilidade de ser compartilhada em partes ou inteira através do operador uniforme. Outro fator que piora nessa condição é os genes caronistas, discutidos na [seção 2.5](#), pois a área azul ao apresentar pior resultado tem alta probabilidade de pegar carona com a área vermelha do cromossomo.

Parâmetros				Resultados GA		Resultados RS		#
P	T	p_c	p_m	Média \pm IC	D.P.	Média \pm IC	D.P.	GA < RS
30	10	$0,8_1$	0,02	$-58,90 \pm 0,54$	8,77	$-39,05 \pm 0,33$	5,43	978
30	10	$0,8_2$	0,02	$-60,21 \pm 0,57$	9,33	$-38,98 \pm 0,32$	5,22	970
30	10	$0,8_u$	0,02	$-60,42 \pm 0,63$	10,30	$-39,39 \pm 0,35$	5,68	960
30	100	$0,8_1$	0,02	$-122,24 \pm 0,71$	11,57	$-48,54 \pm 0,27$	4,49	1000
30	100	$0,8_2$	0,02	$-126,05 \pm 0,75$	12,19	$-47,94 \pm 0,27$	4,48	1000
30	100	$0,8_u$	0,02	$-130,47 \pm 0,73$	11,89	$-48,23 \pm 0,28$	4,54	1000

Tabela 3 – Resultados da média de energia para 1000 testes com modelo Ising alterando tipo de crossover

Nos próximos testes fica fixado o crossover uniforme, além disso pela característica do GA, quanto mais iterações do algoritmo permitida, melhores resultados deve apresentar,

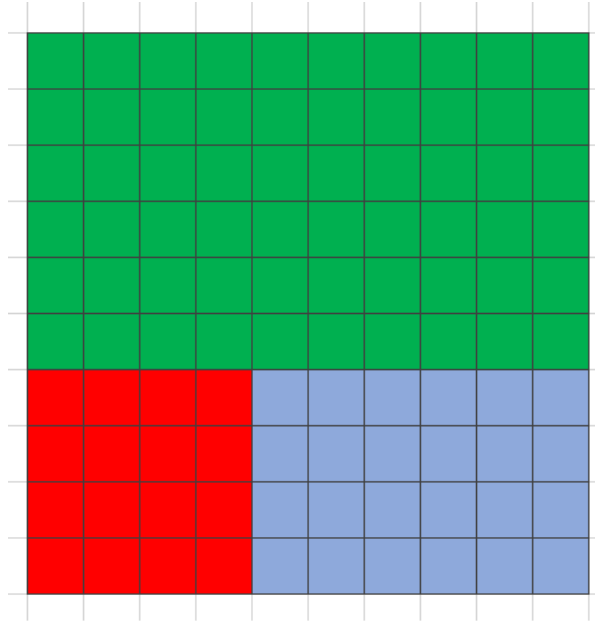


Figura 12 – Exemplo de uma região (vermelha) com baixa energia, próxima de outras com energias superiores

portanto serão avaliados os demais parâmetros temos os resultados apresentados na [Tabela 4](#). Pelos valores de média apresentado é possível ver que os parâmetros $p_c = 0,8$ e $p_m = 0,02$, apresentaram os melhores resultados, quando o p_m é aumentado, como o algoritmo não tem elitismo, é provável que os melhores esquemas se perdem com a mutação. O p_c com valores mais baixos teve um resultado pior, que indica que com menos crossover deve ter ocorrido menos *exploitation* do algoritmo levando a uma convergência mais lenta.

Parâmetros				Resultados GA		Resultados RS		#
P	T	p_c	p_m	Média \pm IC	D.P.	Média \pm IC	D.P.	GA < RS
30	40	0,8	0,02	-105,07 \pm 0,72	11,66	-44,98 \pm 0,31	5,01	1000
30	40	0,8	0,05	-93,90 \pm 0,71	11,52	-44,63 \pm 0,29	4,81	1000
30	40	0,8	0,01	-101,69 \pm 0,74	11,96	-44,91 \pm 0,30	4,85	1000
40	40	0,8	0,02	-111,06 \pm 0,73	11,89	-46,24 \pm 0,30	4,87	1000
40	40	0,8	0,01	-109,59 \pm 0,73	11,84	-46,02 \pm 0,31	5,04	1000
30	40	0,95	0,02	-108,42 \pm 0,75	12,15	-44,91 \pm 0,31	5,14	1000
30	40	0,7	0,02	-101,26 \pm 0,69	11,27	-44,78 \pm 0,31	5,10	1000

Tabela 4 – Resultados da média de energia para 1000 testes com modelo Ising alterando outros parâmetros

Os resultados mostrados foram para o melhor cromossomo encontrado entre todas as gerações do GA. A [Tabela 5](#) mostra mais dois resultados para a implementação do elitismo mantendo os 3 melhores cromossomos para a próxima população. Os valores indicam que houve pequena melhora no algoritmo.

Para completar as avaliações, mantendo os parâmetros $P = 50$, $T = 40$, $p_c = 0,95$, $p_m = 0,02$ e com elitismo mantendo 3 cromossomos, foram salvos os valores de energia

Parâmetros				Resultados GA		Resultados RS		#
P	T	p_c	p_m	Média \pm IC	D.P.	Média \pm IC	D.P.	GA < RS
30	40	0,8	0,02	-110,66 \pm 0,66	10,73	-44,81 \pm 0,30	5,88	1000
30	40	0,95	0,02	-113,69 \pm 0,70	11,36	-44,75 \pm 0,30	4,99	1000

Tabela 5 – Resultados da média de energia para 1000 testes com modelo Ising e usando elitismo mantendo 3 cromossomos

da população dos 1000 testes para as gerações 0, 10, 20, 30 e 40. Depois foi calculado a média e variância dessas gerações para cada um dos testes e finalmente feita a média dos parâmetros entre os 1000 testes obtendo os valores mostrados na [Tabela 6](#)

Geração	0	10	20	30	40
Média	0,00	-47,49	-78,76	-96,46	-106,86
Variância	178,55	158,42	117,09	106,79	103,55

Tabela 6 – Resultados da média de energia para 1000 testes das gerações 0, 10, 20, 30 e 40 do modelo Ising e usando elitismo com 3 cromossomos

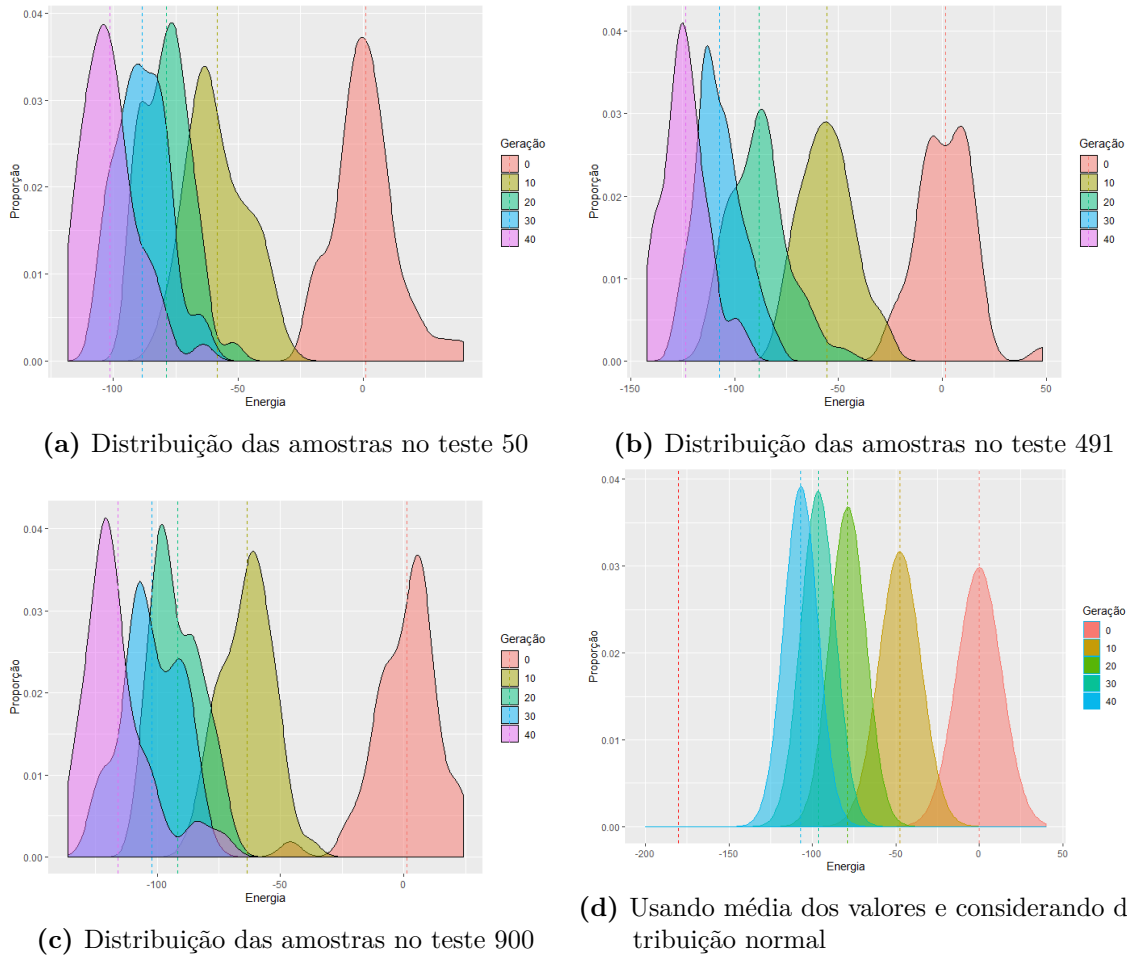


Figura 13 – Distribuições das populações para gerações 0, 10, 20 e 40, com $p_m = 0,02$

Diferente resultados vistos em [Prügel-Bennett e Shapiro \(1994\)](#), a variância reduz pouco, o que é esperado, pois no teste proposto só foi usado o operador de crossover.

Refazendo os testes com $p_m = 0$ obtemos a seguinte [Tabela 7](#). O operador de mutação mantém a variância na população pois ele garante o *exploration*, mantendo a diversidade da população.

Geração	0	10	20	30	40
Média	-0,02	-52,34	-85,29	-96,25	-98,72
Variância	180,96	120,07	36,62	8,01	1,54

Tabela 7 – Resultados da média de energia para 1000 testes com modelo Ising usando elitismo mantendo 3 cromossomos

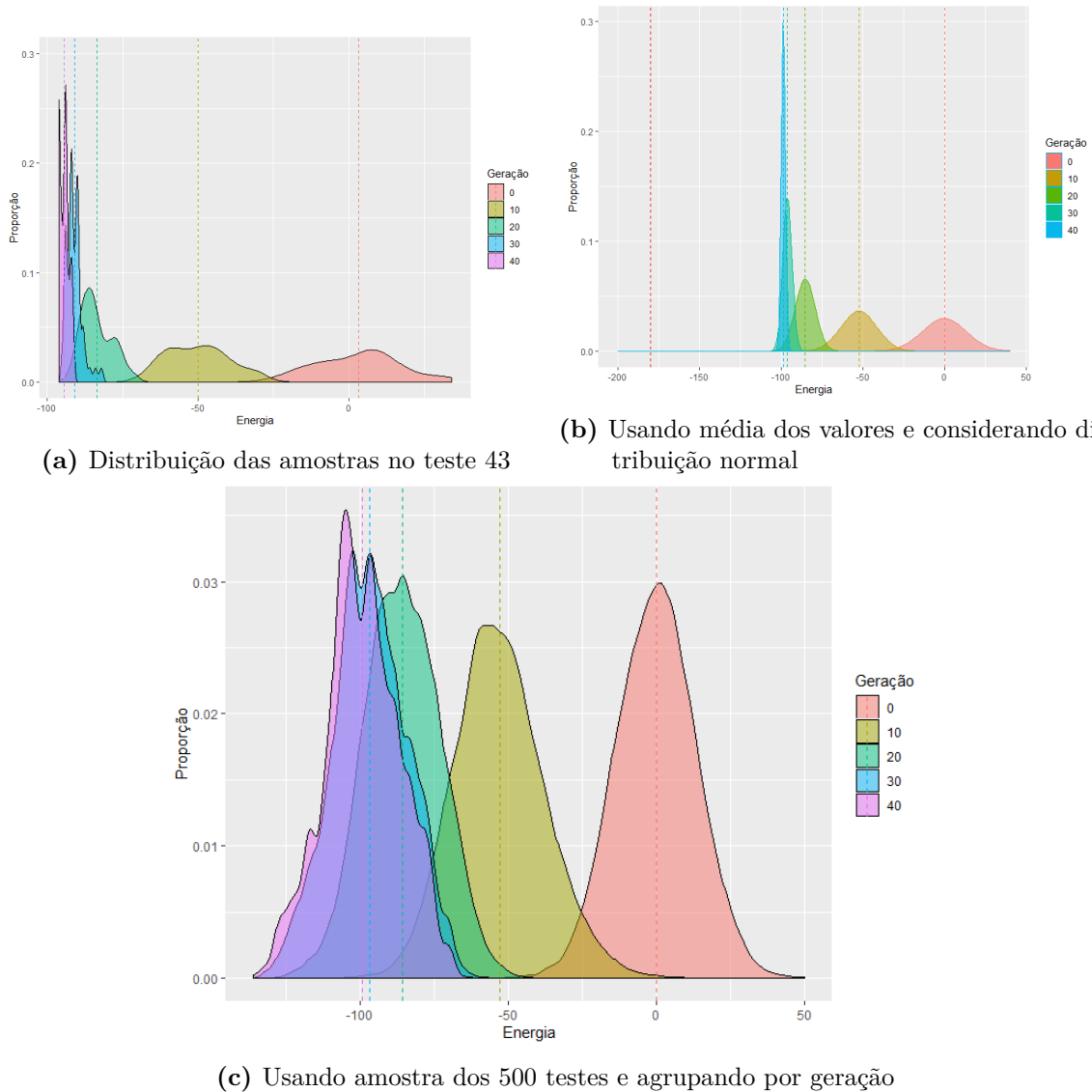


Figura 14 – Distribuições das populações para gerações 0, 10, 20 e 40, com $p_m = 0$,

Na [Figura 15](#) pode ser visto os resultados das médias e variâncias da população durante as gerações. Os gráficos representam a média das duas medidas de cada geração pelos 500 testes realizados. Quando a taxa de mutação é mantida em 0, percebe-se a rápida convergência do GA, chegando a um patamar mínimo de energia, e a variância

com tendência a zero. Ao colocar uma pequena taxa de mutação, o algoritmo converge mais lentamente, porém obtém resultados de média melhores, e a variância se mantém em um patamar. Isso é desejável no GA, pois indica que se manteve a diversidade genética. Além disso também é mostrado novamente o desempenho entre os operadores de crossover de um ponto e uniforme, e como esperado, o uniforme conseguiu médias melhores, mas a variância levou mais iterações para diminuir, justificado pelo fato desse operador obter maior diversidade genética no início, explorando melhor o espaço de soluções e garantindo melhores médias.

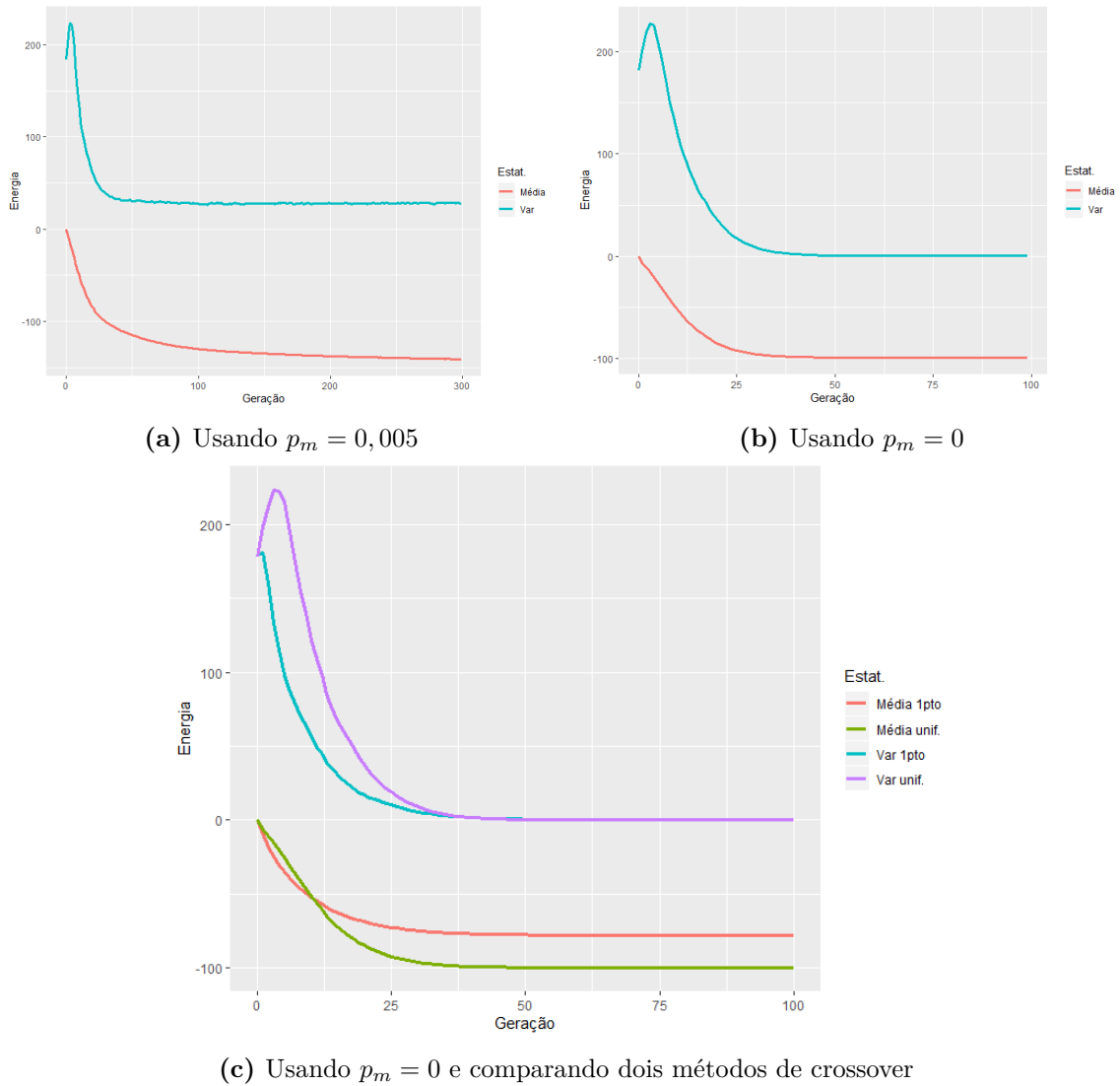


Figura 15 – Valores das médias e variâncias da população durante as gerações com $p_m = 0,005$. Os valores apresentados são da média de 500 testes

Por último na [Figura 16](#) é exposto a evolução do algoritmo durante as gerações com variações no parâmetro de probabilidade de mutação, mostrando o melhor, pior cromossomo da população além de sua avaliação média. As linhas horizontais tracejadas mostram o melhor, pior e média do método de busca aleatório, considerando sempre a amostra para esse método de tamanho $P \cdot T$.

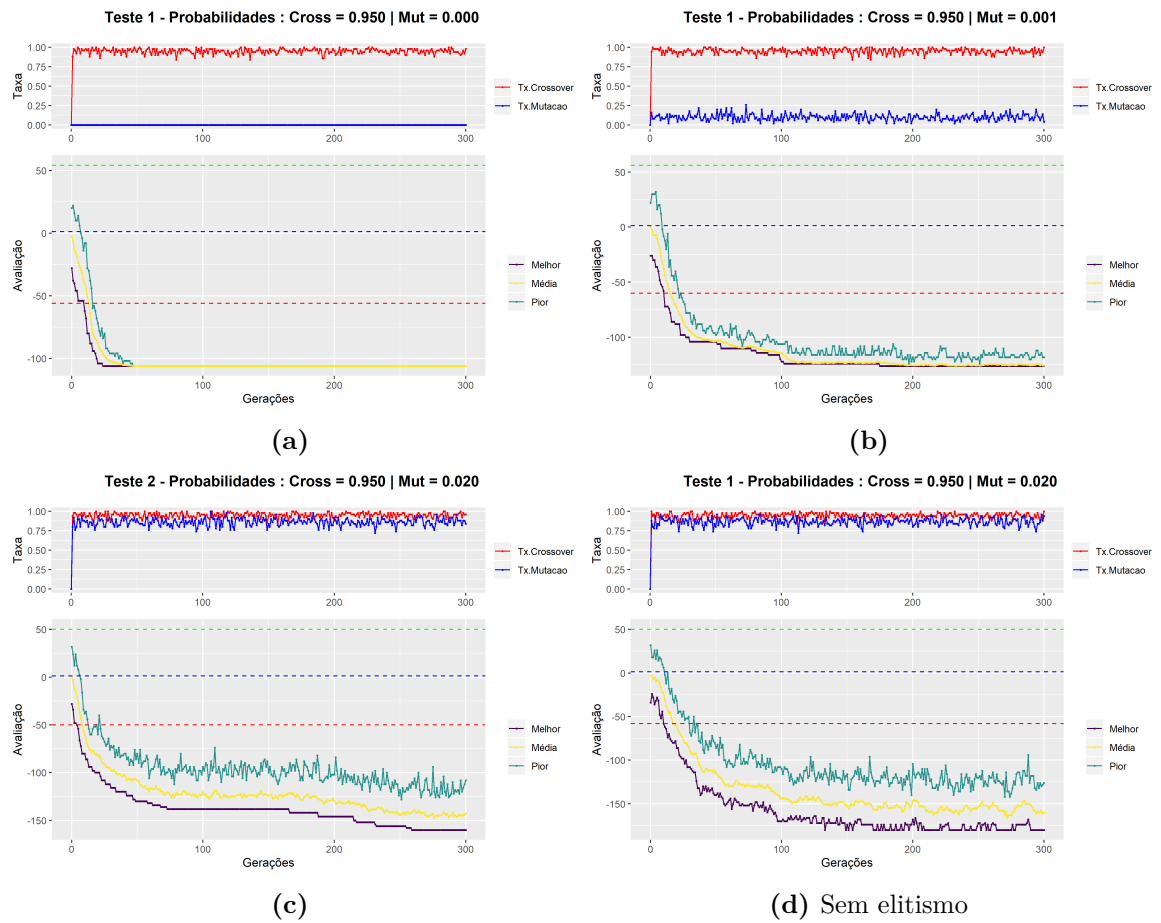


Figura 16 – Evolução do algoritmo genético durante gerações

Outros gráficos de resultados podem ser vistos na [Figura 17](#) do [Anexo A](#).

4 Conclusão

O algoritmo genético é uma outra ferramenta que dispomos pela busca de soluções ótimas. Como visto na literatura, podemos considerar que na verdade é um algoritmo com foco em busca de soluções satisfatórias do que exatamente da solução ótima. Uma das vantagens está na facilidade de implementação do algoritmo, porém possui parâmetros de configuração que ainda não se conhece formas de definir analiticamente os melhores valores, além de dependerem das condições do problema a ser resolvido. Com isso a maior parte dos trabalhos se baseiam em resultados empíricos para ajuste dos parâmetros. Foi visto que o GA também apresenta algumas dificuldades com certas condições, mas que podem ser amenizadas se propriamente codificado e parametrizado.

Nos testes realizados, o GA se mostrou bem mais eficiente que a busca aleatória, alcançando resultados mais expressivos. Portanto em problemas que por limitações inerentes seriam usados o algoritmo de busca aleatória, o algoritmo genético se apresenta como uma outra abordagem possível.

Ficou demonstrado que é muito importante as escolhas feitas para a arquitetura do GA e seus parâmetros podem definir o sucesso do algoritmo. No teste sobre o modelo de Ising, percebeu-se que a utilização de uma função de avaliação, que não era a mais apropriada, resultou em ineficiência do algoritmo obtendo resultados iguais e até inferiores que a busca aleatória, fato contornado usando outra função. Esse tipo de detalhe é exatamente o [LINDEN](#) menciona sobre quanto mais informação sobre o problema conseguir inserir no algoritmo, melhor será a resposta, e por isso não existe uma padrão do GA que seja ideal para todos os problemas.

Outro conceito que impôs diferença nos resultados foi o uso do elitismo, pois ao manter os melhores cromossomos de uma geração para outra, sustentou-se o melhor resultado até o fim do processo. Sem o elitismo era possível ver nos gráficos que o melhor cromossomo em gerações $t + 1$ ficavam com pior resultado que o o melhor da geração t .

No modelo de Ising, o algoritmo mostrou através da análise das médias e variâncias, que caminhava para uma convergência, e com boa probabilidade de se tiver gerações suficientes, encontrar o estado de menor energia

As aplicações para o GA são as mais variadas, podendo ser usado em problemas de busca de máximo ou mínimo de funções multivariadas e também com multiobjetivos, na busca de ordenações mais eficientes, como é o caso do problema do caixeiro viajante, e programação genética, onde o GA consegue criar pequenos algoritmos para solução de problemas.

Um das aplicações interessantes para o algoritmo é na busca dos coeficientes de uma rede neural. Para as redes usadas em processos de *machine learning* para classificação, pode ser usado de forma a determinar os coeficientes iniciais para o algoritmo de *backpropagation*, que por ser uma função com vários máximos locais, dependendo do ponto inicial utilizado pode ficar preso a um que não seja o máximo global. Normalmente o ponto inicial é definido aleatoriamente, e nesse sentido que poderia ser usado o GA, que teria uma população definida por cromossomos de valores reais definindo os coeficientes iniciais da rede. Para cada cromossomo poderia ser executado o *backpropagation* e verificado através dos resultados qual obteve melhor taxa de sucesso de classificação, e usado esse valor como função de avaliação.

Ainda com redes neurais, existem os casos que a rede define o comportamento de um sistema em resposta a sensores de entrada. O caso clássico de demonstração dessa aplicação é do treinar o computador para jogar determinado jogo eletrônico. A população fica definida como cromossomos de valores reais com os coeficientes da rede e o resultado da rede só pode ser visto após determinada simulação. A população do GA testada contra o resultado de pontuação, que será a função de avaliação do algoritmo, é evoluída então combinando os melhores resultados e aplicando mutação em alguns.

Outra característica interessante, que ainda pode ser explorada, é a possibilidade de combinar o GA com outro algoritmo de otimização, por exemplo direcionando a mutação para uma busca de máximo local da função de avaliação. Esse pode ser um diferencial combinando a qualidade do GA de encontrar soluções satisfatórias ou próximas a máximos locais e combinar com um algoritmo mais eficiente de maximização local, como o *hill climbing*.

O algoritmo genético ainda precisa ser muito analisado de forma a entender melhor seu mecanismo de funcionamento e dessa forma conseguir determinar os melhores parâmetros, e tem demonstrado certa popularidade com as novas técnicas de AI e *machine learning*.

Para trabalhos futuros podem ser seguido os testes com o modelo proposto de Ising, trabalhando com contornos e busca de soluções para acoplamentos do modelo, também podem ser explorados as técnicas de combinar com outros métodos. Ainda é necessário também avaliar o algoritmo com relação a seu desempenho computacional e outro passo importante seria o de executar o algoritmo usando paralelismo, pois a fase de avaliação e seleção poderia facilmente ser colocada em tarefas concorrentes, otimizando o tempo de resposta do algoritmo.

Referências

COELLO, C. C.; LAMONT, G. B.; VELDHUIZEN, D. A. van. **Evo-
lutionary Algorithms for Solving Multi-Objective Problems (Ge-
netic 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](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)>.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization,
and Machine Learning**. 1. ed. Reading, MA: Addison-Wesley Professi-
onal, 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](https://www.amazon.com/Genetic-Algorithms-Optimization-Machine-Learning/dp/0201157675?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0201157675)>.

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](https://www.ebook.de/de/product/2864942/john_h_holland_adaptation_in_natural_and_artificial_systems.html)>.

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](https://www.ebook.de/de/product/25481443/lee_jacobson_burak_kanber_genetic_algorithms_in_java_basics.html)>.

JUNIOR, J. E. de L. **Modelo de Ising Aplicado ao estudo da criminalidade**. 56 f.
Dissertação (Mestrado em Ciências) — Escola de Artes, Ciências e Humanidades da
Universidade de São Paulo, São Paulo, 2014.

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](https://www.amazon.com/Concepts-Genetics-William-S-Klug/dp/0321724127?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321724127)>.

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](https://www.ebook.de/de/product/3753808/sam_kwong_kim_fung_man_kit_sang_tang_genetic_algorithms.html)>.

LINDEN, R. **Algoritmos Genéticos**. 2. ed. Rio de Janeiro: Brasport, 2008. ISBN
978-8574523736.

MCCOY, B. M.; WU, T. T. **The Two-Dimensional Ising model**. Cambridge, Mass:
Harvard University Press, 1973. [by] Barry M. McCoy and Tai Tsun Wu., Includes
bibliographical references. ISBN 0674914406.

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>>.

NIX, A. E.; VOSE, M. D. Modeling genetic algorithms with markov chains. **Annals of Mathematics and Artificial Intelligence**, Springer Science and Business Media LLC, v. 5, n. 1, p. 79–88, mar 1992.

PRÜGEL-BENNETT, A.; SHAPIRO, J. L. Analysis of genetic algorithms using statistical mechanics. **Physical Review Letters**, American Physical Society (APS), v. 72, n. 9, p. 1305–1309, feb 1994.

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>>.

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>.

VOSE, M. D.; LIEPINS, G. E. Punctuated equilibria in genetic search. **Complex Systems**, v. 5, 1991.

ANEXO A – Resultados

Cromossomo	Avaliação	$\mathcal{H}(\sigma)$	Taxa de seleção	Proporção estimada
287	222	42	0,067	0,040
277	206	26	0,050	0,037
280	204	24	0,033	0,037
282	204	24	0,017	0,037
285	200	20	0,000	0,036
276	194	14	0,000	0,035
291	194	14	0,083	0,035
286	192	12	0,017	0,034
293	192	12	0,033	0,034
283	190	10	0,033	0,034
288	190	10	0,017	0,034
289	190	10	0,000	0,034
299	188	8	0,017	0,034
273	186	6	0,067	0,033
300	186	6	0,017	0,033
292	184	4	0,033	0,033
274	182	2	0,000	0,033
297	182	2	0,067	0,033
271	180	0	0,033	0,032
281	180	0	0,033	0,032
295	180	0	0,050	0,032
290	178	-2	0,050	0,032
294	178	-2	0,033	0,032
278	176	-4	0,017	0,032
284	176	-4	0,067	0,032
279	174	-6	0,033	0,031
275	172	-8	0,067	0,031
298	172	-8	0,033	0,031
272	164	-16	0,000	0,029
296	160	-20	0,033	0,029

Tabela 8 – Resumo da geração 9 com média de avaliação 185,86 mostrando a taxa de amostragem da seleção e o que era esperado para a proporção usando a função de avaliação da [Equação 3.1](#)

Cromossomo	Avaliação	$\mathcal{H}(\sigma)$	Taxa de seleção	Proporção estimada
178	6,69	-38	0,100	0,063
160	6,05	-36	0,100	0,057
159	5,47	-34	0,033	0,052
166	5,47	-34	0,067	0,052
162	4,95	-32	0,067	0,047
171	4,95	-32	0,000	0,047
180	4,95	-32	0,033	0,047
167	4,48	-30	0,050	0,042
175	4,48	-30	0,017	0,042
161	4,06	-28	0,050	0,038
163	4,06	-28	0,033	0,038
169	4,06	-28	0,017	0,038
151	3,67	-26	0,050	0,035
155	3,67	-26	0,033	0,035
170	3,32	-24	0,017	0,031
176	3,32	-24	0,050	0,031
153	3,00	-22	0,067	0,028
157	3,00	-22	0,033	0,028
174	3,00	-22	0,050	0,028
152	2,72	-20	0,000	0,026
164	2,72	-20	0,050	0,026
165	2,72	-20	0,000	0,026
158	2,46	-18	0,017	0,023
154	2,23	-16	0,033	0,021
168	2,01	-14	0,000	0,019
173	2,01	-14	0,000	0,019
156	1,82	-12	0,033	0,017
179	1,82	-12	0,000	0,017
172	1,22	-4	0,000	0,012
177	1,11	-2	0,000	0,010

Tabela 9 – Resumo da geração 5 com média de avaliação 3.516 mostrando a taxa de amostragem da seleção e o que era esperado para a proporção usando a função de avaliação da [Equação 3.2](#)

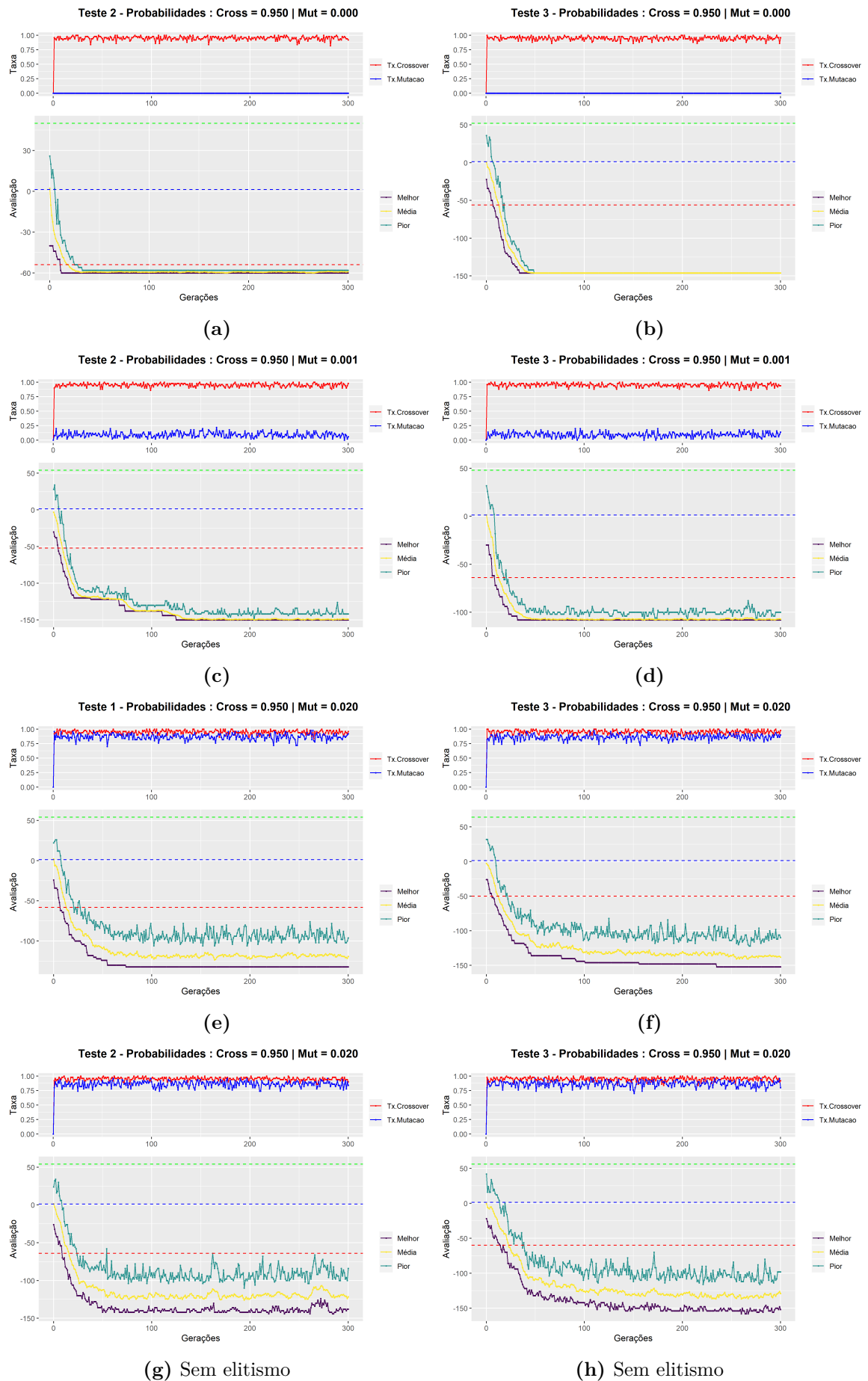


Figura 17 – Evolução do algoritmo genético durante gerações