CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

CAMPUS TIMÓTEO

Paula Peçanha Gonçalves

Comparação entre dois modelos de algoritmos genéticos utilizando diferentes funções de avaliação de minimização

Timóteo

2016

Paula Peçanha Gonçalves

Comparação entre dois modelos de algoritmos genéticos utilizando diferentes funções de avaliação de minimização

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Centro Federal de Educação Tecnológica de Minas Gerais

Campus Timóteo

Graduação em Engenharia de Computação

Orientador: Prof. Douglas Nunes de Oliveira

Timóteo

2016

Paula Peçanha Gonçalves

Comparação entre dois modelos de algoritmos genéticos utilizando diferentes funções de avaliação de minimização

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Timóteo, 05 de julho de 2016:

Prof. Douglas Nunes de Oliveira
Orientador

Prof. Adilson Mendes Ricardo
Professor Convidado

Prof. André Rodrigues da Cruz
Professor Convidado

Timóteo
2016

Aos meus pais,
pelo encorajamento e exemplo de honestidade e dedicação.

Agradecimentos

Agradeço aos meus pais, Paulo e Cláudia, pela paciência, apoio, encorajamento, cuidado, amor incondicional e confiança.

Ao meu irmão, José Mário, pelas palavras de estímulo e por ser meu exemplo.

Ao meu orientador, professor Douglas Nunes, pela disposição em me orientar, por ter partilhado seu conhecimento, por ouvir minhas opiniões e por colaborar sempre com boa vontade.

As minhas avós, Clarice e Sércia, pelas orações e pelo carinho.

A minha madrinha, Márcia, que sempre esteve ao meu lado, se preocupando, me auxiliando e me animando.

Ao meu padrinho, Ivaldo, por ser sempre prestativo.

Ao meu tio Aloísio, pelas caronas e proveitosas conversas.

Ao meu tio Mário, por desde cedo incentivar meus estudos.

A todos os meus familiares pela torcida, principalmente aos tios e tias, primos e primas, que me aconselharam a continuar neste caminho.

A todos os meus amigos, principalmente a Fernanda e ao Mike, que me ajudaram neste trabalho e estão sempre dispostos a contribuir.

A todos os professores e funcionários do CEFET-MG campus Timóteo, que tiveram participação na minha trajetória dentro da instituição.

Ao professor André Rodrigues da Cruz, que se disponibilizou e me auxiliou na correção deste trabalho.

E por último, mas não menos importante, agradeço a Deus pela força e perseverança proporcionada para que eu conseguisse chegar até aqui.

"Se os GAs são tão inteligentes,
por que eles não são ricos?"
(GOLDBERG, 1989, p.89)

Resumo

Das muitas técnicas de busca existentes, entre as mais populares estão os algoritmos genéticos (GAs). Os algoritmos genéticos são técnicas probabilísticas que exploram possíveis soluções para o problema em questão. Eles são intensamente utilizados em diversos projetos que desejam otimizar uma situação, seja na área acadêmica ou industrial. O propósito desta pesquisa é analisar dois modelos de GA distintos e comparar suas respostas pelo teste de Wilcoxon pareado. Neste trabalho, os modelos são descritos e comparados utilizando as mesmas funções objetivo e populações iniciais. O primeiro modelo, o SGA+E, é uma mesclagem entre o *simple genetic algorithm* (SGA), criado por David E. Goldberg, e o *elitist model R2*, elaborado por Kenneth Alan De Jong. O segundo modelo, o *full genetic algorithm* (FGA), foi desenvolvido pelo orientador desta pesquisa, o professor Douglas Nunes de Oliveira. As funções de avaliação empregadas são funções de otimização que almejam minimizar os resultados. O programa implementado, permite que a cada execução os dois modelos utilizem as seis funções objetivo, uma por vez. Para cada função, cria-se aleatoriamente uma população inicial. A cada modelo uma população final é retornada e os melhores indivíduos comparados. Apesar de ambos os modelos disporem dos mesmos operadores genéticos, devido a disparidade em suas estruturas, eles demonstram diferentes resultados finais. Nos dois testes feitos, o primeiro com uma população de 100 indivíduos e o segundo com 200, nenhum dos modelos apresentou melhores populações finais para todos os casos pelo teste de Wilcoxon pareado. O modelo SGA+E produziu melhores resultados para as funções de avaliação que têm a função cosseno em sua fórmula; já o modelo FGA gerou os indivíduos mais adequados para as funções de avaliação que não possuem demasiados ótimos locais.

Palavras-chave: algoritmo genético. modelo de GA. otimização.

Abstract

Regarding the many existing search techniques, among the most popular, we can point out the genetic algorithms (GAs). Genetic algorithms are probabilistic techniques that explore possible solutions of a problem. They are extensively used in various projects that want to optimize a situation, whether in the academic or industrial area. The purpose of this research is to analyze two different models of GA and compare their results using the Wilcoxon signed rank test for paired observations. In this work, two models are described and compared using the same objective functions and initial populations. The first model, SGA+E, is a blend between the simple genetic algorithm (SGA) created by David E. Goldberg, and the elitist model R2 elaborated by Kenneth Alan De Jong. The second model, the full genetic algorithm (FGA) was developed by the mentor of this research, professor Douglas Nunes de Oliveira. The evaluation functions employed are optimization functions that aim to minimize the results. The implemented program allows that at each run, the two models use the six objective functions, one at a time. For each function, the program creates an initial population, which is used by both models. Each model returns one final population in which the best individuals are compared. Although both models dispose of these genetic operators, due to the disparity in their structures, they show different results. In the two tests performed, the first with a population of 100 individuals and second with 200, none of the models showed better final populations for all the cases using the Wilcoxon signed rank test for paired observations. The SGA+E model produced better results for the evaluation functions that have the cosine function in its formula; once that the FGA model generated the most suitable individuals for evaluation functions that do not have too many local optimums.

Keywords: genetic algorithms. GA model. optimization.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

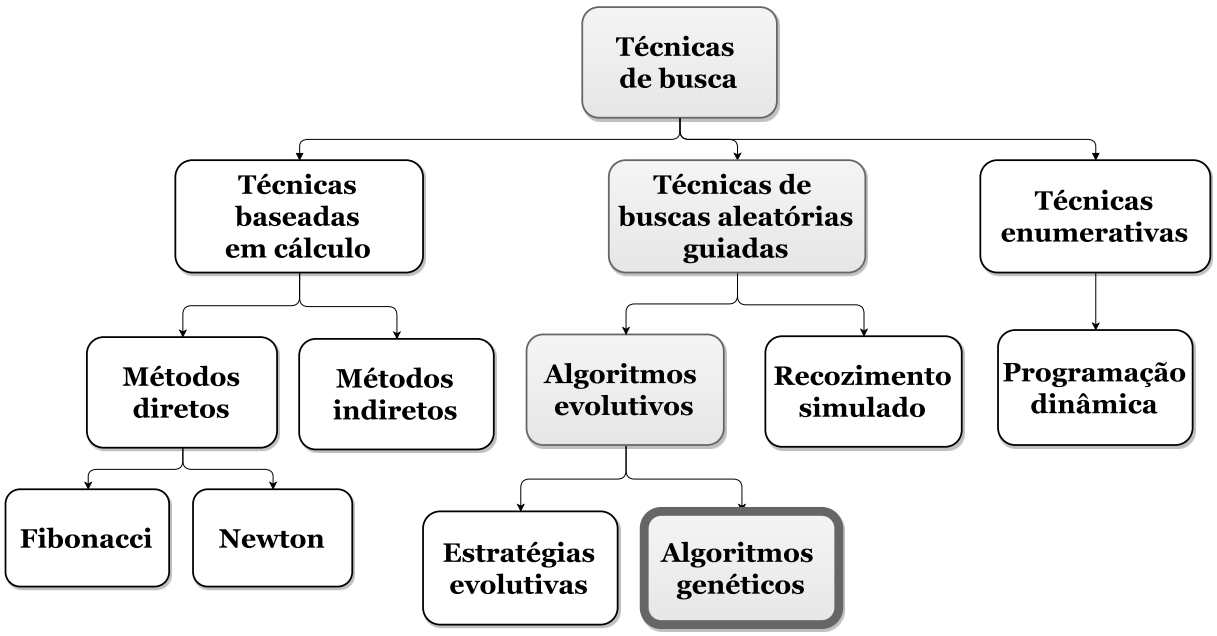
Há um dito popular afirmando que, “o que a natureza cria o homem copia”. Na realidade, este enunciado condiz com muitas tecnologias já desenvolvidas. A imitação da natureza feita pelo ser humano denomina-se biomimetismo e um conhecido exemplo deste é o velcro, tecnologia baseada nas sementes de uma planta que grudam na roupa.

Não apenas os seres, como também comportamentos, podem ser observados na natureza e reproduzidos de forma artificial (TANOMARU, 1995). Foi isto, o que John H. Holland fez em 1975, quando, a partir do conceito de seleção natural, criado por Charles Darwin, desenvolveu o método de busca conhecido como algoritmo genético (GA - *genetic algorithm*) (HOLLAND, 1975-1992; MITCHELL; FORREST; HOLLAND, 1992).

Segundo Goldberg, “os algoritmos genéticos são algoritmos de busca baseados nos mecanismos de seleção natural e na genética da natureza” (GOLDBERG, 1989, p. 1).

Os algoritmos genéticos são considerados algoritmos evolutivos, assim como as estratégias evolutivas (ESs - *evolutionary strategies*) (DIANATI; SONG; TREIBER, 2002). Embora estejam representados de forma separada na Figura 1, alguns artifícios de uma das técnicas podem ser empregados na outra (POZO et al., 2005).

Figura 1 – Árvore representativa das técnicas de busca



Fonte: Adaptado de (DIANATI; SONG; TREIBER, 2002, p. 3)

Os GAs foram elaborados para resolver problemas complexos, os quais estão subordinados a vários fatores. Esses fatores são dependentes entre si e a mudança no valor

de um deles pode alterar o resultado do todo. Os algoritmos genéticos se tornaram tão importantes que existe uma vasta diversidade de áreas em que são aplicados.

Nos Estados Unidos, para diminuir acidentes causados pelo transporte e armazenamento de materiais perigosos, Ardjmand et al. (2016) utilizaram um algoritmo genético para minimizar o custo total e os riscos que os materiais apresentam. O algoritmo foi elaborado para indicar lugares em que os materiais poderiam ser armazenados e quais as melhores rotas para transporte desses. Eles obtiveram boas soluções em pouco tempo, indicando a eficiência da técnica.

Ainda considerando problemas ambientais, quando incidentes ocorrem e pequenos rios são poluídos, há emergência na identificação da fonte poluente. Zhang e Xin (2016) propuseram um modelo de algoritmo genético capaz de localizar a origem da poluição, tendo esta apenas uma ou várias origens. Para localização de múltiplas fontes poluentes, alguns resultados mostraram erros de precisão. Entretanto, com informações de experiências anteriores, o algoritmo conseguiu minimizar os erros e obteve respostas satisfatórias (ZHANG; XIN, 2016).

Outro exemplo de aplicação atual dos algoritmos genéticos, agora na área computacional, é o GA criado por Jônatas L. de Paiva, para diminuir ruídos em imagens digitais. Paiva (2016) propôs uma junção entre um GA e uma técnica de redução de ruídos para imagens. A essa associação dá-se o nome de algoritmo genético híbrido. Dessa forma, foi possível recuperar imagens com resultados melhores que de outros processos com a mesma finalidade.

1.1 Justificativa

Diferentes implementações de modelos de algoritmos genéticos são relatados na literatura (VAVAK; FOGARTY, 1996).

Nesta proposta, apresenta-se e compara-se dois modelos, o primeiro modelo é o *simple genetic algorithm* (SGA), criado por Goldberg (1989), somado ao operador evolucionário elitismo, utilizado pela primeira vez por DeJong (1975); e o segundo modelo foi sugerido pelo professor orientador deste trabalho.

A escolha por estudar e contrapor diferentes modelos de algoritmos genéticos, se justifica na importância que esses têm na indústria e no meio acadêmico.

Os GAs são capazes de solucionar diversos tipos de problemas de otimização em diferentes áreas. Por exemplo, Goldberg (1989, p.125) utilizou os algoritmos genéticos para otimizar compressores de um gasoduto de gás natural, onde o aumento ou diminuição da pressão afetam o consumo de energia (BOOKER; GOLDBERG; HOLLAND, 1989).

Outro exemplo de aplicação é demonstrado por PEREIRA et al. (2016), que

desenvolveram um modelo de GA para analisar a influência do processamento paralelo na otimização de um reator nuclear.

Vários outros exemplos de utilização de algoritmos genéticos podem ser encontrados. Por último, pode-se citar a dissertação de Oliveira (2016) que também propôs um novo modelo de GA. Este porém, com o intuito de minimizar os custos e a perda de energia, ao adquirir novos equipamentos para uma rede coletora de média tensão de parques eólicos.

1.2 Objetivos

Assim como no estudo de Vavak e Fogarty (1996), neste trabalho também é feita a comparação entre dois modelos de algoritmos genéticos. Esta pesquisa tem por finalidade verificar se a estrutura de um modelo de algoritmo genético é capaz de melhorar a solução final para as funções de avaliação utilizadas. Para isso, será analisado qual deles apresenta uma resposta de melhor qualidade para a mesma população inicial, função de avaliação e demais parâmetros por meio do teste de Wilcoxon pareado.

As funções de avaliação empregadas, são funções de otimização que buscam minimizar o valor de saída. Essas funções de avaliação, são usadas para classificar o indivíduo de acordo com suas características físicas e mentais (MITCHELL, 1998). A cada execução do programa, a mesma população inicial será utilizada pela função objetivo escolhida para ambos os modelos. Para cada população inicial, serão geradas duas populações finais, cada uma por meio de um dos modelos propostos.

1.3 Organização do trabalho

Este trabalho apresenta um estudo comparativo entre dois modelos de algoritmos genéticos, e está estruturado em cinco capítulos.

No segundo capítulo estão descritos os assuntos relevantes para melhor compreensão da proposta. Nele são apresentados os algoritmos genéticos, o que são, alguns métodos e operadores genéticos que os compõe; além de uma introdução sobre o teste de Wilcoxon pareado.

No capítulo 3 está explicada a implementação da proposta. Nessa seção são discutidos: o programa, os parâmetros utilizados e as características específicas de cada modelo.

Os resultados obtidos estão contidos no capítulo 4. Um resumo das respostas de todas as execuções, para cada função e modelo utilizado é apresentado e analisado pelo teste de Wilcoxon para observações pareadas. Ainda neste capítulo é demonstrado um teste adicional com uma população com o dobro de indivíduos do que havia no primeiro

teste.

O capítulo 5 encerra este trabalho com a conclusão da proposta e algumas sugestões de projetos a serem desenvolvidos a partir deste.

Para melhor compreensão das análises feitas, adicionalmente são apresentados os resultados na seção de apêndices. Para cada teste realizado, são mostradas seis tabelas com os melhores indivíduos encontrados pelos modelos a cada iteração. O número de tabelas corresponde ao número de funções utilizadas.

2 Fundamentação teórica

Os assuntos abordados neste capítulo são necessários para a elaboração da proposta e melhor entendimento de sua relevância. São eles: o surgimento dos algoritmos genéticos; seus conceitos básicos; como o indivíduo é afetado pelo meio em que habita; e o que são os operadores genéticos. Adicionalmente, tem-se uma breve descrição sobre o teste de Wilcoxon para observações pareadas.

2.1 Algoritmos genéticos

Ao se pensar nas teorias sobre a evolução das espécies, o nome mais popular a ser lembrado é provavelmente o do cientista Charles Darwin. Em uma de suas mais famosas publicações, o livro *The Origin of Species* (A origem das espécies), Darwin introduz o conceito de seleção natural.

A seleção natural afirma que os indivíduos que sobrevivem ao processo natural de evolução, são aqueles que possuem características que os auxiliam a melhor se adaptar ao ambiente (FRANCIS, 2007).

A adaptação de uma espécie está diretamente relacionada ao ambiente em que ela vive. Por exemplo, um organismo capaz de se camuflar junto ao espaço físico no qual se encontra, tem menos chances de ser capturado por seu predador, do que um organismo sem essa habilidade. Sendo assim, aquele tem mais chances de sobreviver e se reproduzir.

Com o ideal de aplicar os princípios da seleção natural no universo artificial para resolução de problemas matemáticos, John H. Holland, seus colegas e alunos, desenvolveram os algoritmos genéticos (GAs) (MITCHELL, 1998).

Em 1975, Holland publicou a primeira edição do livro *Adaptation in natural and artificial systems*, que relacionava os conceitos básicos da biologia com a forma computacional associada. Ademais apresentou em quais áreas os algoritmos genéticos poderiam ser utilizados, e as formas de identificar algumas propriedades importantes dos indivíduos (HOLLAND, 1975-1992).

Holland (1975-1992), almejava criar um programa no qual o computador buscaria as melhores soluções para um problema de otimização. Ele queria que o algoritmo explorasse as possibilidades de forma a atender uma demanda, e como na natureza, adaptar-se ao ambiente.

Em seu livro, Holland (1975-1992) menciona *the adaptive plan* (o plano de adaptação), no qual o indivíduo mais adequado ao ambiente teria mais chances de sobreviver. Para

qualificar um indivíduo como adaptado ou não, ele propôs uma medida de performance do indivíduo, o que é chamado de *fitness*.

De acordo com Goldberg (1989), os GAs são algoritmos de busca que objetivam melhorar a eficiência e eficácia da busca, de forma a aumentar sua performance e reduzir seu custo. Algumas vantagens dos algoritmos genéticos sobre outros métodos de busca são (GOLDBERG, 1989, p. 7):

- a busca de solução utilizando vários pontos e não apenas um;
- a implementação das funções objetivo, o que implica na não necessidade de informações adicionais;
- e o uso de probabilidades, o qual permite que com a mesma população inicial, possa-se obter diferentes resultados.

A principal característica dos GAs, e o motivo por sua ampla utilização, é o grau de generalização que o GA possui, o que lhe permite solucionar os mais diversos tipos de problemas de otimização (LINDEN, 2012).

Em geral, os algoritmos genéticos são utilizados para resolver questões com diversas variáveis interdependentes, onde a alteração de qualquer uma dessas pode afetar o resultado como um todo (WHITLEY, 1994; COPPIN, 2010). Na biologia, o conceito para explicar a dependência dos genes e a influência que estes sofrem do material genético de seus ancestrais é chamada de epistasia (HOLLAND, 1975-1992; HOLLAND, 1992).

2.1.1 Conceitos fundamentais

No algoritmo genético as soluções candidatas para o problema são chamadas de indivíduos. À estrutura deste dá-se o nome de genótipo, o qual é formada pelos genes. Cada indivíduo possui um número finito de genes (BOOKER; GOLDBERG; HOLLAND, 1989). Geralmente, em um único GA, todos os indivíduos possuem o mesmo número de genes. A posição que este ocupa é denominada *locus* e o valor dentro do gene é denominado alelo.

Segundo Mitchell (1998), os alelos são os responsáveis pelo fenótipo do indivíduo, suas características físicas e mentais. O conjunto dos indivíduos formam uma população. A implementação de um algoritmo genético, geralmente se inicia com uma população de indivíduos gerados de forma aleatória (WHITLEY, 1994).

Holland (1975-1992), DeJong (1975), Goldberg (1989) e outros estudiosos dos algoritmos genéticos, representavam os indivíduos como sequências de números binários. Deste modo, os alelos somente poderiam ser 0 ou 1.

Rechenberg, por outro lado criou, e posteriormente Schwefel desenvolveu, as estratégias evolutivas (ESs), que codificam os indivíduos com números reais (BACK; HOFFMEISTER; SCHWEFEL, 1991).

Os conceitos e implementações dos algoritmos genéticos, e das estratégias evolutivas, se entrelaçaram ao passar dos anos, embora possam ser desenvolvidos de forma independente (MITCHELL, 1998).

Como mencionado anteriormente, o *fitness* é utilizado para classificar o quão adaptado o indivíduo está ao ambiente (FILHO, 2005; KOZENY, 2015). Este número é gerado por meio de uma função de avaliação, também conhecida como função objetivo.

Os indivíduos que forem avaliados como uma melhor solução, melhor *fitness*, tem mais chances de se reproduzir e passar seu material genético para os descendentes (WHITLEY, 1994).

Conforme Filho (2005, p. 8) menciona em sua dissertação: "a seleção natural atua apenas no espaço fenotípico". Ou seja, o resultado da função objetivo é o único responsável pela competição entre os indivíduos.

Na biologia, a evolução das espécies é estudada comparando-se a geração atual com as gerações anteriores. Da mesma forma, no processo artificial, as soluções candidatas passam por várias gerações e em cada uma dessas sofrem os efeitos do ambiente e dos operadores genéticos.

2.1.2 Impactos do ambiente

Os estudos de Darwin sobre seleção natural, evidenciam que o meio em que os indivíduos habitam tem grande influência em sua sobrevivência. Exemplos disso, são a escassez e disputa por alimentos, nos quais os mais fortes tem mais chances de permanecerem vivos; bem como as artimanhas de camuflagem para escapar dos predadores, em que os mais ágeis tem maior probabilidade de sobreviverem (FRANCIS, 2007).

Duas influências do ambiente são discutidas a seguir: a seleção e o elitismo. Serão mostradas algumas formas conhecidas de como esses métodos são implementados nos algoritmos genéticos.

2.1.2.1 Seleção

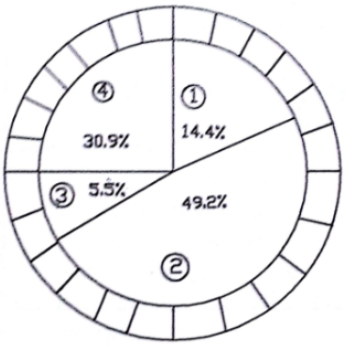
O método de seleção é o mecanismo utilizado para eleger os melhores indivíduos a progenitores da próxima geração. Ele atua com base no resultado da avaliação dos indivíduos. Algumas das principais técnicas de seleção são o *roulette wheel*, seleção por torneio e a seleção baseada em ranking.

O *roullete wheel* permite que todos os indivíduos tenham uma probabilidade de

serem selecionados. Entretanto, essa probabilidade é proporcional ao *fitness* do mesmo. Quanto melhor o *fitness*, maior a probabilidade que o indivíduo tem de ser escolhido para a reprodução (GOLDBERG, 1989).

Nesta técnica, o valor de avaliação de todos os indivíduos é somado em uma variável, o que equivale ao espaço total da roleta. Cada indivíduo recebe então, o espaço da roleta correspondente ao valor de seu *fitness* dividido pelo *fitness* total da população (Figura 2). Desta forma, existe também a possibilidade do pior indivíduo ser escolhido e o melhor ser descartado (FILHO, 2005).

Figura 2 – Exemplo de roleta em uma população de quatro indivíduos



Fonte: (GOLDBERG, 1989, p. 11)

Já na seleção por torneio, o *fitness* de cada indivíduo é comparado com outros indivíduos da mesma população, escolhidos aleatoriamente. A cada comparação, o que tiver o melhor *fitness* ganha a disputa (GABRIEL; DELBEM, 2008). Ao final das competições, tem-se, em ordem de melhor *fitness*, os indivíduos selecionados para reprodução.

A seleção baseada em ranking foi porposta por Baker (1985). Nesta técnica, os indivíduos são colocados em um ranking de acordo com seu *fitness*, em que o melhor indivíduo ocupa a posição de ápice e o pior fica em último lugar.

Sendo assim, o método utiliza as posições dos competidores para atribuir-lhes a probabilidade de serem selecionados. Este tipo de seleção foi criada com o propósito de evitar a convergência prematura do algoritmo (MITCHELL, 1998).

2.1.2.2 Elitismo

O elitismo é também um processo de seleção. Foi desenvolvido por DeJong (1975), e sua finalidade é garantir que o indivíduo com o melhor *fitness* passará para a próxima geração.

O aluno de Holland, percebeu que, com o passar das gerações, algumas soluções ótimas poderiam ser descartadas e seu material genético perdido. Para evitar essa ocorrência, De Jong propôs a seguinte regra:

> Seja a*(t) o melhor indivíduo gerado até o tempo t. Se, depois de gerar A(t+1) na forma habitual, a*(t) não estiver presente na geração A(t+1), então, inclua a*(t) em A(t+1) como o (N+1)º membro. (DEJONG, 1975, p. 102)

Em que A(t) é a geração atual e A(t+1) a próxima geração.

2.1.3 Operadores genéticos

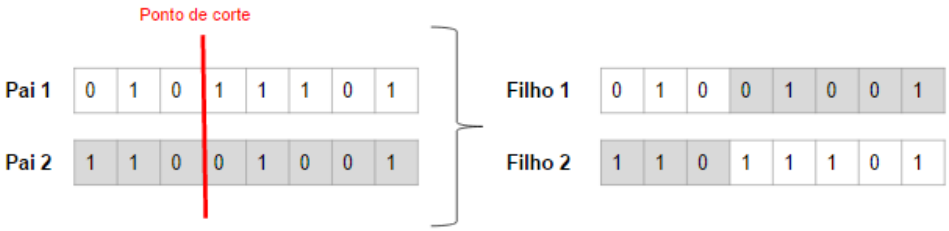
Os operadores genéticos são recursos utilizados pela natureza para aumentar a diversidade dos indivíduos, passar adiante características essenciais para a sobrevivência e gerar novos atributos.

2.1.3.1 Recombinação

A recombinação ou cruzamento, é o processo em que os indivíduos escolhidos pela seleção são utilizados para gerar os filhos. Alguns dos métodos mais conhecidos são o *crossover* de ponto único e *crossover* de dois ou n pontos, o *crossover* uniforme e o *crossover* aritmético.

O *crossover* de ponto único escolhe aleatoriamente um ponto entre dois *locus* dos indivíduos. Todos os genes a esquerda do pai 1 e os genes a direita do pai 2, formam o primeiro filho. O restante, a esquerda do pai 2 e direita do pai 1, formam o segundo filho, como mostra a Figura 3 (FILHO, 2005).

Figura 3 – *Crossover* de ponto único



Fonte: Adaptado de (FILHO, 2005, p. 11 e 12)

No *crossover* de dois pontos, existem dois pontos de corte, e o material genético é distribuído entre os filhos de acordo com a Figura 4 (ZUBEN, 2000). De forma similar, o *crossover* de n pontos escolhe n pontos de corte e separa o material genético entre os descendentes.

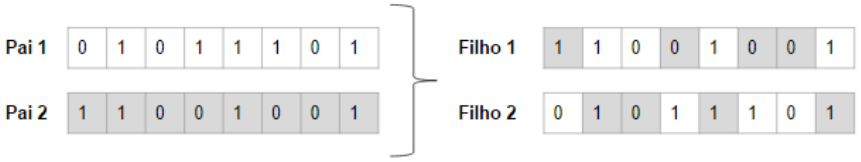
No *crossover* uniforme existe uma probabilidade para que o filho receba o gene do pai 1 ou pai 2. A cada *locus* é sorteado a origem do alelo que vai preencher o gene do filho, como ilustra a Figura 5 (ZUBEN, 2000).

O *crossover* aritmético (Figura 6), já não pode ser utilizado para codificações binárias. A cada cruzamento, uma porcentagem alfa (α) é sorteada para a recombinação.

Figura 4 – *Crossover* de dois pontos

Fonte: Baseado em (MITCHELL, 1998, p. 121)

Figura 5 – *Crossover* uniforme



Fonte: Adaptado de (FILHO, 2005, p. 12)

Esta porcentagem é utilizada para saber o quanto será herdado do genótipo dos pais seguindo o seguinte critério:

$$filho_a = \alpha pai_1 + (1 - \alpha) pai_2 \tag{2.1}$$

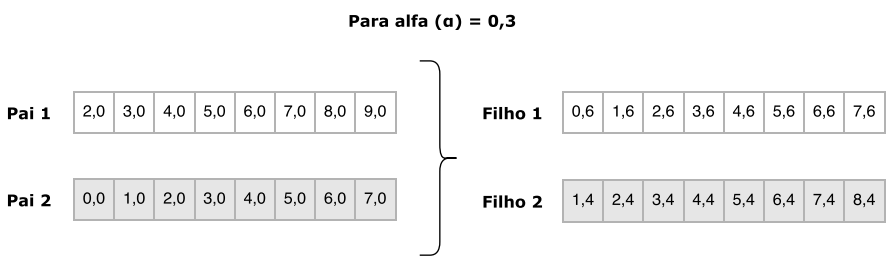
$$filho_b = (1 - \alpha) pai_1 + \alpha pai_2 \tag{2.2}$$

De acordo com Filho (2005):

> Esse operador é particularmente apropriado para problemas de otimização numérica com restrições, onde a região factível é convexa. Se x_1 e x_2 pertencem à região factível, combinações convexas de x_1 e x_2 também serão factíveis, garantindo que o *crossover* não vai gerar indivíduos inválidos para o problema em questão. (FILHO, 2005, p. 13)

x_1 e x_2 representam os indivíduos que serão recombinados para formar os filhos, ou seja, x_1 e x_2 são os pais.

Figura 6 – *Crossover* aritmético



Fonte: a autora

2.1.3.2 Mutação

O operador da mutação altera de forma aleatória o valor de um ou mais alelos de alguns indivíduos. O seu principal papel, é não permitir que um alelo seja excluído ao passar das gerações (WHITLEY, 1994; COBB; GREFENSTETTE, 1993). Ou seja, que características boas possam ser reinseridas nos genes da população. Três tipos populares de implementação da mutação são: a mutação pontual, a mutação uniforme e a mutação gaussiana.

A mutação pontual é utilizada para indivíduos binários. Uma probabilidade de mutação (p_m) é estabelecida previamente, a qual indicará a possibilidade de o alelo sofrer ou não a mutação. Ao ser escolhido como mutante, uma probabilidade é sorteada a cada gene do indivíduo e comparada a probabilidade de mutação. Se o número sorteado for compatível (menor ou igual) a p_m, o valor do alelo será mutado de 0 para 1 ou vice-versa (GOLDBERG, 1989).

Diferentemente da mutação pontual, a mutação uniforme é utilizada para codificações de números reais. Além disso, o indivíduo selecionado como mutante sofre alteração em apenas um de seus genes, o qual é escolhido de forma aleatória. Entretanto, esta modificação deve estar dentro do limite do intervalo permitido (FILHO, 2005).

Suponha que o indivíduo seja formado pelo seguinte vetor: $a = [a_1, a_2, a_3, ..., a_{n-1}, a_n]$, se a posição escolhida para mutação for a terceira, o novo indivíduo será composto por: $a' = [a_1, a_2, a'_3, ..., a_{n-1}, a_n]$. Esta alteração pode ser determinada por alguma função de distribuição uniforme.

A mutação gaussiana, assim como a mutação uniforme, também é utilizada para números reais. Nas estratégias evolutivas, a distribuição gaussiana ou distribuição normal, utiliza média zero e um desvio padrão σ, de acordo com a seguinte fórmula (LINDEN, 2012, p. 280):

$$N(0, \sigma, u) = \frac{e^{\frac{-1}{2}(\frac{u}{\sigma})^2}}{\sigma\sqrt{2\pi}} \tag{2.3}$$

em que u é um número real positivo entre zero e um, com exclusão deste, $u \in U[0, 1)$.

O gene x_i é então modificado somando a seu valor o resultado da distribuição normal.

$$x'_i = x_i + N(0, \sigma, x) \tag{2.4}$$

A mutação gaussiana pode ocorrer em um ou mais genes do indivíduo, onde também existe uma taxa de mutação determinada antecipadamente (ZUBEN, 2000).

2.2 Teste de Wilcoxon pareado

Em 1945, Frank Wilcoxon desenvolveu uma técnica não paramétrica para comparar se duas amostras são iguais estatisticamente ou se existem diferenças significativas entre elas (WILCOXON; KATTI; WILCOX, 1970). No caso dos algoritmos genéticos, utiliza-se os *fitness* das populações finais como amostras do teste.

O teste de Wilcoxon pareado consiste em, primeiramente, considerar que as amostras são simétricas em relação a zero. Em seguida, são formados pares de dois valores, um proveniente da primeira amostra, $A1$, e o outro originário da segunda amostra, $A2$. Calculando-se a diferença de cada par obtém-se uma terceira amostra, Dif (ESTATCAMP, 2005).

Amostra $A1$	18	22	9	24	13
Amostra $A2$	12	25	10	15	6
Dif	6	-3	-1	9	7

As seguintes hipóteses são estipuladas:

- $H_0 : \Delta = 0$ e $H_1 : \Delta \neq 0$;
- $H_0 : \Delta = 0$ e $H_1 : \Delta > 0$;
- $H_0 : \Delta = 0$ e $H_1 : \Delta < 0$.

em que Δ é a mediana das diferenças entre os pares das amostras, H_0 corresponde a hipótese nula e H_1 a hipótese alternativa (ESTATCAMP, 2005).

Se a hipótese nula for verdadeira, significa que as amostras não tem diferenças significativas. Entretanto, se essa hipótese for falsa, pode-se afirmar que existe diferença de localização entre as amostras (ESTATCAMP, 2005).

Após estabelecidas as hipóteses, Dif é ordenado de forma crescente considerando apenas os valores absolutos dos itens. A cada item é associado um posto P_i, o que equivale a posição deste em Dif.

Posteriormente, é verificado se a diferença entre as amostras são valores positivos ou negativos. Define-se então um indicador c_i, em que $c_i = 1$ se a diferença for maior que zero, e $c_i = 0$ caso contrário (WILCOXON; KATTI; WILCOX, 1970).

Dif	-1	-3	6	7	9
P_i	1	2	3	4	5
c_i	0	0	1	1	1

Calcula-se então os postos positivos da amostra Dif e os soma em uma variável T^+. Ou seja, T^+ é a soma de todos os postos em que a diferença entre as amostras resultou

em um número positivo.

$$T^{+} = \sum_{i=1}^{n} P_i c_i \tag{2.5}$$

em que n é o número de itens contidos na amostra.

$T^{+} = 1 * 0 + 2 * 0 + 3 * 1 + 4 * 1 + 5 * 1 = 3 + 4 + 5 = 12$

O software livre *R-project* é capaz de fazer esses cálculos utilizando a linguagem R, que de acordo com Beasley (2004, p. 2) "é uma linguagem e ambiente para computação estatística e gráficos".

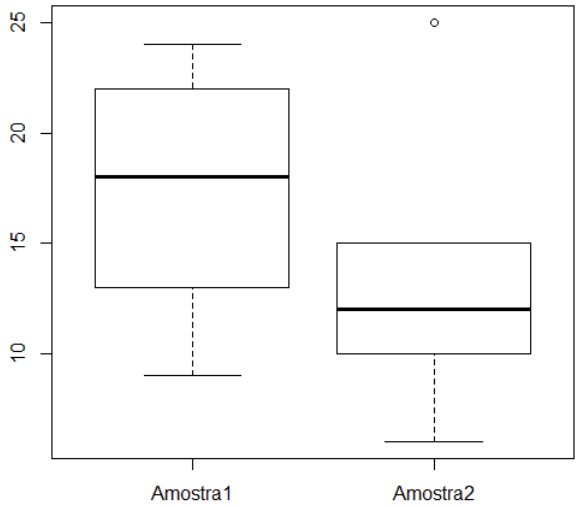
Nesse software, as duas amostras são passadas como vetores, como o seguinte exemplo:

$a1$<-$c(18.0, 22.0, 9.0, 24.0, 13.0)$

$a2$<-$c(12.0, 25.0, 10.0, 15.0, 6.0)$

Para melhor visualização dos dados, pode-se plotar um *boxplot*, que é um gráfico composto pelos quartis e as medianas das amostras. Quartis são as divisões em quatro partes iguais dos dados da amostra ordenados em forma crescente. Com o comando: $boxplot(a1, a2, names = c("Amostra1", "Amostra2"))$, obtém-se o *boxplot* da Figura 7.

Figura 7 – *Boxplot* do *R-project* para o exemplo



Fonte: a autora

No *R-project*, para utilizar o teste de Wilcoxon pareado, o objeto *paired* tem que ser verdadeiro (*true*). Sendo assim, a sintaxe para o teste é: $wilcox.test(a1, a2, paired = TRUE)$, o que irá retornar o valor da estatística V, que é o somatório dos postos positivos (T^{+}); e o *p-value* ou *P-valor*, o qual indica se as amostras possuem ou não diferenças significativas (BEASLEY, 2004).

O intervalo de confiança deste teste é de 95%. Isto significa que, se o *P-valor* $< 0,05$ as amostras são estatisticamente diferentes; em oposição, se *P-valor* for maior que 5%, denota que as amostras não apresentam diferenças significativas.

O resultado do exemplo retornado pelo *R-project* foi:

Wilcoxon signed rank test
data: a1 and a2
V = 12, p-value = 0.3125
alternative hypothesis: true location shift is not equal to 0.

Como *p-value* $= 0,3125$ $(> 0,05)$, é possível afirmar que as medianas das amostras não estão consideralvelmente distantes.

No caso de o retorno do *P-valor* ser um número menor que 0,05, para descobrir qual das amostras gerou uma melhor população, calcula-se as medianas das mesmas. Se o desejo é maximizar os resultados, a maior mediana indicará a melhor resposta; se é minimizar, a menor mediana é quem retrata a melhor amostra.

Para descobrir a mediana (*median*) das amostras pelo software *R-project*, basta executar o comando $summary(a)$, no qual a é a amostra passada por parâmetro. Para o exemplo acima, executando $summary(a1)$ a mediana da amostra $A1$ é igual a 18.0, e para $summary(a2)$ a mediana da amostra $A2$ é igual a 12.0.

3 Implementação da proposta

Neste capítulo está descrita em detalhes a implementação da proposta, as funções objetivo utilizadas e os operadores genéticos escolhidos. Adicionalmente, são apresentados os modelos e suas principais diferenças.

3.1 O programa implementado

A implementação do programa foi feita utilizando a ferramenta da Microsoft, Visual Studio 2015, no sistema operacional Windows 10 Home, 64-bit. A máquina usada tem processador Intel Core i5-3210M 2.50 GHz; 8 GB de memória RAM; e 720 GB de espaço de armazenamento.

A escolha da linguagem de programação C++ é um diferencial dos demais trabalhos que usualmente são encontrados em Java, como no livro do Linden (2012); em Matlab, com a ferramenta *GA solver* usada por Dao, Abhary e Marian (2016); e em estudos mais antigos, como em LISP, no livro do Koza (1992); em Pascal, no GA desenvolvido por Goldberg (1989), entre outros.

Nesta proposta, foram utilizadas seis diferentes funções para avaliar os indivíduos e assim poder comparar o desempenho dos dois modelos de algoritmos genéticos apresentados. As funções selecionadas, são funções de minimização aplicadas a problemas de otimização. Usando os alelos de cada indivíduo, as funções calculam e retornam um número real positivo, o qual indica o quão adaptado ele está ao ambiente.

Como Kumar, Kumar e Jarial (2016) mencionam em seu trabalho, algumas dessas funções são reconhecidas como funções de *benchmark* padrão na área de otimização (SURJANOVIC; BINGHAM, 2015). As funções adotadas foram:

(1) Função Griewank (Figura 8):

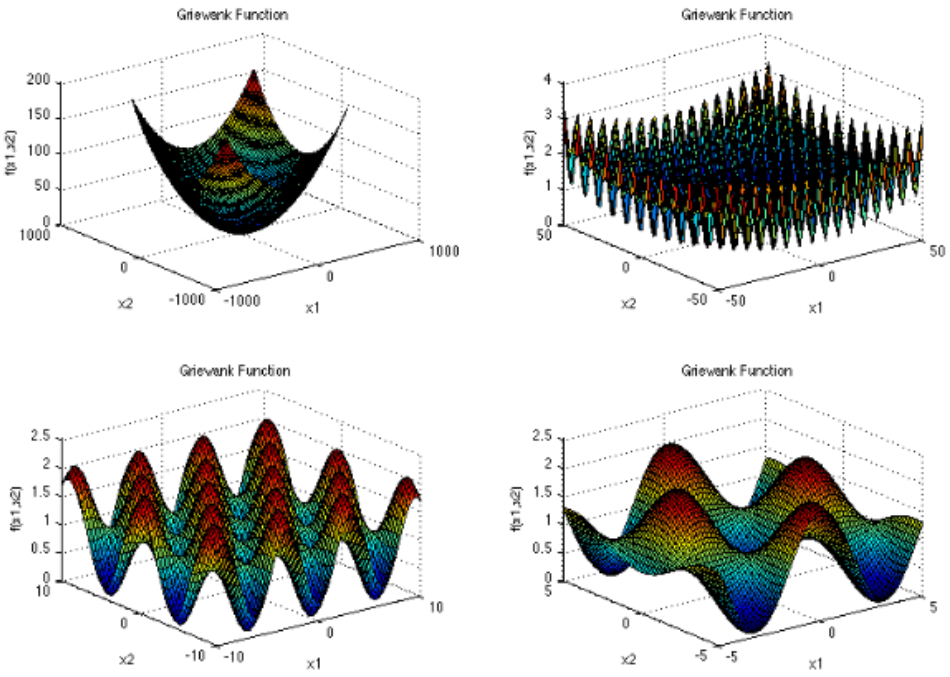
$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{3.1}$$

onde d é igual ao número de dimensões e $x_i \in$ [-600, 600], para todo i = 1, ..., d. A função Griewank é uma função de minimização não-linear e multimodal, que pode ter vários ótimos locais.

(2) Função Rastrigin (Figura 9):

$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)] \tag{3.2}$$

Figura 8 – Gráficos da função Griewank



Fonte: (SURJANOVIC; BINGHAM, 2015)

onde d é igual ao número de dimensões e $x_i \in$ [-5.12, 5.12], para todo i = 1, ..., d. A função Rastrigin soma o cosseno a função de De Jong que é:

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{3.3}$$

Esta função tem seu mínimo global quando $x_i = 0$, já a função Rastrigin, por causa da adição do cosseno, tem vários mínimos locais, os quais são distribuídos de forma regular (MOLGA; SMUTNICKI, 2005).

(3) Função Schwefel (Figura 10):

$$f(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|}) \tag{3.4}$$

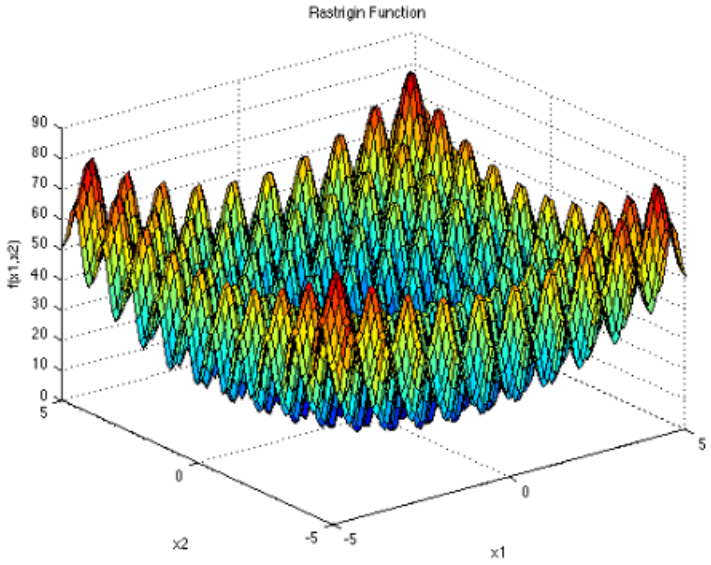
onde d é igual ao número de dimensões e $x_i \in$ [-500, 500], para todo i = 1, ..., d. A função Schwefel também é uma função não-linear e multimodal, entretanto não possui tantos mínimos locais quanto a função Rastrigin.

(4) Função Dixon-Price (Figura 11):

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2 \tag{3.5}$$

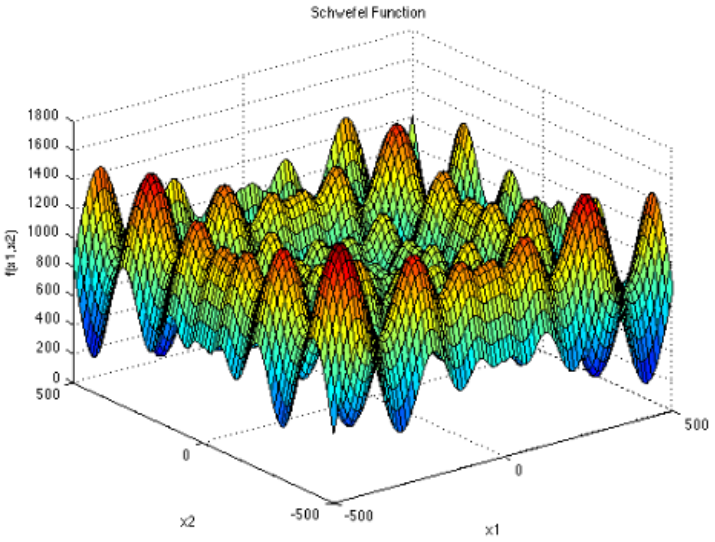
onde d é igual ao número de dimensões e $x_i \in$ [-10, 10], para todo i = 1, ..., d. A função Dixon-Price tem sua topologia formada por uma parábola longa, algo semelhante a um vale.

Figura 9 – Gráfico da função Rastrigin



Fonte: (SURJANOVIC; BINGHAM, 2015)

Figura 10 – Gráfico da função Schwefel



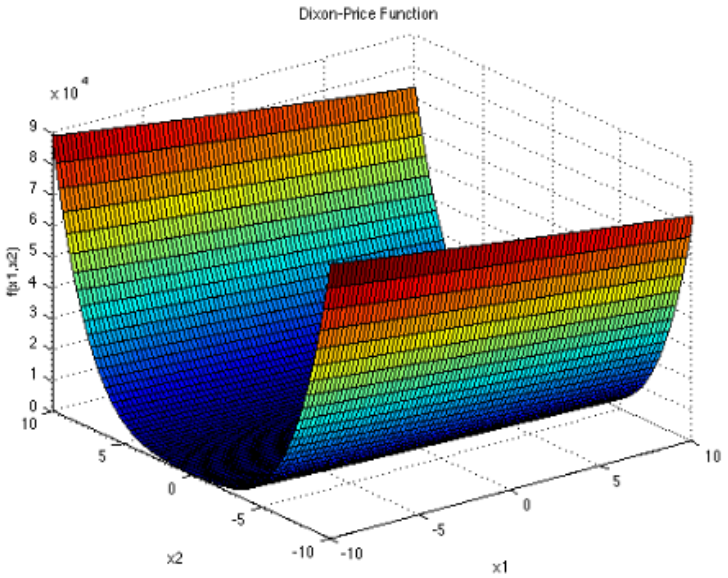
Fonte: (SURJANOVIC; BINGHAM, 2015)

(5) Função Rosenbrock (Figura 12):

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{3.6}$$

onde d é igual ao número de dimensões e $x_i \in$ [-2.048, 2.048], para todo i = 1, ..., d. A função Rosenbrock é mais uma das funções concebidas por DeJong (1975). Ela tem o formato de parábola como um vale estreito e de pouca inclinação, no qual alguns algoritmos

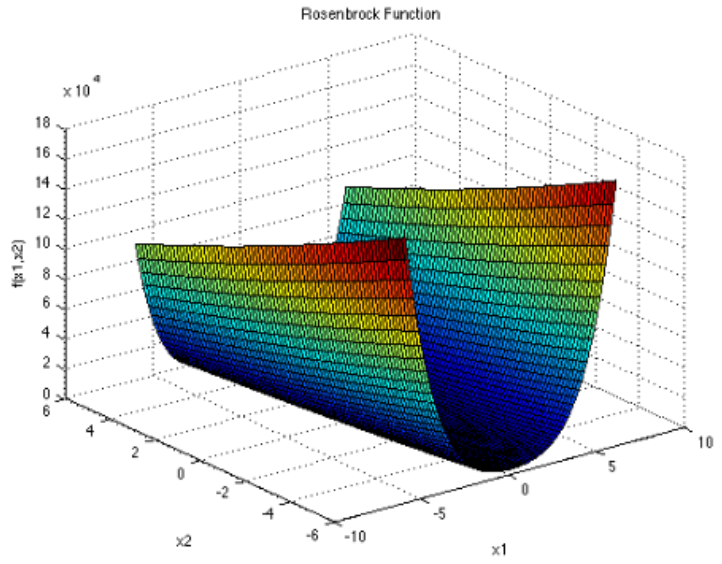
Figura 11 – Gráfico da função Dixon-Price



Fonte: (SURJANOVIC; BINGHAM, 2015)

podem ter dificuldade para encontrar um ótimo global (MOLGA; SMUTNICKI, 2005; OLIVEIRA, 2004).

Figura 12 – Gráfico da função Rosenbrock



Fonte: (SURJANOVIC; BINGHAM, 2015)

(6) Função Levy (Figura 13):

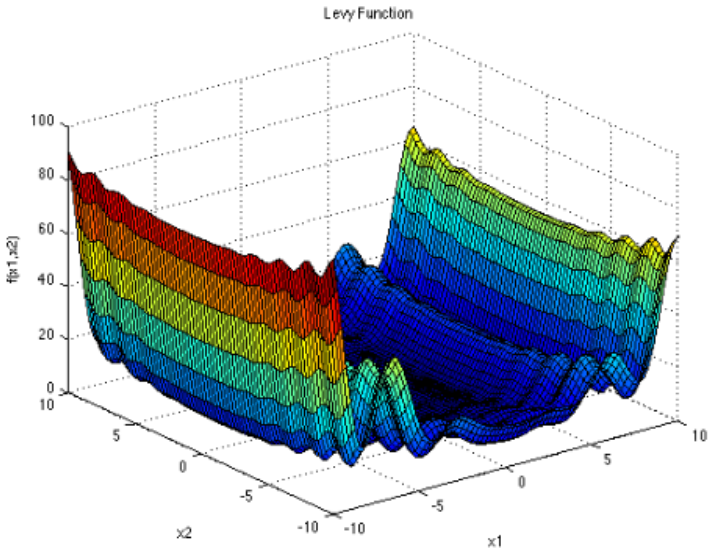
$$f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2[1 + 10\sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + \sin^2(2\pi w_d)] \quad (3.7)$$

onde

$$w_i = 1 + \frac{x_i - 1}{4} \tag{3.8}$$

para todo i = 1,..., d , onde d é igual ao número de dimensões e $x_i \in$ [-10, 10], para todo i = 1, ..., d. A função Levy é uma função multimodal que pode ser usada de forma bidimensional ou com diversas dimensões.

Figura 13 – Gráfico da função Levy



Fonte: (SURJANOVIC; BINGHAM, 2015)

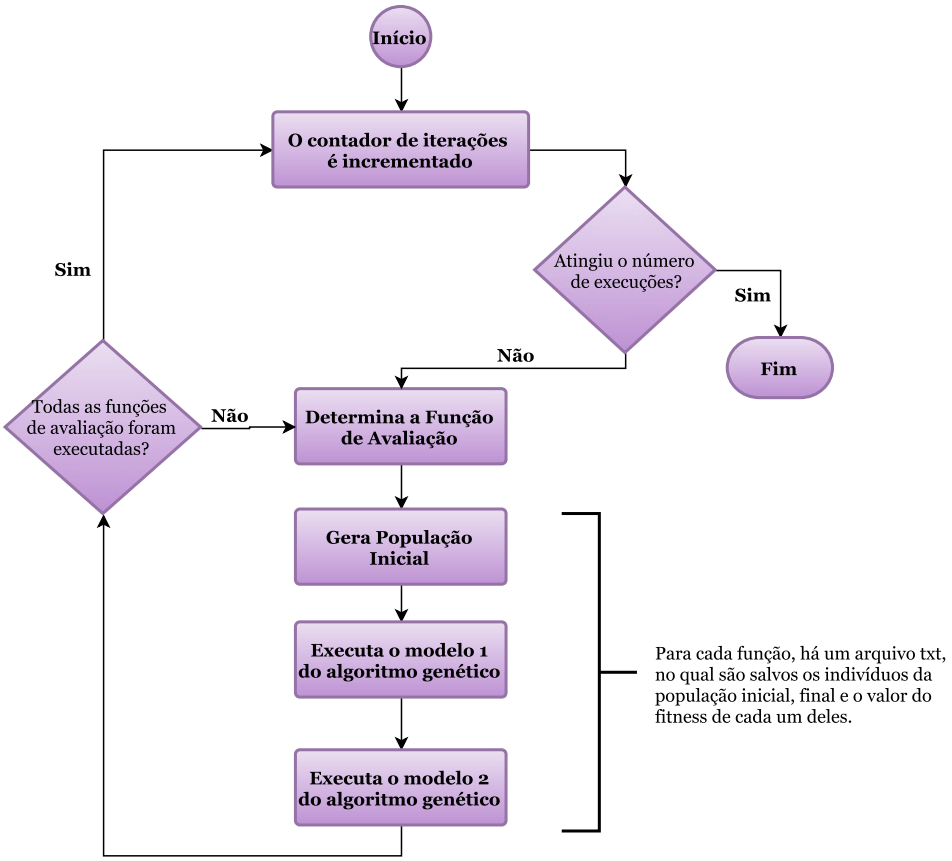
A cada iteração do programa, as funções objetivo são executadas pelos dois modelos. A cada função, uma população inicial é gerada, respeitando o intervalo dos números reais ao qual ela se limita. A mesma população inicial é o ponto de partida dos dois modelos, que ao final do número de gerações propostos, retorna uma população final (Figura 14). Desta forma, a cada execução do programa, seis populações iniciais são criadas e doze populações finais retornadas.

O código foi executado no total de 25 vezes (Algoritmo 1). Ao todo, foram geradas 150 populações iniciais (25 para cada função) e 300 populações finais (150 para cada modelo – 50 para cada função).

O tamanho da população é fixo e a cada geração tem-se n indivíduos ($n = 100$ para o primeiro teste e $n = 200$ para o segundo). Cada um desses indivíduos tem dez dimensões (genes). Até retornar o resultado final, o primeiro modelo passa por 600 gerações, e o segundo, passa por 300 gerações. A diferença no número de gerações existe para que o número de avaliações executadas por cada um dos modelos seja igual.

Para ambos os modelos, a cada geração, dois indivíduos são selecionados por elitismo. Ou seja, os dois melhores indivíduos de cada geração são passados adiante

Figura 14 – Fluxograma resumido do programa implementado



Fonte: a autora

Algorithm 1 Algoritmo do programa

```
1: for execução do 1 ao 25
2:     for função do 1 ao 6
3:         Gerar população inicial com indivíduos randômicos;
4:         Calcular o fitness de cada indivíduo;
5:
6:         Executar modelo SGA+E por 600 gerações;
7:         Calcular o fitness de cada indivíduo da população final;
8:
9:         Executar modelo FGA por 300 gerações; ▷ Usando a mesma população inicial
10:        Calcular o fitness de cada indivíduo da população final;
11:    end for
12: end for
```

sem sequer sofrer alteração em seus genes. Como as funções de avaliação são funções de minimização, o melhor indivíduo é aquele que apresenta o *fitness* de menor valor.

Por ser o objetivo deste estudo a comparação entre os dois modelos de algoritmos genéticos, os métodos de seleção, recombinação e mutação são os mesmos. A diferença entre os procedimentos se dá na ordem e em como os métodos são empregados.

A codificação utilizada foi a codificação real, na qual os alelos dos indivíduos são números reais pertencentes ao intervalo da função de avaliação usada pelo modelo.

O tipo de seleção escolhido é o *roullete wheel*. Como já visto anteriormente, este método permite que todos os indivíduos tenham chance de se tornar progenitores da próxima geração, mesmo aquele que não é um bom candidato. Entretanto, a probabilidade de ser selecionado pelo *roullete wheel* é proporcional a seu *fitness*, quanto melhor o *fitness*, maior a chance do indivíduo ser escolhido.

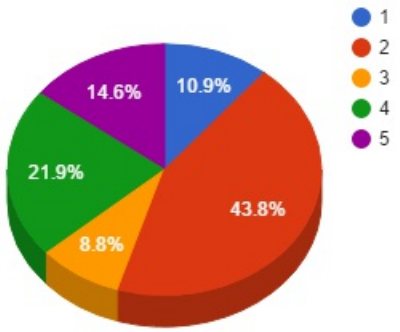
Na aplicação criada, como as funções objetivo são funções de minimização, o melhor valor do *fitness* é o menor encontrado. Dessa forma, para criar a roleta, o valor do *fitness* de cada indivíduo é somado em uma variável utilizando a seguinte fórmula:

$$fitnessTotal(f) = \sum_{i=1}^{n} \frac{1}{f_i} \tag{3.9}$$

em que n é o número total de indivíduos, no caso 20, i o indivíduo que está sendo analisado, e f_i o valor do *fitness* de cada um deles. Assim, o que tiver o menor *fitness* ocupará um espaço maior na roleta (Figura 15).

Figura 15 – Exemplo do espaço ocupado na roleta por um indivíduo

Indivíduo	Fitness	1 / Fitness
1	40	2.50E-02
2	10	1.00E-01
3	50	2.00E-02
4	20	5.00E-02
5	30	3.33E-02



Fonte: Adaptado de (FILHO, 2005, p. 15)

O método *roulette wheel* permite que indivíduos não adaptados sejam selecionados, porém os mais aptos ao ambiente (melhores avaliados), tem mais chance de se reproduzir e propagar seus genes.

Quanto ao operador genético para recombinação, foi adotado o *crossover* aritmético. Com este método, dois progenitores geram dois descendentes, o que pode ser visto como uma estratégia para manter o mesmo tamanho da população.

Com o *crossover* aritmético, parte dos alelos dos dois genes será somado para produzir os alelos dos filhos. A proporção de quanto do alelo dos pais será usada depende de uma variável aleatória uniforme, alfa ($\alpha \in [0, 1]$). Essa proporção pode ser diferente a cada recombinação.

A estratégia eleita para a mutação, foi a mutação gaussiana. Nesta estratégia, todos os genes do indivíduo podem sofrer uma alteração e a probabilidade de mutação escolhida foi de 30% ($p_m = 0{,}3$).

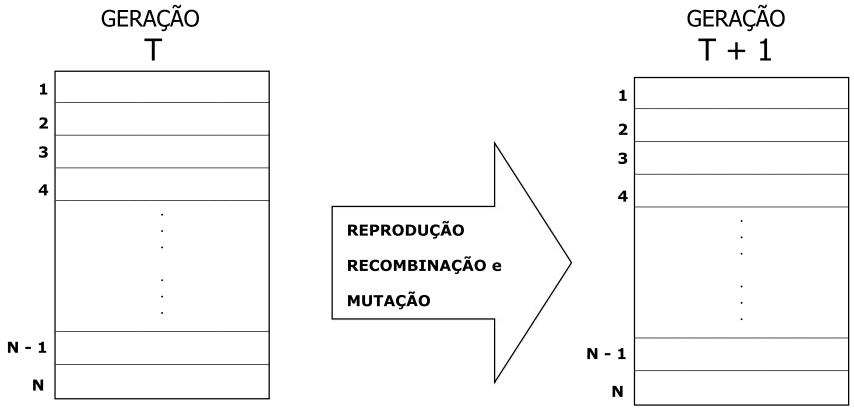
Na implementação feita, se nenhum gene sofrer alteração, escolhe-se aleatoriamente um *locus* e aplica-se a mutação gaussiana ao mesmo. Em outras palavras, ao ser selecionado como mutante, é garantido que pelo menos um alelo do indivíduo será mutado.

A mutação é um artifício utilizado, para que ao buscar um resultado, os algoritmos genéticos possam ter a chance de sair de máximos ou mínimos locais e explorar outras soluções.

3.2 Modelo 1: SGA+E

O primeiro modelo proposto foi baseado em dois algoritmos genéticos. Um deles é o *simple genetic algorithm* (SGA), criado por Goldberg (1989), ilustrado na Figura 16. No SGA, a partir de uma população inicial é criada uma nova e a cada execução sequencial dos procedimentos tem-se uma geração. Para que ocorra a transição de uma população para a próxima geração, essa deve passar pelo método de seleção; pela recombinação; e por último, pelo processo de mutação.

Figura 16 – *Simple Genetic Algorithm*



Fonte: (GOLDBERG, 1989, p. 61)

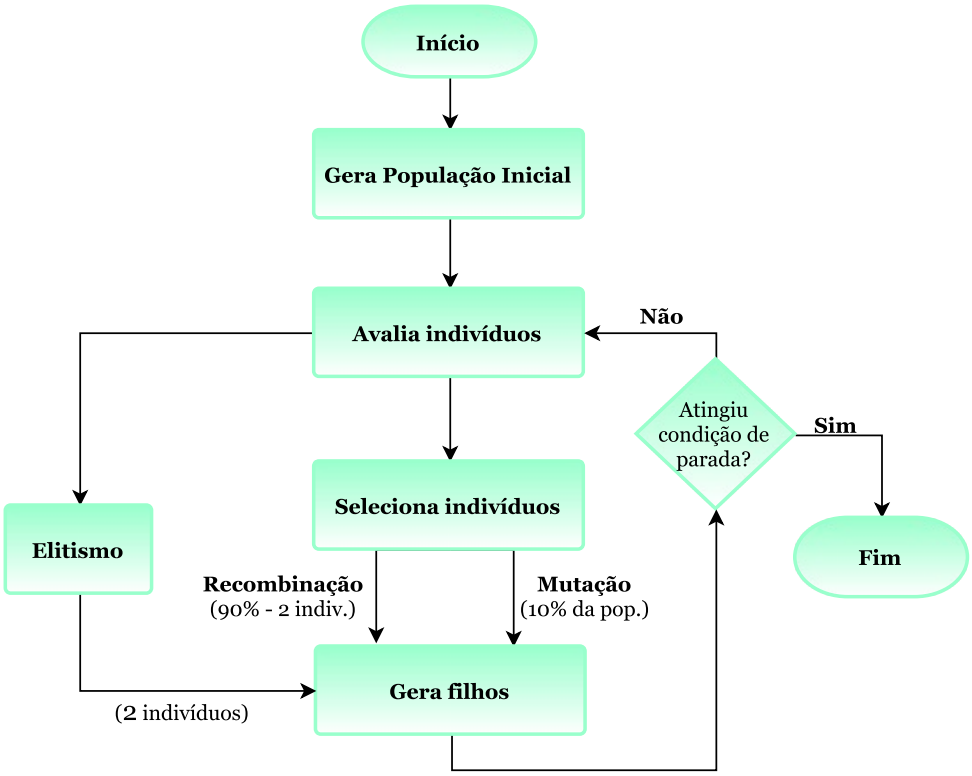
No SGA, assim como nos modelos deste trabalho, o método de seleção adotado é o *roullete wheel*. Para a recombinação, é utilizado o *crossover* de ponto único. Já a mutação, considerada um fator secundário na evolução das espécies por Goldberg (1989), foi implementada com baixa probabilidade. Foi usada a mutação pontual, uma vez que os indivíduos eram representados de forma binária.

O outro algoritmo utilizado, foi elaborado por DeJong (1975), com o nome de *Elitist Model R2*. A principal característica deste GA é a preservação dos indivíduos de melhor

fitness ao passar de uma geração para a próxima. DeJong (1975) propôs a introdução do elitismo, com o intuito de garantir que os melhores indivíduos não fossem descartados.

O modelo 1, *Simple Genetic Algorithm + Elitism*, SGA+E, soma o elitismo ao SGA, incorporando ainda algumas características das estratégias evolutivas. A primeira delas é a utilização de números reais para compor os indivíduos. A segunda e a terceira são os métodos de recombinação e mutação, já citados anteriormente.

Figura 17 – Fluxograma do modelo SGA+E



Fonte: a autora

Neste modelo, após a população inicial ser criada, começa-se a execução dos seguintes processos:

- os indivíduos são avaliados de acordo com a função objetivo;
- dois indivíduos são escolhidos por elitismo para passar para a próxima geração;
- são selecionados os progenitores pelo método *roullete wheel*;
- os indivíduos selecionados passam pelo processo de recombinação;
- 10% dos indivíduos da população sofrem mutação, com exceção dos escolhidos por elitismo.

Os processos se repetem até chegar a geração de número 600, e assim, retornam a população final (Figura 17 e Algoritmo 2).

Algorithm 2 Algoritmo do modelo SGA+E

```
1: População atual recebe a população inicial;
2: Cada indivíduo é avaliado pela função objetivo;
3: for geração do 1 ao 600
4:     Avalia os indivíduos da população gerada;
5:     Seleciona dois indivíduos por elitismo;
6:     Seleciona o restante pelo método roullete wheel;
7:     Recombina os indivíduos por crossover aritmético;        ▷ Indivíduos não eleitos
8:     Muta os indivíduos por mutação gaussiana;        ▷ Indivíduos não eleitos, nem
   recombinados
9: end for
```

3.3 Modelo 2: FGA

O segundo modelo foi proposto pelo professor e orientador deste trabalho, Douglas Nunes. A grande diferença deste para o anterior está na competição pela sobrevivência entre pais, filhos e mutantes. Por isso foi denominado FGA, o que corresponde a *full genetic algorithm*.

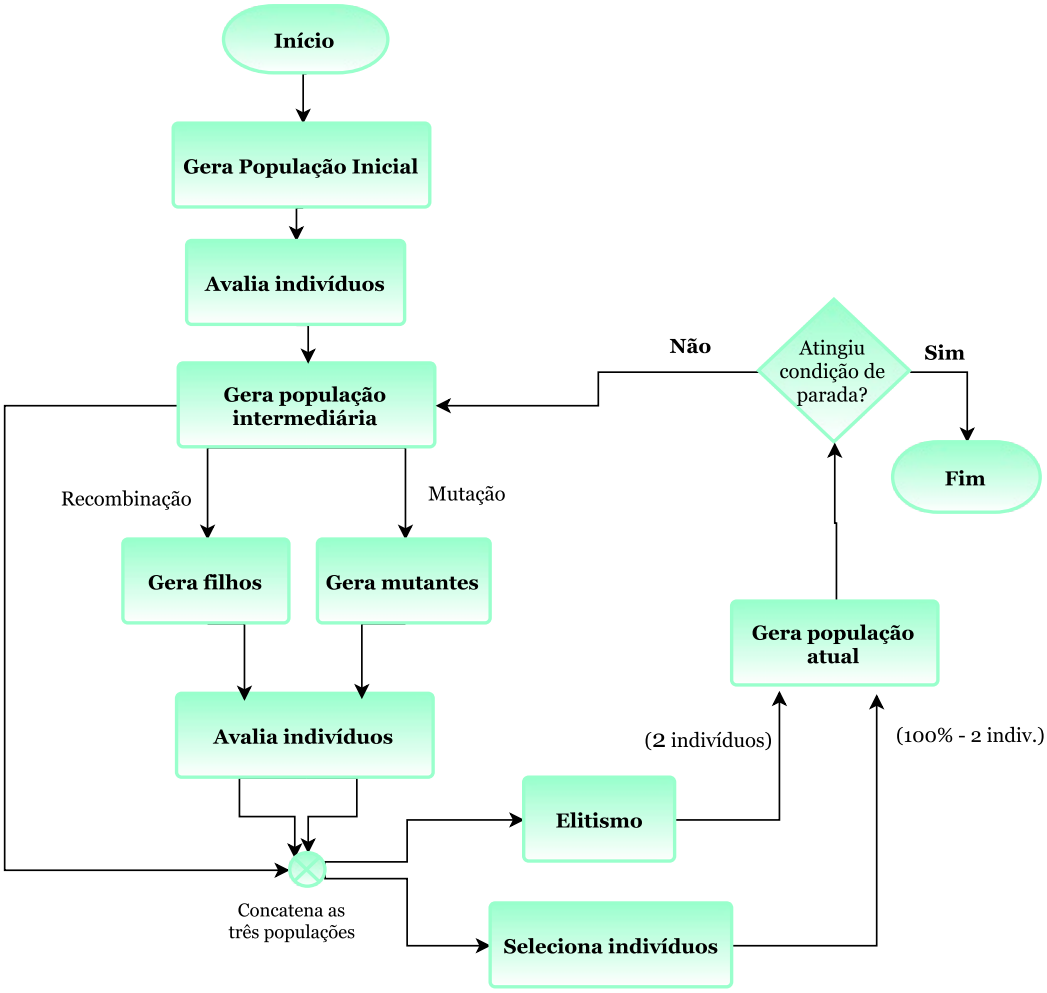
Assim como no SGA+E, o FGA é iniciado com uma população inicial, a mesma utilizada para o modelo 1. Esta população é avaliada e a partir dela são geradas mais duas: a população de filhos e a população de mutantes. Todos os filhos são gerados pelo processo de recombinação por *crossover* aritmético. E na população de mutantes, todos os indivíduos sofrem alteração.

Os filhos e os mutantes são avaliados pela função objetivo corrente. As três populações, pais (população inicial, na 1ª geração), filhos e mutantes, são aglomeradas em uma só. Em outras palavras, elas formam uma população única de tamanho três vezes maior que a inicial.

Para formar a próxima geração, com mesmo tamanho que a população inicial, dois indivíduos são selecionados por elitismo e o restante pelo método *roullete wheel*. Esta vai continuar gerando os filhos, os mutantes, e passando pelos mesmos processos até chegar a geração de número 300 (Figura 18 e Algoritmo 3).

Para que a comparação entre os dois modelos seja justa, a quantidade de avaliações feita pela função objetivo deve ser a mesma. No FGA, a cada chamada do procedimento de avaliação, o número de indivíduos avaliados é duas vezes maior que no SGA+E. Assim, a diferença no número de gerações se faz necessária para que a quantidade de avaliações seja igual nos dois modelos.

Figura 18 – Fluxograma do modelo FGA



Fonte: a autora

Algorithm 3 Algoritmo do modelo FGA

1: População atual recebe a população inicial;
2: Cada indivíduo é avaliado pela função objetivo;
3: **for** geração **do** 1 ao 300
4: Gera uma população recombinando os indivíduos por *crossover* aritmético;
5: Gera uma população mutando os indivíduos por mutação gaussiana;
6: Concatena a população atual + indivíduos recombinados + indivíduos mutados;
7: Avalia os indivíduos da população gerada;
8: Seleciona dois indivíduos por elitismo;
9: Seleciona o restante pelo método *roullete wheel*;
10: **end for**

4 Resultados

Nesta seção, serão analisados os *fitness* dos indivíduos das populações finais de todas as funções, os quais serão utilizados para comparar os modelos propostos neste trabalho.

4.1 Análise dos resultados

Como já mencionado, o programa implementado foi executado 25 vezes. Para cada execução, as seis funções objetivo foram utilizadas. Para cada função objetivo, uma população inicial foi criada, e para cada modelo, uma população final foi encontrada. Ao final de cada modelo, o *fitness* de todos os indivíduos foram salvos.

Para analisar e comparar os resultados obtidos, a cada iteração foi escolhido o indivíduo de melhor *fitness* retornado por cada modelo. Como as funções utilizadas são funções de minimização, o melhor valor avaliado é também o menor entre todos os outros.

No primeiro teste tem-se uma população formada por 100 indivíduos de dez dimensões. Com o resultado do melhor indivíduo de cada iteração, foram construídas tabelas para cada função utilizada. Essas tabelas são apresentadas na seção de Apêndices. Resumindo as tabelas formadas em apenas uma, formou-se a Tabela 1.

Tabela 1 – Resumo do teste 1

Função	Modelo	Menor valor	Média	Maior valor
1: Griewank	SGA+E	0.00E+00	**2.15E-03**	4.34E-03
	FGA	6.55E-01	2.34E+00	4.49E+00
2: Rastringin	SGA+E	9.01E-01	**4.37E+00**	8.24E+00
	FGA	1.36E-01	5.61E+00	1.06E+01
3: Schwefel	SGA+E	4.18E+03	4.19E+03	4.19E+03
	FGA	1.47E+03	**1.97E+03**	2.57E+03
4: Dixon-Price	SGA+E	7.62E-01	9.85E-01	1.54E+00
	FGA	6.19E-01	**7.35E-01**	1.09E+00
5: Rosenbrock	SGA+E	8.41E+00	1.29E+01	4.69E+01
	FGA	7.87E+00	**8.89E+00**	9.46E+00
6: Levy	SGA+E	8.25E-02	2.61E-01	5.01E-01
	FGA	4.44E-04	**1.93E-03**	4.28E-03

Fonte: a autora

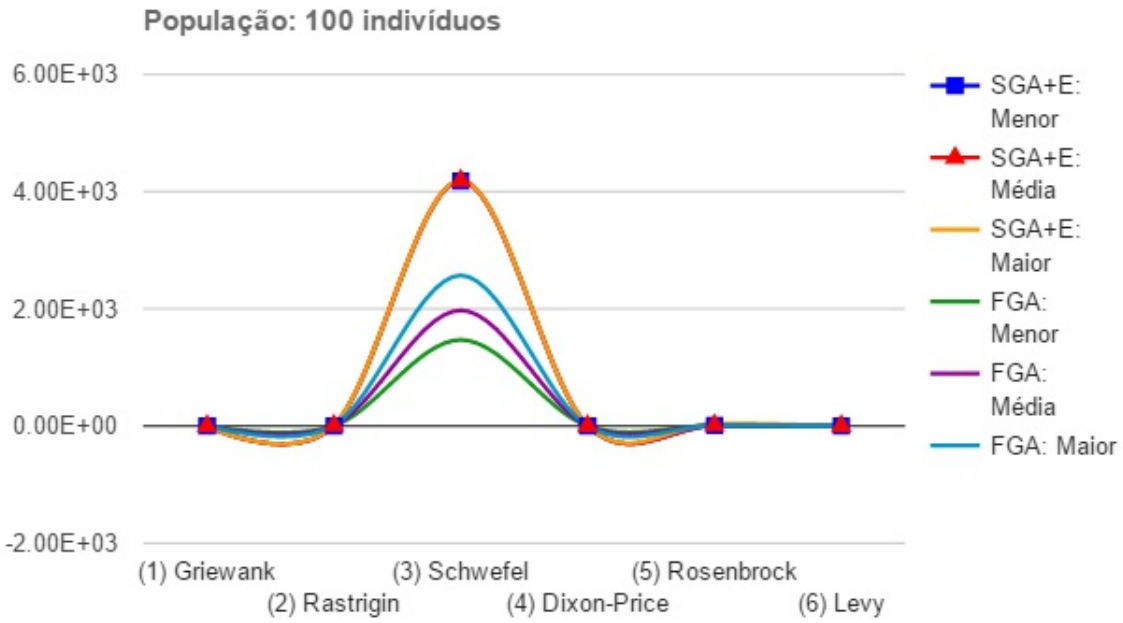
Na Tabela 1:

- o campo de menor valor, corresponde ao menor *fitness* encontrado pelo algoritmo;

- o campo média, corresponde à média aritmética dos 25 menores *fitness* resultantes da função (os números em destaque são os que apresentam menor valor quando os dois modelos são comparados);
- o campo de maior valor, corresponde ao maior *fitness*, dentre os menores de cada execução encontrados pelo algoritmo;
- o campo mediana, corresponde ao cálculo da mediana para os 25 menores *fitness* resultantes da função.

Para melhor visualização dos valores encontrados, tem-se o gráfico da Figura 19.

Figura 19 – Gráfico da tabela 1



Fonte: a autora

Tabela 2 – Aplicação do teste de Wilcoxon pareado

Função	Estatística	P-valor
1: Griewank	0	5.96E-08
2: Rastrigin	97	**8.02E-02**
3: Schwefel	325	5.96E-08
4: Dixon-Price	319	2.70E-05
5: Rosenbrock	320	5.96E-07
6: Levy	325	5.96E-08

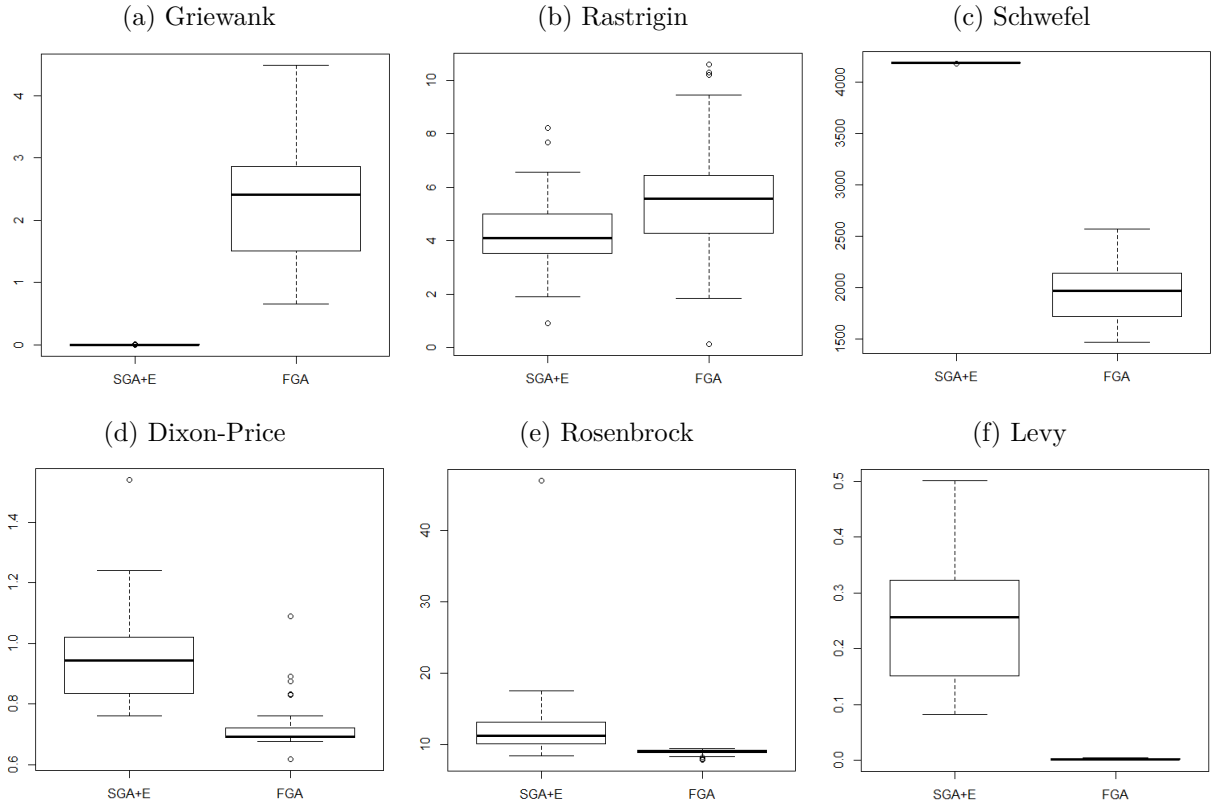
Fonte: a autora

A Tabela 2, apresenta as respostas obtidas pelo teste de Wilcoxon pareado para análise dos resultados utilizando o software *R-project*. Nessa tabela, apenas para a função

Rastrigin o *P-valor* encontrado não foi menor do que $0,05$, o que indica que os valores das amostras, para esta função, não apresentam diferença estatística.

Para observação das amostras, as estatísticas dessas foram plotadas no *R-project* e são mostradas na Figura 20.

Figura 20 – *Boxplots* do *R-project* para o teste 1



Fonte: a autora

Para definir qual dos dois modelos demonstrou melhor performance nas demais funções, as quais *P-valor* $< 0,05$, calculou-se as medianas das amostras. Os resultados do teste estão representados na Tabela 3.

Na função Griewank, o modelo SGA+E mostrou uma mediana menor, e portanto, um melhor desempenho uma vez que almejava-se minimizar os valores de adequação dos indivíduos. Entretanto, nas funções Schwefel, Dixon-Price, Rosenbrock e Levy, o modelo FGA foi quem retornou medianas menores, demonstrando uma melhor qualidade de resposta para estes casos.

Tabela 3 – Valor da mediana para o teste 1

Função	Modelo	Mediana
1: Griewank	SGA+E	**2.02E-03**
	FGA	2.41E+00
2: Rastringin	SGA+E	4.10E+00
	FGA	5.57E+00
3: Schwefel	SGA+E	4.19E+03
	FGA	**1.97E+03**
4: Dixon-Price	SGA+E	9.43E-01
	FGA	**6.93E-01**
5: Rosenbrock	SGA+E	1.12E+01
	FGA	**9.00E+00**
6: Levy	SGA+E	2.56E-01
	FGA	**1.42E-03**

Fonte: a autora

4.1.1 Teste adicional

Para verificar os resultados encontrados no primeiro teste, outro foi feito com mudança no tamanho da população. Diferentemente do experimento anterior, neste as populações são formadas por 200 indivíduos e 20 indivíduos sofrem mutação no modelo SGA+E.

As tabelas encontradas estão na seção de Apêndices. Resumidas em apenas uma, tem-se a Tabela 4. Nesta os valores em destaque representam as melhores médias, a cada função de avaliação, quando os dois modelos são comparados.

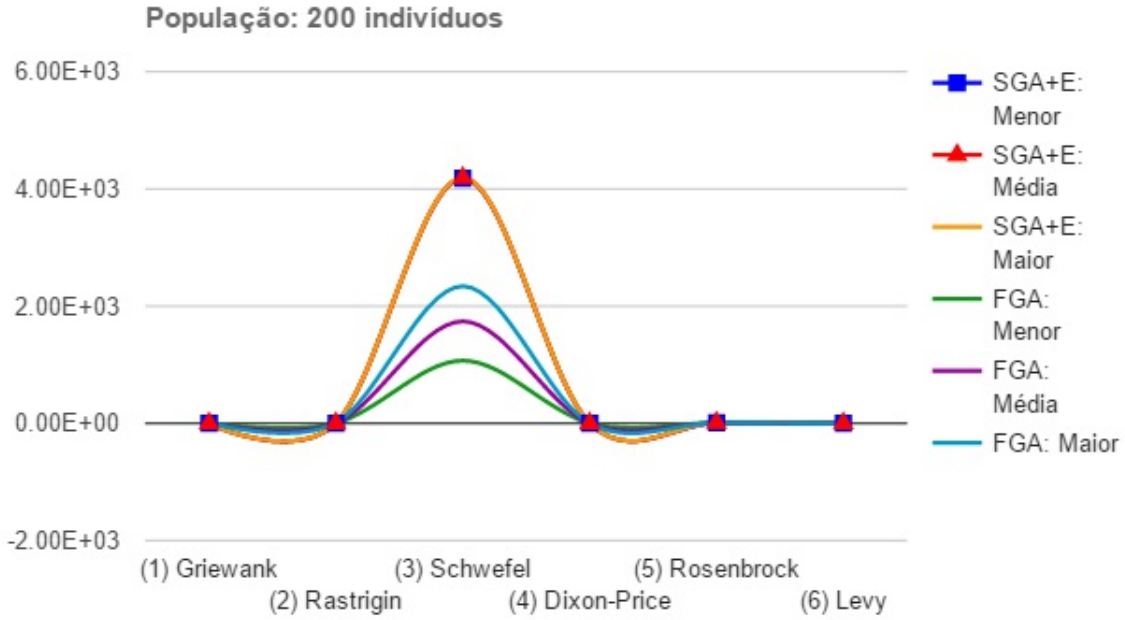
Tabela 4 – Resumo do teste 2

Função	Modelo	Menor valor	Média	Maior valor
1: Griewank	SGA+E	0.00E+00	**4.72E-03**	8.32E-03
	FGA	2.19E-01	1.22E+00	4.34E+00
2: Rastringin	SGA+E	4.84E-01	**2.46E+00**	5.23E+00
	FGA	1.16E+00	4.66E+00	8.48E+00
3: Schwefel	SGA+E	4.18E+03	4.19E+03	4.19E+03
	FGA	1.07E+03	**1.74E+03**	2.34E+03
4: Dixon-Price	SGA+E	7.22E-01	1.05E+00	1.94E+00
	FGA	6.72E-01	**7.48E-01**	1.02E+00
5: Rosenbrock	SGA+E	7.48E+00	1.04E+01	1.51E+01
	FGA	8.40E+00	**8.94E+00**	9.58E+00
6: Levy	SGA+E	1.42E-02	2.38E-02	3.53E-02
	FGA	3.58E-04	**1.37E-03**	2.21E-03

Fonte: a autora

Para melhor visualização dos valores encontrados, tem-se o gráfico da Figura 21.

Figura 21 – Gráfico da tabela 4



Fonte: a autora

A Tabela 5, apresenta as respostas obtidas pelo teste de Wilcoxon pareado para análise das populações. Nessa tabela, todas as medidas de *P-valor* encontradas são menores que $0,05$. Sendo assim, pôde-se definir qual dos dois modelos demonstrou melhor performance para todas as funções.

Tabela 5 – Aplicação do teste de Wilcoxon pareado

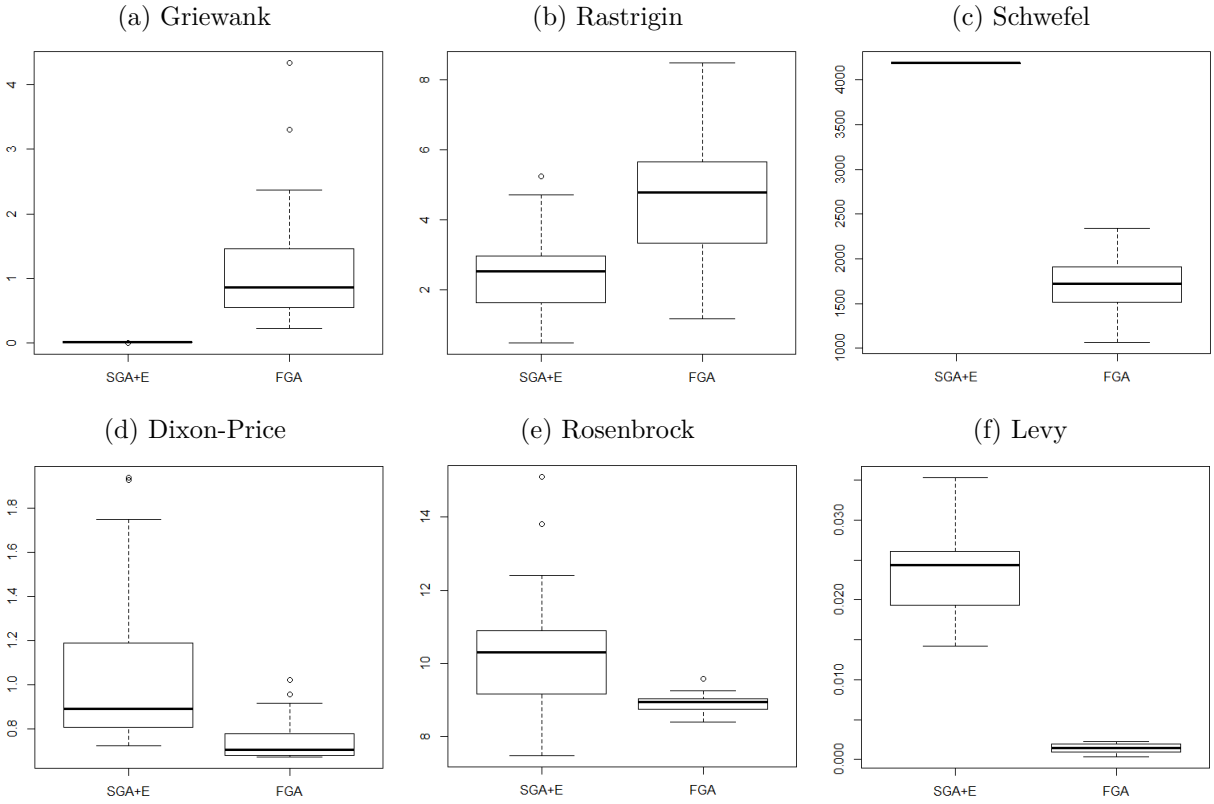
Função	Estatística	P-valor
1: Griewank	0	5.96E-08
2: Rastrigin	27	7.50E-05
3: Schwefel	325	1.31E-05
4: Dixon-Price	292	1.88E-04
5: Rosenbrock	297	8.80E-05
6: Levy	325	5.96E-08

Fonte: a autora

Para observação das amostras, as estatísticas dessas foram plotadas no *R-project* e são mostradas na Figura 22.

O cálculo das medianas das amostras estão representados na Tabela 6. Nesta, o modelo SGA+E encontrou melhores populações que o FGA quando a função de avaliação utilizada foi a função Griewank e a função Rastringin. Assim como no primeiro teste, para

Figura 22 – *Boxplots* do *R-project* para o teste 2



Fonte: a autora

as funções Schwefel, Dixon-Price, Rosenbrock e Levy, o modelo FGA demonstrou melhor performance que o modelo SGA+E.

Tabela 6 – Valor da mediana para o teste 2

Função	Modelo	Mediana
1: Griewank	SGA+E	**5.02E-03**
	FGA	8.62E-01
2: Rastringin	SGA+E	**2.52E+00**
	FGA	4.77E+00
3: Schwefel	SGA+E	4.19E+03
	FGA	**1.72E+03**
4: Dixon-Price	SGA+E	8.88E-01
	FGA	**7.02E-01**
5: Rosenbrock	SGA+E	1.03E+01
	FGA	**8.94E+00**
6: Levy	SGA+E	2.44E-02
	FGA	**1.46E-03**

Fonte: a autora

A definição dos parâmetros dos algoritmos genéticos depende do problema em questão e do desempenho desejado. De acordo com Mitchell (1998), muitos estudiosos afirmam que o ideal seria o uso de parâmetros autoadaptáveis, que se adaptem em tempo real dependendo da performance do programa.

4.1.2 Tempo de execução

O tempo de execução gasto por um programa é algo relevante para determinar sua performance. Desse modo, deve ser levado em conta o tempo consumido por cada modelo.

Para os testes realizados, foi encontrada a Tabela 7, na qual o tempo está representado em segundos.

Tabela 7 – Tempo de execução dos testes

Teste	Modelo	Tempo (s)	Nº de avaliações
1º teste	SGA+E	46,748	9.015.000
	FGA	55,271	
2º teste	SGA+E	95,897	18.030.000
	FGA	114,792	

Fonte: a autora

Nos testes executados a diferença do tempo de execução dos modelos é pequena. Para melhor observação do tempo, outras experiências foram feitas. Para estas, os indivíduos também eram compostos por dez genes; o tamanho da população foi variado, e consequentemente, o número de indivíduos que sofrem mutação no modelo SGA+E variou na mesma proporção (Tabela 8).

Tabela 8 – Tempo de execução para diferentes parâmetros

Tempo de execução dos modelos				
Parâmetros		Modelo	Tempo (s)	Nº de avaliações
Dimensões: 10	População: 400 indivíduos	SGA+E	205,727	36.060.000
		FGA	245,151	
	População: 500 indivíduos	SGA+E	285,356	45.075.000
		FGA	336,050	
	População: 600 indivíduos	SGA+E	339,259	54.090.000
		FGA	403,187	

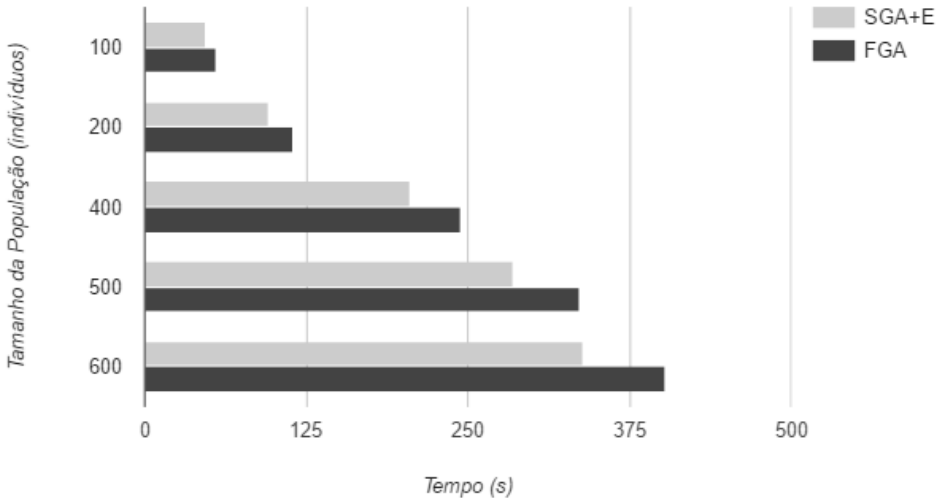
Fonte: a autora

Observa-se que o tempo de execução do modelo FGA foi maior que do modelo SGA+E para todos os testes. Isto ocorre devido a criação e avaliação de duas populações de mesmo tamanho da inicial a mais que o FGA efetua.

O número de avaliações é diretamente proporcional ao tamanho da população. Nas tabelas acima, nota-se, por exemplo, que o número de avaliações dobra quando o tamanho da população também dobra.

Para ilustração dos tempos de execução, com as populações de tamanho igual a 100 e 200 (do primeiro e segundo teste) até 600, tem-se o gráfico da Figura 23.

Figura 23 – Tempos de execução



Fonte: a autora

5 Conclusão

Para os dois testes executados, com a análise das amostras pelo teste de Wilcoxon pareado, apenas no primeiro teste, com a função Rastrigin como função de avaliação, que um modelo não se destacou em relação ao outro.

No restante, obteve-se melhores populações finais para o modelo SGA+E com as funções de avaliação: Griewank e Rastrigin, esta somente para o segundo teste. No entanto, o modelo FGA apresentou melhores populações finais para as funções de avaliação: Schwefel, Dixon-Price, Rosenbrock e Levy, em ambos os testes.

Com estes resultados, pode-se concluir que para as funções com muitos ótimos locais e que têm suas fórmulas compostas pela função cosseno, o modelo SGA+E se sobressaiu e produziu melhores resultados. Para as funções com um número menor de mínimos locais e que não dependem da função cosseno, o modelo FGA foi quem gerou os indivíduos mais adequados.

Constata-se assim, que nenhum dos modelos apresenta desempenho superior para todos os casos. Na verdade, a performance dos modelos depende das funções de avaliação utilizadas. Ou seja, as diferentes estruturas dos modelos é favorecida ou desfavorecida pela função objetivo escolhida.

Para trabalhos futuros, sugere-se as seguintes propostas:

- comparar os modelos propostos utilizando problemas multi-objetivos;
- utilizar outras funções de avaliação, como por exemplo, funções *benchmark* de maximização;
- agregar outros operadores genéticos aos modelos, como deleção e inversão;
- a utilização de parâmetros autoajustáveis;
- executar o programa um número maior de vezes;
- aumentar o número de dimensões dos indivíduos.

Referências

ARDJMAND, E. et al. Applying genetic algorithm to a new bi-objective stochastic model for transportation, location, and allocation of hazardous materials. *Expert Systems with Applications*, Elsevier, v. 51, p. 49–58, 2016. Acesso em: 24 maio 2016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417415008441>. Citado na página 12.

BACK, T.; HOFFMEISTER, F.; SCHWEFEL, H. A survey of evolution strategies. In: *Proceedings of the 4th international conference on genetic algorithms*. [s.n.], 1991. p. 2–9. Acesso em: 11 mar. 2016. Disponível em: <https://www.researchgate.net/profile/Hans-Paul_Schwefel/publication/2611416_A_Survey_of_Evolution_Strategies/links/00b4952e27b72c73fa000000.pdf>. Citado na página 17.

BAKER, J. E. Adaptive selection methods for genetic algorithms. In: HILLSDALE, NEW JERSEY. *Proceedings of an International Conference on Genetic Algorithms and their applications*. 1985. p. 101–111. Acesso em: 07 jun. 2016. Disponível em: <https://books.google.com.br/books?hl=en&lr=lang_en|lang_pt&id=lI17AgAAQBAJ&oi=fnd&pg=PA101&dq=j.+e.+baker+genetic+algorithm&ots=0Kp-b4O85s&sig=e6gyGO4ckL7cXb-fQY4nr-SMx2s#v=onepage&q=j.e.bakergeneticalgorithm&f=false>. Citado na página 18.

BEASLEY, C. R. Bioestatística usando r: Apostila de exemplos para o biólogo. 2004. Acesso em: 10 jul. 2016. Disponível em: <ftp://heanet.archive.gnewsense.org/disk1/CRAN/doc/contrib/Beasley-BioestatisticaUsandoR.pdf>. Citado na página 23.

BOOKER, L. B.; GOLDBERG, D. E.; HOLLAND, J. H. Classifier systems and genetic algorithms. *Artificial intelligence*, Elsevier, v. 40, n. 1, p. 235–282, 1989. Acesso em: 24 maio 2016. Disponível em: <http://www.sciencedirect.com/science/article/pii/0004370289900507>. Citado 2 vezes nas páginas 12 e 16.

COBB, H. G.; GREFENSTETTE, J. J. *Genetic algorithms for tracking changing environments*. [S.l.], 1993. Acesso em: 25 mar. 2016. Disponível em: <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA294075>. Citado na página 21.

COPPIN, B. *Inteligência artificial*. [S.l.]: LTC, 2010. Citado na página 16.

DAO, S. D.; ABHARY, K.; MARIAN, R. Maximising performance of genetic algorithm solver in matlab. *Eng. Lett*, v. 24, n. 1, p. 75–83, 2016. Acesso em: 06 jun. 2016. Disponível em: <https://www.researchgate.net/profile/Son_Dao4/publication/296198573_Maximising_Performance_of_Genetic_Algorithm_Solver_in_Matlab/links/56d38e4c08aeb52500d188bb.pdf>. Citado na página 25.

DEJONG, K. A. Analysis of the behavior of a class of genetic adaptive systems. 1975. Acesso em: 05 maio 2016. Disponível em: <https://deepblue.lib.umich.edu/handle/2027.42/4507>. Citado 7 vezes nas páginas 12, 16, 18, 19, 27, 32 e 33.

DIANATI, M.; SONG, I.; TREIBER, M. *An introduction to genetic algorithms and evolution strategies*. [S.l.], 2002. Acesso em: 24 maio 2016. Disponível em: <http://pami.uwaterloo.ca/~sd625/Students/msm/gaes.pdf>. Citado na página 11.

ESTATCAMP. *Técnicas não paramétricas*: Teste de wilcoxon pareado. 2005. Acesso em: 10 jul. 2016. Disponível em: <http://www.portalaction.com.br/tecnicas-nao-parametricas/teste-de-wilcoxon-pareado>. Citado na página 22.

FILHO, F. O. M. M. Aplicação de modelos de estimação de fitness em algoritmos genéticos. v. 184, 2005. Acesso em: 30 mar. 2016. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000377030>. Citado 6 vezes nas páginas 17, 18, 19, 20, 21 e 31.

FRANCIS, K. *Charles Darwin and the origin of species*. Greenwood Publishing Group, 2007. Acesso em: 24 maio 2016. Disponível em: <http://www.evolbiol.ru/docs/docs/large_files/charles_darwin.pdf>. Citado 2 vezes nas páginas 15 e 17.

GABRIEL, P. H. R.; DELBEM, A. C. B. *Fundamentos de algoritmos evolutivos*. ICMC-USP, 2008. Acesso em: 23 mar. 2016. Disponível em: <http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_75.pdf>. Citado na página 18.

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. [S.l.]: Addison-Wesley, 1989. Citado 7 vezes nas páginas 11, 12, 16, 18, 21, 25 e 32.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. [S.l.]: MIT Press, 1975–1992. Citado 3 vezes nas páginas 11, 15 e 16.

HOLLAND, J. H. Genetic algorithms computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, v. 267, n. 1, p. 44–50, 1992. Acesso em: 24 maio 2016. Disponível em: <http://papers.cumincad.org/data/works/att/7e68.content.pdf>. Citado na página 16.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992. v. 1. Acesso em: 04 jun. 2016. Disponível em: <https://books.google.com.br/books?hl=en&lr=lang_en|lang_pt&id=Bhtxo60BV0EC&oi=fnd&pg=PP17&dq=Genetic+Programming+Koza&ots=9pgSdtb5LU&sig=c_iIeitc5_wKloDsmFhLFvaIX_8#v=onepage&q=lisp&f=false>. Citado na página 25.

KOZENY, V. Genetic algorithms for credit scoring: Alternative fitness function performance comparison. *Expert Systems with Applications*, Elsevier, v. 42, n. 6, p. 2998–3004, 2015. Acesso em: 24 maio 2016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417414007143>. Citado na página 17.

KUMAR, A.; KUMAR, D.; JARIAL, S. A comparative analysis of selection schemes in the artificial bee colony algorithm. *Computación y Sistemas*, v. 20, n. 1, 2016. Acesso em: 09 jun. 2016. Disponível em: <http://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/2228/2104>. Citado na página 25.

LINDEN, R. *Algoritmos genéticos (3a ediçao)*. [S.l.]: Brasport, 2012. 279–291 p. Citado 3 vezes nas páginas 16, 21 e 25.

MITCHELL, M. *An introduction to genetic algorithms*. MIT press, 1998. Acesso em: 24 maio 2016. Disponível em: <http://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf>. Citado 7 vezes nas páginas 13, 15, 16, 17, 18, 20 e 42.

MITCHELL, M.; FORREST, S.; HOLLAND, J. H. The royal road for genetic algorithms: Fitness landscapes and ga performance. In: CAMBRIDGE: THE MIT PRESS. *Proceedings of the first european conference on artificial life*. 1992. p. 245–254. Acesso em: 25 maio 2016. Disponível em: <http://web.cecs.pdx.edu/~mm/ecal92.pdf>. Citado na página 11.

MOLGA, M.; SMUTNICKI, C. Test functions for optimization needs. *Test functions for optimization needs*, 2005. Acesso em: 14 jun. 2016. Disponível em: <http://dl.dadvarmarzieh.com/File/Booklet/13940824131547698.pdf>. Citado 2 vezes nas páginas 26 e 28.

OLIVEIRA, A. C. Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos. *Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos*, 2004. Acesso em: 14 jun. 2016. Disponível em: <http://mtc-m16.sid.inpe.br/col/sid.inpe.br/jeferson/2004/08.16.16.50/doc/publicacao.pdf>. Citado na página 28.

OLIVEIRA, K. L. M. d. Otimização da rede coletora de média tensão de parques eólicos utilizando um algoritmo genético modificado. Universidade Federal de Juiz de Fora, 2016. Acesso em: 14 jun. 2016. Disponível em: <http://repositorio.ufjf.br:8080/xmlui/handle/ufjf/1337>. Citado na página 13.

PAIVA, J. L. d. *Um algoritmo genético híbrido para supressão de ruídos em imagens*. Tese (Doutorado) — Universidade de São Paulo, 2016. Acesso em: 24 maio 2016. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-11042016-105926/en.php>. Citado na página 12.

PEREIRA, C. M. d. N. A. et al. O impacto do processamento paralelo na otimização de projeto neutrônico de reator através de algoritmo genético. Instituto de Engenharia Nuclear, 2016. Acesso em: 14 jun. 2016. Disponível em: <http://carpedien.ien.gov.br:8080/handle/ien/1666>. Citado na página 12.

POZO, A. et al. Computação evolutiva. *Universidade Federal do Paraná, 61p.(Grupo de Pesquisas em Computação Evolutiva, Departamento de Informática-Universidade Federal do Paraná)*, 2005. Acesso em: 28 mar. 2016. Disponível em: <http://www.inf.ufpr.br/aurora/disciplinas/datamining/Ceapostila.pdf>. Citado na página 11.

SURJANOVIC, S.; BINGHAM, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets: Optimization Test Problems*. 2015. Acesso em: 28 mar. 2016. Disponível em: <http://www.sfu.ca/~ssurjano/optimization.html>. Citado 5 vezes nas páginas 25, 26, 27, 28 e 29.

TANOMARU, J. Motivação, fundamentos e aplicações de algoritmos genéticos. In: *II Congresso Brasileiro de Redes Neurais*. [s.n.], 1995. p. 373–403. Acesso em: 31 mar. 2016. Disponível em: <http://www.inf.ufrgs.br/~danielnm/ia/tutorial_ag.pdf>. Citado na página 11.

VAVAK, F.; FOGARTY, T. C. Comparison of steady state and generational genetic algorithms for use in nonstationary environments. In: IEEE. *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. 1996. p. 192–195. Acesso em: 25 mar. 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=542359&url=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=542359>. Citado 2 vezes nas páginas 12 e 13.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing*, v. 4, p. 65–85, 1994. Acesso em: 29 mar. 2016. Disponível em: <http://cobweb.cs.uga.edu/~potter/CompIntell/ga_tutorial.pdf>. Citado 3 vezes nas páginas 16, 17 e 21.

WILCOXON, F.; KATTI, S.; WILCOX, R. A. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics*, Markham Publishing Co. Chicago, v. 1, p. 171–259, 1970. Acesso em: 10 jul. 2016. Disponível em: <https://books.google.com.br/books?hl=en&lr=lang_en|lang_pt&id=TH4A_39UkTkC&oi=fnd&pg=PA171&dq=Wilcoxon+Signed+Rank+Test&ots=5DTCyF_Rj3&sig=gdGRnn8NfBqeZ_E7X6WqEevi4Mc#v=onepage&q=Wilcoxon%20Signed%20Rank%20Test&f=false>. Citado na página 22.

ZHANG, S.-p.; XIN, X.-k. Pollutant source identification model for water pollution incidents in small straight rivers based on genetic algorithm. *Applied Water Science*, Springer, p. 1–9, 2016. Acesso em: 24 maio 2016. Disponível em: <http://link.springer.com/article/10.1007/s13201-015-0374-z>. Citado na página 12.

ZUBEN, F. J. V. Computação evolutiva: uma abordagem pragmática. *Anais da I Jornada de Estudos em Computação de Piracicaba e Região (1a JECOMP)*, v. 1, p. 25–45, 2000. Acesso em: 08 jun. 2016. Disponível em: <ftp://calhau.dca.fee.unicamp.br/pub/docs/vonzuben/ia013_1s07/tutorialEC.pdf>. Citado 2 vezes nas páginas 19 e 21.

Apêndices

APÊNDICE A – Tabelas do primeiro teste

Nesta seção estão as tabelas com o melhor indivíduo de cada modelo, a cada iteração. As tabelas estão divididas de acordo com a função objetivo utilizada. Lembrando que para este teste, a população é formada por 100 indivíduos de dez dimensões; a cada geração, dois são passados por elitismo, e no modelo SGA+E, dez sofrem mutação.

Tabela 9 – Resultado das populações finais utilizando a função Griewank (100 indivíduos)

Função 1: Griewank		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	0.00E+00	1.09E+00
2	2.08E-03	1.77E+00
3	2.02E-03	1.52E+00
4	1.72E-03	2.47E+00
5	1.74E-03	4.49E+00
6	2.65E-03	2.87E+00
7	1.63E-03	2.00E+00
8	3.58E-03	2.26E+00
9	2.05E-03	1.20E+00
10	1.75E-03	2.41E+00
11	1.20E-03	1.51E+00
12	1.73E-03	2.76E+00
13	1.92E-03	6.55E-01
14	2.06E-03	4.23E+00
15	2.48E-03	4.09E+00
16	1.72E-03	3.02E+00
17	2.95E-03	2.64E+00
18	2.18E-03	2.55E+00
19	1.70E-03	1.43E+00
20	2.24E-03	3.11E+00
21	1.90E-03	2.85E+00
22	1.91E-03	8.34E-01
23	3.12E-03	1.51E+00
24	3.05E-03	2.99E+00
25	4.34E-03	2.15E+00
Menor valor:	0.00E+00	6.55E-01
Média:	2.15E-03	2.34E+00
Maior valor:	4.34E-03	4.49E+00
Mediana:	2.02E-03	2.41E+00

Fonte: a autora

Tabela 10 – Resultado das populações finais utilizando a função Rastrigin (100 indivíduos)

Função 2: Rastrigin		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	7.68E+00	4.28E+00
2	4.90E+00	9.46E+00
3	2.88E+00	4.32E+00
4	4.10E+00	8.69E+00
5	3.74E+00	3.26E+00
6	2.38E+00	4.59E+00
7	8.24E+00	1.36E-01
8	3.69E+00	3.21E+00
9	4.99E+00	5.57E+00
10	5.88E+00	4.27E+00
11	5.47E+00	6.06E+00
12	4.57E+00	6.46E+00
13	3.98E+00	4.27E+00
14	3.54E+00	4.35E+00
15	4.93E+00	7.29E+00
16	6.58E+00	6.18E+00
17	2.86E+00	1.83E+00
18	3.82E+00	5.64E+00
19	4.20E+00	3.08E+00
20	1.89E+00	1.06E+01
21	9.01E-01	6.37E+00
22	2.84E+00	1.03E+01
23	3.68E+00	1.02E+01
24	4.86E+00	6.16E+00
25	6.54E+00	3.61E+00
Menor valor:	9.01E-01	1.36E-01
Média:	4.37E+00	5.61E+00
Maior valor:	8.24E+00	1.06E+01
Mediana:	4.10E+00	5.57E+00

Fonte: a autora

Tabela 11 – Resultado das populações finais utilizando a função Schwefel (100 indivíduos)

Função 3: Schwefel		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	4.19E+03	1.80E+03
2	4.19E+03	1.90E+03
3	4.19E+03	2.28E+03
4	4.19E+03	2.13E+03
5	4.19E+03	2.17E+03
6	4.19E+03	2.57E+03
7	4.19E+03	2.51E+03
8	4.19E+03	1.85E+03
9	4.19E+03	2.01E+03
10	4.19E+03	1.92E+03
11	4.19E+03	2.22E+03
12	4.19E+03	1.68E+03
13	4.19E+03	2.02E+03
14	4.19E+03	1.97E+03
15	4.19E+03	1.66E+03
16	4.19E+03	2.07E+03
17	4.19E+03	2.09E+03
18	4.19E+03	1.47E+03
19	4.18E+03	1.72E+03
20	4.19E+03	1.61E+03
21	4.19E+03	1.67E+03
22	4.18E+03	1.88E+03
23	4.19E+03	1.70E+03
24	4.19E+03	2.31E+03
25	4.19E+03	2.14E+03
Menor valor:	4.18E+03	1.47E+03
Média:	4.19E+03	1.97E+03
Maior valor:	4.19E+03	2.57E+03
Mediana:	4.19E+03	1.97E+03

Fonte: a autora

Tabela 12 – Resultado das populações finais utilizando a função Dixon-Price (100 indivíduos)

Função 4: Dixon-Price		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	7.98E-01	6.87E-01
2	1.03E+00	7.08E-01
3	8.01E-01	8.92E-01
4	1.24E+00	7.61E-01
5	9.20E-01	7.04E-01
6	8.40E-01	6.94E-01
7	9.43E-01	6.80E-01
8	8.03E-01	6.83E-01
9	9.10E-01	6.93E-01
10	9.41E-01	6.93E-01
11	1.54E+00	6.95E-01
12	1.02E+00	6.90E-01
13	9.89E-01	7.02E-01
14	8.26E-01	6.19E-01
15	8.35E-01	8.29E-01
16	9.65E-01	6.90E-01
17	1.54E+00	8.74E-01
18	1.20E+00	7.21E-01
19	1.13E+00	1.09E+00
20	8.53E-01	8.33E-01
21	9.68E-01	6.84E-01
22	7.64E-01	6.91E-01
23	7.62E-01	6.93E-01
24	1.00E+00	6.77E-01
25	1.00E+00	6.90E-01
Menor valor:	7.62E-01	6.19E-01
Média:	9.85E-01	7.35E-01
Maior valor:	1.54E+00	1.09E+00
Mediana:	9.43E-01	6.93E-01

Fonte: a autora

Tabela 13 – Resultado das populações finais utilizando a função Rosenbrock (100 indivíduos)

Função 5: Rosenbrock		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	1.35E+01	9.03E+00
2	1.31E+01	8.30E+00
3	9.21E+00	8.85E+00
4	1.01E+01	9.08E+00
5	1.03E+01	9.29E+00
6	1.26E+01	9.06E+00
7	1.12E+01	8.97E+00
8	1.02E+01	9.43E+00
9	1.12E+01	7.97E+00
10	1.17E+01	9.14E+00
11	1.75E+01	8.15E+00
12	1.19E+01	7.87E+00
13	1.42E+01	8.94E+00
14	4.69E+01	9.46E+00
15	8.86E+00	8.58E+00
16	1.11E+01	9.01E+00
17	1.32E+01	9.28E+00
18	1.40E+01	8.81E+00
19	9.84E+00	9.44E+00
20	8.56E+00	8.50E+00
21	8.41E+00	9.00E+00
22	1.18E+01	8.79E+00
23	1.09E+01	9.10E+00
24	9.86E+00	8.89E+00
25	1.10E+01	9.25E+00
Menor valor:	8.41E+00	7.87E+00
Média:	1.28E+01	8.89E+00
Maior valor:	4.69E+01	9.46E+00
Mediana:	1.12E+01	9.00E+00

Fonte: a autora

Tabela 14 – Resultado das populações finais utilizando a função Levy (100 indivíduos)

Função 6: Levy		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	8.27E-02	7.31E-04
2	4.43E-01	9.22E-04
3	1.84E-01	1.26E-03
4	2.26E-01	4.28E-03
5	1.39E-01	6.31E-04
6	2.65E-01	1.22E-03
7	3.84E-01	2.61E-03
8	3.23E-01	3.43E-03
9	5.01E-01	3.57E-03
10	1.51E-01	2.83E-03
11	4.67E-01	1.42E-03
12	3.04E-01	8.29E-04
13	1.61E-01	3.23E-03
14	1.99E-01	4.44E-04
15	1.39E-01	6.56E-04
16	2.26E-01	1.33E-03
17	1.10E-01	4.64E-04
18	4.77E-01	3.15E-03
19	2.68E-01	1.99E-03
20	2.87E-01	8.22E-04
21	8.25E-02	3.71E-03
22	1.02E-01	1.70E-03
23	4.67E-01	3.78E-03
24	2.70E-01	2.31E-03
25	2.56E-01	1.01E-03
Menor valor:	8.25E-02	4.44E-04
Média:	2.61E-01	1.93E-03
Maior valor:	5.01E-01	4.28E-03
Mediana:	2.56E-01	1.42E-03

Fonte: a autora

APÊNDICE B – Tabelas do teste adicional

Nesta seção estão as tabelas do segundo teste com o melhor indivíduo de cada modelo, a cada iteração. As tabelas estão divididas de acordo com a função objetivo utilizada. Lembrando que para este teste, a população é formada por 200 indivíduos; a cada geração, dois são passados por elitismo, e no modelo SGA+E, 20 sofrem mutação.

Tabela 15 – Resultado das populações finais utilizando a função Griewank (200 indivíduos)

Função 1: Griewank		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	0.00E+00	1.46E+00
2	3.17E-03	1.87E+00
3	4.32E-03	1.15E+00
4	6.23E-03	1.41E+00
5	5.13E-03	7.11E-01
6	3.77E-03	1.60E+00
7	5.26E-03	2.37E+00
8	5.47E-03	4.52E-01
9	5.34E-03	4.34E+00
10	7.86E-03	2.36E+00
11	6.73E-03	6.89E-01
12	6.40E-03	2.19E-01
13	2.47E-03	8.62E-01
14	3.47E-03	2.19E-01
15	5.48E-03	9.52E-01
16	3.44E-03	5.02E-01
17	8.32E-03	4.93E-01
18	5.02E-03	3.31E+00
19	3.89E-03	9.81E-01
20	3.97E-03	8.33E-01
21	3.13E-03	3.89E-01
22	6.01E-03	6.57E-01
23	2.82E-03	5.42E-01
24	5.44E-03	1.30E+00
25	4.92E-03	7.55E-01
Menor valor:	0.00E+00	2.19E-01
Média:	4.72E-03	1.22E+00
Maior valor:	8.32E-03	4.34E+00
Mediana:	5.02E-03	8.62E-01

Fonte: a autora

Tabela 16 – Resultado das populações finais utilizando a função Rastrigin (200 indivíduos)

Função 2: Rastrigin		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	5.23E+00	5.32E+00
2	4.47E+00	5.88E+00
3	1.62E+00	8.20E+00
4	1.65E+00	3.51E+00
5	4.72E+00	1.16E+00
6	7.51E-01	3.22E+00
7	2.64E+00	4.56E+00
8	1.73E+00	3.49E+00
9	1.72E+00	5.66E+00
10	2.63E+00	7.73E+00
11	4.84E-01	3.50E+00
12	2.14E+00	5.15E+00
13	3.43E+00	6.20E+00
14	2.62E+00	5.18E+00
15	4.57E+00	4.77E+00
16	3.64E+00	5.51E+00
17	2.33E+00	1.51E+00
18	2.69E+00	3.61E+00
19	8.94E-01	8.20E+00
20	2.90E+00	2.36E+00
21	1.11E+00	3.32E+00
22	2.95E+00	8.48E+00
23	1.02E+00	2.13E+00
24	2.52E+00	5.30E+00
25	9.68E-01	2.54E+00
Menor valor:	4.84E-01	1.16E+00
Média:	2.46E+00	4.66E+00
Maior valor:	5.23E+00	8.48E+00
Mediana:	2.52E+00	4.77E+00

Fonte: a autora

Tabela 17 – Resultado das populações finais utilizando a função Schwefel (200 indivíduos)

Função 3: Schwefel		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	4.19E+03	1.47E+03
2	4.19E+03	1.68E+03
3	4.18E+03	1.55E+03
4	4.19E+03	1.66E+03
5	4.18E+03	2.01E+03
6	4.18E+03	2.28E+03
7	4.19E+03	1.75E+03
8	4.19E+03	1.52E+03
9	4.18E+03	2.01E+03
10	4.19E+03	2.06E+03
11	4.19E+03	1.90E+03
12	4.19E+03	1.80E+03
13	4.19E+03	2.05E+03
14	4.19E+03	1.67E+03
15	4.19E+03	1.91E+03
16	4.18E+03	1.72E+03
17	4.18E+03	1.47E+03
18	4.19E+03	1.47E+03
19	4.19E+03	1.07E+03
20	4.19E+03	1.44E+03
21	4.19E+03	1.82E+03
22	4.18E+03	1.50E+03
23	4.19E+03	1.77E+03
24	4.19E+03	1.59E+03
25	4.18E+03	2.34E+03
Menor valor:	4.18E+03	1.07E+03
Média:	4.19E+03	1.74E+03
Maior valor:	4.19E+03	2.34E+03
Mediana:	4.19E+03	1.72E+03

Fonte: a autora

Tabela 18 – Resultado das populações finais utilizando a função Dixon-Price (200 indivíduos)

Função 4: Dixon-Price		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	1.36E+00	7.02E-01
2	8.11E-01	8.31E-01
3	1.16E+00	7.02E-01
4	7.85E-01	6.76E-01
5	1.93E+00	6.79E-01
6	7.54E-01	6.93E-01
7	8.21E-01	7.17E-01
8	1.75E+00	6.73E-01
9	7.22E-01	1.02E+00
10	1.19E+00	7.02E-01
11	1.24E+00	6.97E-01
12	8.16E-01	7.34E-01
13	1.94E+00	7.78E-01
14	8.02E-01	7.20E-01
15	1.22E+00	9.17E-01
16	8.04E-01	6.72E-01
17	8.43E-01	8.48E-01
18	8.88E-01	6.89E-01
19	9.77E-01	8.62E-01
20	9.55E-01	6.73E-01
21	7.49E-01	9.55E-01
22	9.21E-01	7.36E-01
23	8.77E-01	6.88E-01
24	1.16E+00	6.72E-01
25	7.52E-01	6.75E-01
Menor valor:	7.22E-01	6.72E-01
Média:	1.05E+00	7.48E-01
Maior valor:	1.94E+00	1.02E+00
Mediana:	8.88E-01	7.02E-01

Fonte: a autora

Tabela 19 – Resultado das populações finais utilizando a função Rosenbrock (200 indivíduos)

Função 5: Rosenbrock		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	1.38E+01	9.57E+00
2	7.89E+00	8.83E+00
3	1.09E+01	8.90E+00
4	1.08E+01	8.64E+00
5	1.00E+01	8.66E+00
6	1.16E+01	8.96E+00
7	1.24E+01	8.71E+00
8	1.06E+01	9.58E+00
9	9.17E+00	8.68E+00
10	9.75E+00	9.24E+00
11	1.05E+01	8.94E+00
12	8.87E+00	9.03E+00
13	1.20E+01	8.81E+00
14	1.09E+01	8.40E+00
15	9.85E+00	8.71E+00
16	8.89E+00	8.75E+00
17	1.04E+01	9.08E+00
18	9.50E+00	8.85E+00
19	9.37E+00	9.04E+00
20	9.03E+00	8.92E+00
21	7.48E+00	8.94E+00
22	1.51E+01	9.03E+00
23	1.11E+01	9.00E+00
24	9.01E+00	8.97E+00
25	1.03E+01	9.13E+00
Menor valor:	7.48E+00	8.40E+00
Média:	1.04E+01	8.93E+00
Maior valor:	1.51E+01	9.58E+00
Mediana:	1.03E+01	8.94E+00

Fonte: a autora

Tabela 20 – Resultado das populações finais utilizando a função Levy (200 indivíduos)

Função 6: Levy		
Menor valor do *fitness* da população		
Iteração	Pop Final: SGA+E	Pop Final: FGA
1	2.61E-02	5.66E-04
2	1.93E-02	2.21E-03
3	2.99E-02	1.12E-03
4	2.52E-02	1.06E-03
5	2.00E-02	8.84E-04
6	1.46E-02	2.04E-03
7	2.59E-02	1.96E-03
8	3.53E-02	1.72E-03
9	2.39E-02	1.55E-03
10	2.25E-02	1.97E-03
11	3.39E-02	1.43E-03
12	2.75E-02	1.21E-03
13	2.52E-02	2.04E-03
14	2.35E-02	3.58E-04
15	1.42E-02	1.85E-03
16	1.83E-02	1.78E-03
17	2.61E-02	8.74E-04
18	1.93E-02	1.46E-03
19	3.46E-02	5.05E-04
20	1.71E-02	6.54E-04
21	2.44E-02	1.68E-03
22	1.68E-02	9.11E-04
23	1.83E-02	1.95E-03
24	2.79E-02	4.07E-04
25	2.52E-02	2.05E-03
Menor valor:	1.42E-02	3.58E-04
Média:	2.38E-02	1.37E-03
Maior valor:	3.53E-02	2.21E-03
Mediana:	2.44E-02	1.46E-03

Fonte: a autora