CURSO DE SISTEMAS DE INFORMAÇÃO

ANÁLISE DE ALGORITMOS EVOLUCIONÁRIOS NA RESOLUÇÃO DE FUNÇÕES DE BENCHMARK IRRESTRITAS

RICK CAMELO MARTINS

Palmas – TO

2017

CURSO DE SISTEMAS DE INFORMAÇÃO

ANÁLISE DE ALGORITMOS EVOLUCIONÁRIOS NA RESOLUÇÃO DE FUNÇÕES DE BENCHMARK IRRESTRITAS

RICK CAMELO MARTINS

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins – UNITINS, com orientação do professor Me. Douglas Chagas, como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Palmas, junho de 2017.

CURSO DE SISTEMAS DE INFORMAÇÃO

ANÁLISE DE ALGORITMOS EVOLUCIONÁRIOS NA RESOLUÇÃO DE FUNÇÕES DE BENCHMARK IRRESTRITAS

RICK CAMELO MARTINS

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins – UNITINS, com orientação do professor Me. Douglas Chagas, como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____

Me. Douglas Chagas da Silva

_____

Me. Paulo Vitoriano Dantas Pereira

_____

Me. Marco Antônio Firmino de Sousa

DEDICATÓRIA

“Dedico este trabalho ao meu pai Valquires,
minha mãe Rosineide e aos meus irmãos.”

AGRADECIMENTOS

Agradeço a Deus por ter chegado aonde cheguei, por ter me dado força quando parecia não existir dentro de mim.

Agradeço aos meus pais pela confiança, a minha família por sempre estarem ao meu lado.

Agradeço a minha namorada, pelas palavras de incentivo e pela paciência quando não pude estar presente.

Agradeço aos meus amigos Denis Sousa, Matheus Germano e Gabriel Ferreira por toda a ajuda e por terem me acompanhado durante toda essa caminhada.

Aos amigos da vida, pessoas maravilhosas que conheci e que nem sempre nos vemos, mas que sempre me deram total apoio quando precisei.

Agradeço também aos professores da minha formação acadêmica, em especial ao meu professor orientador Me. Douglas Chagas por todo o conhecimento que me repassou, pela disponibilidade e profissionalismo que fez com que esse trabalho se tornasse possível.

Meus sinceros agradecimentos a todos!

Ama-se mais o que se conquista com esforço. (Benjamin Disraeli)

RESUMO

Os Algoritmos Evolucionários (AE), baseados em evolução natural, têm sido utilizados por diversas áreas de conhecimento principalmente para resolução de problemas de otimização na qual algoritmos tradicionais determinísticos se tornavam onerosos ou incapazes de resolver. Desde sua concepção, a ideia de AE gerou diversos tipos de abordagens como os Algoritmos Genéticos (GA), as Estratégias Evolutivas (EE), Evolução Diferencial (DE), dentre outros. O intuito deste trabalho é avaliar o desempenho dos algoritmos evolucionários Algoritmo Genético Simples (SGA) e DE na resolução de funções de *benchmark* irrestritas. Devido às características distintas das funções, será possível inferir, de acordo com o comportamento dos algoritmos e análise estatística dos resultados, qual se destaca para variados tipos de problemas.

Palavras-chave: Algoritmo Genético; Computação Evolutiva; Evolução Diferencial.

ABSTRACT

The Evolutionary Algorithms (EA), based on natural evolution, have been used by many areas of knowledge mostly in optimization problems solving in which traditional deterministic algorithm would become expensive or unable to resolve. Since it's conception, the idea of EA produced several types of approaches like Genetic Algorithm (GA), Evolutionary Strategies (EE), Differential Evolution (DE), among others. The intention of this paper is to evaluate the performance of the evolutionary algorithms Simple Genetic Algorithm (SGA) and DE in the resolution of unrestricted benchmark functions. In reason of the functions' different characteristics, it will be possible to infer, based on the algorithms' behavior and the statistical analysis of the results, which stands out for various types of problems.

Keywords: Evolutionary Computation; Differential Evolution; Genetic Algorithm.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A aplicação de algoritmos evolucionários na resolução de problemas de grande complexidade tem cada vez mais se firmado nos últimos anos. O comportamento desses algoritmos possibilita entre outros, prover mecanismos que facilitam sua utilização em diversas áreas do conhecimento. A simplicidade conceitual e a flexibilidade que os mesmos possuem, também são considerados fatores bastante relevantes e que justifica suas abordagens na resolução de diversos problemas.

As técnicas computacionais apresentadas por essa família de algoritmos são baseadas nos processos naturais de evolução biológica, fundamentados principalmente nos trabalhos de Darwin e Mendel, desenvolvidos em sua grande parte na primeira metade do século XIX. Dessa maneira, os algoritmos evolucionários têm sido objetos de pesquisa e, constituem um conjunto de métodos extremamente eficientes na resolução de problemas de otimização (Costa, 2003; Oliveira, 2006; Da Costa et al., 1999).

Este trabalho está organizado da seguinte forma: no capitulo 2 é apresentada uma breve explicação sobre o funcionamento dos algoritmos e os problemas de otimização nos quais são aplicados. No capítulo 3 têm-se a definição da configuração dos métodos desenvolvidos e as particularidades de cada algoritmo. No capítulo 4 são discutidos, e por fim, têm-se as considerações finais no capítulo 5.

1.1 Motivação

Os algoritmos determinísticos foram os primeiros a serem estudados pela comunidade científica com o intuito de resolver problemas de otimização, porém logo descobriu-se sua limitação para a resolução de problemas com características descontínuas e multimodais, o que motivou o direcionamento das pesquisas para métodos estocásticos (Costa, 2003; Parreiras, 2006).

A Otimização Evolucionária utiliza de métodos estocásticos de busca mono e multiobjetivo baseados em evolução natural, apresentando como principais vantagens o fato de conseguir encontrar ótimos globais em funções com grau elevado de complexidade e não necessitar de cálculo de derivadas (Parreiras, 2006).

Os Algoritmos Genéticos (GA) foram desenvolvidos por John Holland (1975), constituem métodos de busca semi-aleatória baseada na teoria da evolução das espécies de Charles Darwin, onde alguns princípios básicos propostos por Darwin agem sobre uma população de possíveis soluções a fim de realizar uma competição de sobrevivência entre elas, garantindo que sempre seja criadas soluções melhores para o problema (Oliveira, 2006).

O algoritmo Evolução Diferencial (DE), por sua vez, foi proposto por Storn e Price (1995) com o intuito de buscar melhores resultados com abordagem diferente das utilizadas nos GAs (Oliveira, 2006).

Busca-se neste trabalho avaliar os algoritmos evolucionários Algoritmo Genético Simples e Evolução Diferencial, permitindo inferir quais apresentam melhores desempenho para um conjunto de funções de *benchmark* irrestritas.

1.2 Objetivos

1.2.1 Objetivo Geral

- Comparar o desempenho entre os algoritmos evolucionários Evolução Diferencial (DE) e Algoritmo Genético Simples (SGA) para a resolução de funções de *benchmark* irrestritas.

1.2.2 Objetivos Específicos

- Estudar e fundamentar os conceitos envolvendo computação evolucionária e os métodos Algoritmo Genético e Evolução Diferencial;
- Analisar as funções a serem executadas pelos algoritmos;
- Implementar os métodos e ajustar os parâmetros dos algoritmos estudados;
- Verificar o desempenho dos algoritmos desenvolvidos para o conjunto de funções de testes;
- Analisar e discutir os resultados dos testes.

2 REFERENCIAL TEÓRICO

2.1 Computação Evolucionária

Charles Darwin, naturalista inglês nascido em 1809, acreditava que todas as espécies sofriam alterações ao passar do tempo, porém não tinha capacidade de definir como isso ocorria (Marques, 2017). Os estudos de Jean-Baptiste Lamarck que afirmavam, dentre outras coisas, que a evolução das espécies se dava ao longo do tempo onde os órgãos mais utilizados tornavam-se mais desenvolvidos enquanto os órgãos menos utilizados sofriam menos alterações, serviu para Darwin dar um pontapé de partida em sua teoria. Um exemplo utilizado por Lamarck para defender suas teorias era o exemplo do pescoço das girafas, as quais, segundo ele, ao tentar se alimentar das folhas que ficavam no alto das árvores, começaram a esticar seus pescoços, ficando esses cada vez maiores à medida que as novas gerações iam necessitando desta característica (Marques, 2017).

Analisando os estudos de Lamarck, Darwin propôs que as girafas não desenvolviam seus pescoços pela necessidade de alcançar as folhas nas partes mais altas das árvores, mas sim porque somente as girafas com pescoço mais comprido tinham capacidade de sobreviver e então essa característica era passada para as próximas gerações, sendo esse processo denominado por ele como Seleção Natural (Marques, 2017).

A partir destas análises que Darwin lançou sua principal obra: Sobre a Origem das Espécies por meio da Seleção Natural (Darwin, 1859), onde ele formula toda a teoria da evolução das espécies através da seleção natural onde os indivíduos com características mais úteis para a sobrevivência propagavam-nas para as próximas gerações.

A Computação Evolucionária (CE) é um termo muito utilizado na Tecnologia da Informação para se referir as técnicas computacionais que se baseiam na teoria da evolução natural de Charles Darwin (Costa, 2003; Gaspar-Cunha, Takahashi, Antunes, 2002). A ideia por trás do desenvolvimento de tais técnicas era aplicar o modelo evolucionário para a resolução de problemas, dentre eles destacam-se os problemas de otimização, os quais receberam as principais contribuições da CE (Von Zuben, 2000).

Barbosa (2005) comenta sobre a CE:

> É considerada um ramo da ciência da computação que se fundamenta em um novo paradigma para a resolução de problemas, não exigindo o conhecimento de uma sistemática prévia de resolução e que se baseia nos mecanismos encontrados na natureza, à luz da teoria da evolução natural de Darwin. (BARBOZA, 2005, p. 26)

"Algoritmos evolucionários usam modelos computacionais dos processos naturais de evolução como uma ferramenta para resolver problemas." (LINDEN, 2008, p. 40).

"Dá-se o nome de Computação Evolutiva à investigação de modelos computacionais que de alguma forma se inspiram no processo de evolução natural, em que indivíduos são submetidos à variação genética, seleção e adaptação." (DA COSTA et al., 1999, p. 36).

Segundo Von Zuben (2000), a principal vantagem da computação evolutiva se dá pela capacidade de resolver problemas apenas descrevendo-os matematicamente, sem que seja necessário especificar uma sequência de passos para a resolução do mesmo, uma vez que os Algoritmos Evolucionários (AEs) seguem uma sequência utilizada por uma grande quantidade de problemas, tornando-os robustos e flexíveis.

> A computação evolutiva deve ser entendida como um conjunto de técnicas e procedimentos genéricos e adaptáveis, a serem aplicados na solução de problemas complexos, para os quais outras técnicas conhecidas são ineficazes ou nem sequer são aplicáveis. (VON ZUBEN, 2000, p. 2)

Seguindo o pensamento de Zuben, a computação evolutiva não deve ser vista como uma ferramenta "pronta para o uso", mas como um conjunto de processos que devem ser adaptados ao contexto ao qual necessita-se de sua aplicação.

Embora haja uma relativa diversidade de AEs, em geral, os algoritmos genéticos são compostos por elementos base em comum, segundo Da Costa et al. (1999) e Costa (2003), são eles:

- População de indivíduos candidatos a potenciais soluções (População inicial);
- Seleção, mecanismo que realize a medição da adequação de cada indivíduo em relação aos outros;
- Mecanismo de controle de diversidade, geralmente representados pelos operadores genéticos.

São denominados indivíduos todas as potenciais soluções para o problema. População é a denominação de um conjunto de indivíduos. Nos AEs, os indivíduos (ou cromossomos) são representados por uma lista ordenada ou por uma árvore de atributos, descritos através de um alfabeto finito (Von Zuben, 2000). Tanto na lista como na árvore, cada atributo corresponde à um gene. A quantidade de genes utilizados tanto na lista quanto na árvore está relativa à representação do problema em uma solução-candidata (indivíduo). A árvore de atributos é utilizadas quando não é possível representar um indivíduo através de uma lista ordenada, que geralmente têm tamanho único, tornando seu tamanho variável.

"A *seleção* faz com que os indivíduos com melhor desempenho (em termos do problema que está a ser resolvido) tendam a reproduzir-se e, consequentemente, tendam a manter-se presentes na população" (Costa, 2003, p. 21, grifo do autor). Cada indivíduo da população é submetido ao processo de análise de adequação (seleção), na qual será analisado o grau de adequação do mesmo em relação à solução ótima do problema. Em outras palavras, os indivíduos que possuem as características mais próximas da solução ótima serão utilizados para desenvolver a próxima geração de indivíduos, desta forma, os indivíduos das próximas gerações terão características cada vez mais próximas da solução desejada.

Segundo Gaspar-Cunha, Takahashi e Antunes (2002, p. 69), "Esquemas de seleção - definem como os indivíduos mais bem adaptados são favorecidos na disputa pela procriação".

"Os operadores genéticos proporcionam a diversidade na população necessária à procura eficiente de soluções para o problema" (COSTA, 2003, p. 21). Eles são responsáveis por alterar as características dos indivíduos, aumentando o espaço de procura, evitando vícios e convergências prematuras à solução, gerando novas soluções-candidatas.

Von Zuben (2000) aponta a forma de utilização dos operadores genéticos como a principal diferença entre os AE:

> Os sistemas baseados em computação evolutiva mantêm uma população de soluções potenciais, aplicam processos de seleção baseados na adaptação de um indivíduo e também empregam outros operadores "genéticos". Diversas abordagens para sistemas baseados em evolução foram propostas, sendo que as principais diferenças entre elas dizem respeito aos operadores genéticos empregados. (VON ZUBEN, 2000, p. 9)

Gaspar-Cunha, Takahashi e Antunes (2002, p. 75) comentam sobre a atuação dos operadores genéticos, "é através desse processo de experimentação de recombinações e/ou variações sobre padrões promissores que o espaço de busca é explorado".

"Os operadores genéticos consistem em aproximações computacionais de fenômenos vistos na natureza, como a reprodução sexuada, a mutação genética e quaisquer outros que a imaginação dos programadores consiga reproduzir." (LINDEN, 2008, p. 41).

Em geral, os dois principais tipos de operadores genéticos considerados são a mutação e o cruzamento (ou recombinação). No cruzamento, são selecionados pares de cromossomos os quais serão utilizados para gerar novos indivíduos a partir da troca de atributos entre os mesmos. A mutação tem a função de inserir novas características aos indivíduos para aumentar o espaço de procura.

A procura por soluções nos AEs se dá pela utilização de iteração, que em analogia ao processo evolutivo, é chamada de geração. Para cada geração, são verificados os critérios de parada do algoritmo, não satisfeitos os critérios de parada, a população de indivíduos é submetida ao processo de seleção e à atuação de operadores genéticos.

Desde a sua concepção, surgiram vários AEs, dentre elas podemos destacar as Estratégias Evolutivas (EEs), propostas por Rechenberg (1973), os Algoritmos Genéticos (GAs), proposto por Holland (1975) e a Programação Evolutiva (PE) proposta por Fogel (1966).

2.2 Algoritmos Genéticos

Os Algoritmos Genéticos (GAs) são algoritmos inspirados no processo de seleção natural e na genética propostos por Holland (1975) para a resolução de problemas, principalmente os ligados à otimização. Tratando-se de um tipo de algoritmo evolucionário, os GAs baseiam-se nos princípios da teoria da evolução propostos por Charles Darwin, que propõe, entre outras coisas, que os indivíduos mais aptos têm maiores chances de sobrevivência no ambiente, garantindo a evolução das espécies e a manutenção de suas características.

“Algoritmos genéticos (GA) são um ramo dos algoritmos evolucionários e como tal podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural.” (LINDEN, 2008, p. 43).

“Algoritmos Genéticos têm se mostrado muito eficientes para busca de soluções ótimas, ou aproximadamente ótimas em uma grande variedade de problemas, pois não impões muitas das limitações encontradas nos métodos de busca tradicionais.” (REZENDE, 2003, p. 228).

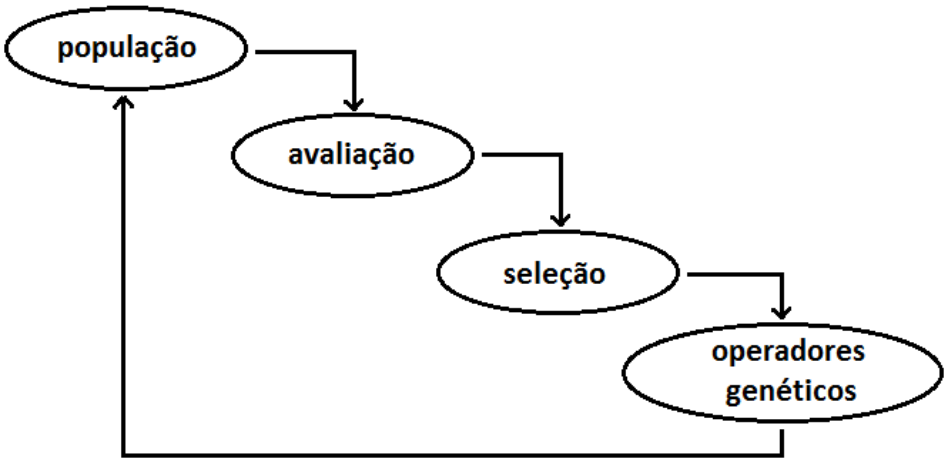

Figura 1. Processos básicos de um Algoritmo Genético.
(Adaptado de Costa, 2003.)

O funcionamento de um GA é composto por quatro componentes básicos (Costa, 2003), como ilustra a Figura 1:

- População de indivíduos, onde cada indivíduo representa uma possível solução para o problema;
- Avaliação, etapa na qual avalia-se a aptidão de cada indivíduo ;
- Seleção, os melhores indivíduos tendem a ser selecionados como progenitores (pais) para a geração de novos indivíduos (descendentes);
- Operadores genéticos, os indivíduos descendentes são gerados através do processo de recombinação, onde os descendentes herdarão algumas características dos indivíduos progenitores. Além da recombinação, os indivíduos descendentes são submetidos ao processo de mutação para possibilitar o aparecimento de novas características.

Após a ação dos operadores genéticos, os indivíduos descendentes juntam-se aos indivíduos progenitores para formar uma nova população de indivíduos, repetindo-se os

processos por uma determinada quantidade de gerações ou até que se satisfaça os critérios de parada.

Como descrito anteriormente, os GAs são processados de acordo com o algoritmo abaixo (Costa, 2003):

1. INICIAÇÃO

   A população inicial de indivíduos é criada aleatoriamente.
2. AVALIAÇÃO

   Cada indivíduo é na população avaliado de acordo com uma medida do desempenho.
3. SELEÇÃO

   Um conjunto de indivíduos é selecionado para gerar descendentes de tal forma que os que tiverem melhor desempenho têm maior probabilidade de serem selecionados.
4. OPERADORES GENÉTICOS

   Ao conjunto de indivíduos selecionados são aplicados operadores genéticos como a recombinação e a mutação. SE não se verificar o critério de parada ENTÃO voltar ao passo 2. SENÃO terminar.

2.2.1 Codificação

Nos GAs, os indivíduos são denominados cromossomos em analogia ao sistema genético. Cada gene do cromossomo podem ser representados por diversos valores, chamados de alelos. Define-se locus do gene o local do cromossomo onde localiza-se o gene (Costa, 2003).

> A decodificação é responsável por traduzir as informações contidas nos cromossomos em soluções do problema, sobre as quais calcula-se o valor da função de adequação e se verifica a existência de possíveis infactibilidades por violação de restrições. (DA COSTA, 1999, p. 52)

"O cromossomo é uma estrutura de dados, geralmente vetores ou cadeias de valores binários (estrutura mais tradicional, porém nem sempre a mais indicada), que representa uma possível solução do problema a ser otimizado" (REZENDE, 2003, p. 231).

A Figura 2 ilustra a representação de um cromossomo e gene.

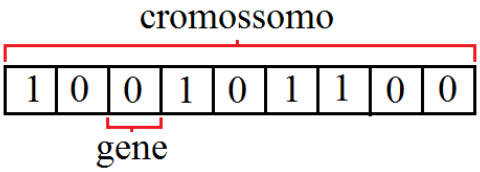

Figura 2. Representação de cromossomo e gene em GA.
(Adaptado de Parreiras, 2006)

Nos GAs tradicionais, os cromossomos são representados por sequências de dígitos binários (Costa, 2003). Entretanto, Von Zuben (2000) adverte que em diversas aplicações práticas a utilização deste tipo de codificação gera desempenho insatisfatório e que em caso de aplicação em problemas de otimização numérica com parâmetros reais, algoritmos representados por números inteiros ou ponto flutuante obtêm resultados mais satisfatórios.

> Tendo em vista que os AG's trabalham com manipulação de strings de determinados alfabetos (representação), deve-se especificar a codificação com a qual se faz corresponder cada ponto do domínio do problema com um Gene ou conjunto de Genes do Cromossomo. (Guimarães e Ramalho, 2001)

"Em geral, o cromossomo representa o conjunto de parâmetros da função-objetivo cuja resposta será maximizada ou minimizada." (REZENDE, 2003, p. 231).

"A definição inadequada da codificação pode levar a problemas de convergência prematura do algoritmo genético. A estrutura de um cromossomo deve representar uma solução como um todo, e deve ser a mais simples possível." (VON ZUBEN, 2003, p.12).

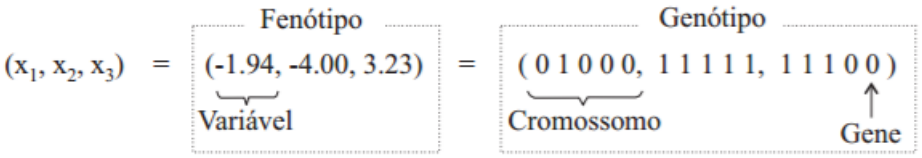

Figura 3. Fenótipo e Genótipo em GA.
(Parreiras, 2006)

Na biologia, denomina-se genótipo a totalidade do pacote genético do indivíduo, em GAs, o genótipo é equivalente à estrutura. A interação do genótipo com o ambiente é, biologicamente, denominada fenótipo, em GAs, fenótipos são as soluções após a decodificação da estrutura (genótipo) (Costa, 2003). A figura 3 ilustra a diferença entre fenótipo e genótipo.

2.2.2 Seleção

"A seleção determina quais os indivíduos da população que são escolhidos para progenitores. Aos indivíduos progenitores aplicam-se operadores genéticos gerando novos indivíduos." (Costa, 2003). O autor comenta ainda que a seleção em GAs é geralmente probabilística, onde os indivíduos que têm maior desempenho têm maior chance de serem selecionados.

> A seleção é o instrumento pelo qual os algoritmos evolucionários conduzem a busca para as regiões mais promissoras do espaço e, que, efetivamente, permite o processo de otimização. Ela simula o efeito da seleção natural, sendo responsável por escolher os indivíduos da população que deixarão descendentes para a geração seguintes, ou seja, aqueles que participarão das operações genéticas. (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2002, p. 75)

Os métodos de seleção mais utilizados são descritos a seguir:

2.2.2.1 Seleção por Roleta

A seleção por roleta é um método probabilístico onde simula-se uma roleta, cada indivíduo representa uma fatia da roleta proporcional ao seu desempenho; posteriormente, é escolhido um número aleatório que irá definir o indivíduo escolhido com base na roleta, o indivíduo no qual a fatia englobe o número escolhido é o selecionado. O processo se repete até que seja selecionada a quantidade desejada de indivíduos (Costa, 2003; Rezende, 2003).

2.2.2.2 Seleção por torneio

Neste método de seleção, um número x de indivíduos, definido previamente, é escolhido aleatoriamente na população, dentre os indivíduos escolhidos, é selecionado o que tiver o melhor desempenho entre eles. O processo se repete até que seja selecionada a quantidade desejada de indivíduos (Costa, 2003; Rezende, 2003).

2.2.2.3 Seleção por Truncagem

Neste método de seleção, é realizado o ordenamento dos indivíduos com base em sua medida de desempenho, após o ordenamento, somente os melhores indivíduos são selecionados para progenitores. A quantidade de indivíduos que serão selecionados deve ser definida (Costa, 2003; Linden, 2008).

2.2.3 Recombinação (Cruzamento)

A recombinação, também chamada de cruzamento ou *crossover*, é um operador genético que tem como finalidade a geração de novos indivíduos através da combinação entre outros indivíduos. A geração dos novos indivíduos se dá pela junção de atributos genéticos dos indivíduos progenitores (Oliveira, 2006).

"O operador de cruzamento propõe-se a recombinar partes de estruturas previamente selecionadas pelo processo de seleção, na esperança de se derivar estruturas ainda melhores." (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2002, p. 76).

Os novos indivíduos terão características de seus progenitores, porém serão diferentes dos mesmos. O cruzamento mais simples tem somente um ponto de corte o qual é definido aleatoriamente. Há diversos tipos de operadores de recombinação na literatura, entre eles, se destacam o *crossover* de dois pontos, que como o próprio nome sugere, trabalha com dois pontos de corte para a troca de material genético, além do *crossover uniforme* (SYSWERDA, 1989) em que cada *locus* do primeiro descendente é preenchido por um bit de um dos progenitores, a definição do progenitor contribuir para o *locus* é definido através de alguma probabilidade constante (Von Zuben, 2000).

"Uma conclusão a que se pode chegar é que cada operador de *crossover* é particularmente eficiente para uma determinada classe de problemas e extremamente ineficiente para outras." (VON ZUBEN, 2000, p. 14).

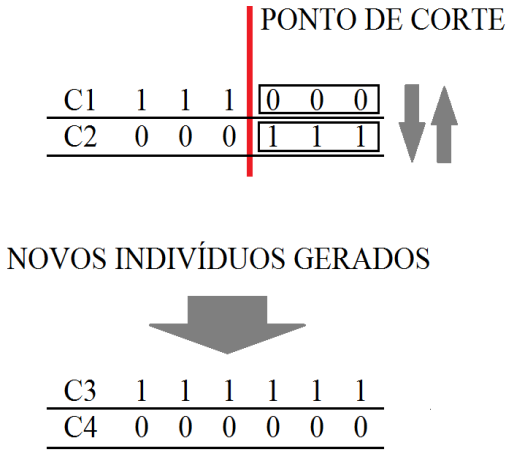

Figura 4. Exemplo de cruzamento em GA.
(Adaptado de Rezende, 2003)

Na figura 4 é exposto um exemplo de cruzamento com um ponto de corte. Foram selecionados dois indivíduos aleatoriamente, C1 e C2. Definiu-se o ponto de corte, no terceiro gene. Os dois indivíduos gerados, C3 e C4, preservaram os primeiros atributos de um dos indivíduos progenitores, o C3 do C1 e o C4 do C2. O restante dos atributos foram preenchidos pelos últimos três atributos do outro indivíduo progenitor.

2.2.3.1 Taxa de cruzamento

A taxa de cruzamento é utilizada para controlar o alcance do operador de cruzamento sobre a população. Quanto mais alta for a taxa de cruzamento, mais rápida será a introdução de novas estruturas à população, porém deve-se ter o cuidado ao aumentar esta taxa para que não se perca estruturas de alta aptidão (Guimarães e Ramalho, 2001).

2.2.4 Mutação

O operador de mutação é utilizado para inserir novas características aos cromossomos. "A mutação é um operador unário que consiste em perturbar ligeiramente (com uma probabilidade pequena) os indivíduos descendentes gerados pela recombinação." (COSTA, 2003, p. 56). Segundo Costa (2003), a mutação também pode garantir que um determinado caractere genético não desapareça na população.

> A mutação deriva um novo descendente mediante modificações na estrutura de um indivíduo selecionado. Estas modificações são normalmente de pequena escala ("perturbações"), e visam tanto a exploração de regiões vizinhas àquele indivíduo quanto a (re)introdução de material genético a fim de preservar a diversidade da população. (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2002, p. 76).

Segundo Rezende (2003), este operador é necessário para introduzir e manter a diversidade genética da população.

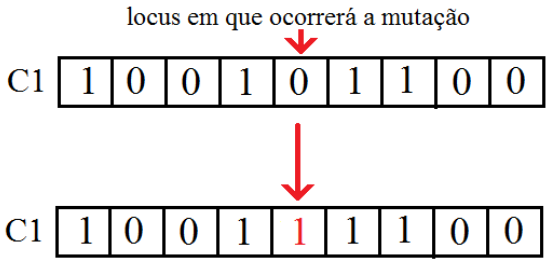

Figura 5. Exemplo de mutação binária em GA.
(Adaptado de Costa, 2003)

A Figura 5 ilustra o funcionamento de uma mutação binária simples em um cromossomo em GA. O cromossomo C1 em seu locus 4 sofre alteração, mudando seu atributo de 0 para 1.

2.2.4.1 Taxa de Mutação

A taxa de mutação é utilizada para determinar a probabilidade com a qual a mutação ocorrerá. Uma taxa de mutação baixa garante que um determinado locus não fique preso a um valor, garantindo assim que se possa chegar a qualquer lugar no espaço de busca, por outro lado, um valor muito alto pode tornar a busca essencialmente aleatória, além de aumentar a probabilidade de uma boa solução ser destruída (Guimarães e Ramalho, 2001).

2.2.5 Elitismo

O Elitismo (De Jong, 1975), tem como objetivo manter a melhor solução na geração seguinte já que esta pode ter sofrido com a ação dos operadores genéticos.

> "Elistismo (De Jong, 1975) consiste em manter, na geração seguinte, um determinado número de melhores soluções (ou soluções de elite). Estes indivíduos poderiam ser eliminados (embora pouco provável) no processo evolutivo, caso não fossem selecionados de maneira determinística, ou destruídos por cruzamento e mutação." (DA COSTA, 1999).

2.3 Evolução Diferencial

Evolução Diferencial (Differential Evolution - DE) é um algoritmo evolucionário utilizado para a resolução de problemas de otimização. Foi proposto por Storn e Price (1995) a fim de obter melhores resultados utilizando métodos um pouco diferentes dos até então encontrados em GAs. Essas diferenças são detalhadas nas seções subsequentes.

2.3.1 Estrutura da população

Segundo Storn e Price (1995), a implementação mais versátil de um DE se dá pela utilização de duas populações de vetores $Np\ D$ - dimensionais com valores reais como parâmetros. Expresso por Px a população atual é composta por vetores $x_{i.g.}$ que foram considerados aceitáveis como ponto inicial ou comparados à outros vetores:

$$\begin{aligned} &P_{x.g.} = \left(x_{i.g.}\right), \ \ i = 0,1,\dots,Np-1. \quad g = 0,1,\dots,g_{max.} \\ &x_{i.g.} = \left(x_{j.i.g.}\right), \qquad j = 0,1,\dots,D-1. \end{aligned} \tag{Eq.1}$$

Inicia-se os índices com 0 para simplificar o trabalho com arrays e aritmética modular. O índice g representa a geração a qual um determinado vetor pertence. O índice i representa a posição do vetor na população que vai de 0 a $Np-1$. Em cada vetor, o indexador utilizado para os parâmetros é j que vai de 0 a $D-1$. Quando iniciados, os DE escolhem aleatoriamente vetores da população inicial e realizam mutação para formar uma população intermediária, $P_{v.g.}$, de vetores Np modificados, $V_{i.g.}$ (Storn e Price, 1995; Oliveira, 2006; Guimarães, 2009):

$$\begin{aligned} P_{v.g.} &= \left(v_{i.g.}\right), \qquad i = 0,1,\dots,Np-1, \;\; g = 0,1,\dots,g_{max}, \\ v_{i.g.} &= \left(v_{j.i.g.}\right), \qquad i = 0,1,\dots,D-1. \end{aligned} \tag{Eq.2}$$

Cada vetor da população atual é recombinado com um vetor da população modificada e gera um novo vetor, chamado de vetor experimental, $U_{i.g.}$. A população de vetores experimentais é representada por P_u:

$$\begin{aligned} P_{u.g.} &= \left(u_{i.g.}\right), \qquad i = 0,1,\dots,Np-1, \qquad g = 0,1,\dots,g_{max}, \\ u_{i.g.} &= \left(u_{j.i.g.}\right), \qquad i = 0,1,\dots,D-1. \end{aligned} \tag{Eq.3}$$

Neste processo de recombinação, os vetores experimentais sobrescrevem a população modificada, deste modo, um único vetor conterá ambas as populações (Storn e Price, 1995; Oliveira, 2006; Souza, 2008).

2.3.2 Mutação

O processo de mutação consiste na escolha de três vetores (indivíduos) de maneira aleatória na população inicial (ou corrente), estes vetores serão utilizados para gerar um novo vetor obtido através da mutação entre eles. Devido à necessidade de três indivíduos para a realização da mutação, a população Np deve ser obrigatoriamente maior ou igual a quatro vetores. Para os vetores selecionados aleatoriamente, x_a, x_b e x_c, o vetor x_a sofrerá uma perturbação proporcional à diferença vetorial, também chamada de vetor diferença, calculada entre x_b e x_c e multiplicada pelo fator de mutação (ou perturbação) $\boldsymbol{Fp}$. "O fator de perturbação Fp é um número real, positivo pertencente ao intervalo [0,2] e controla a amplitude do vetor diferença." (Storn e Price, 1995, Oliveira, 2006; Guimarães, 2009).

$$x_{mut} = x_a + Fp\,(x_b - x_c) \tag{Eq.4}$$

A equação IV ilustra a maneira como ocorre a mutação em DE. Os índices a, b e c são valores entre 1 e Np, distintos entre si.

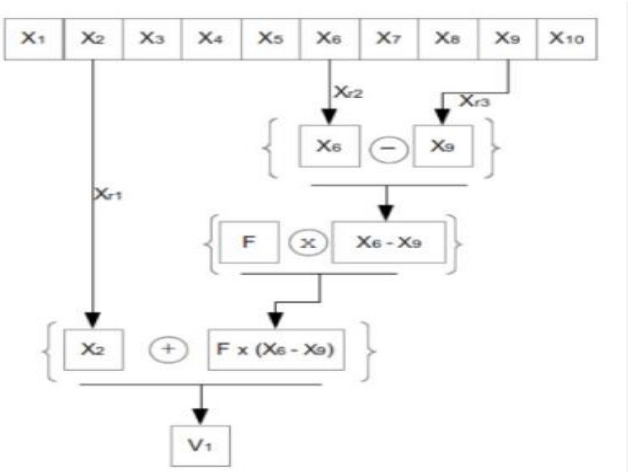

Figura 6. Exemplo de mutação em DE.
(Pacheco, 2017)

A Figura 6 ilustra o funcionamento do operador de mutação em DE. Foram selecionados aleatoriamente os vetores x_2, x_6 e x_9, o vetor x_2 sofrerá a perturbação resultante da multiplicação do fator de perturbação e o vetor diferencial obtido na diferença entre os vetores x_6 e x_9, resultando no vetor doador (Oliveira, 2006), v_1.

2.3.3 Recombinação

O operador de recombinação foi introduzido ao DE porque Storn e Price (1995) sentiram a necessidade de aumentar a diversidade da população. O funcionamento deste operador se dá pelo cruzamento entre vetores da população inicial (ou corrente, também chamada de alvo) e da população doadora. A recombinação é feita componente a componente, ou seja, para cada posição do vetor será validado qual gene que deverá ser transmitido, se o gene do vetor da população corrente ou da população doadora.

$$v_{rec_{ij}} = \begin{cases} v_{cor_{ij}}, se\ rand_i \leq P_c \\ v_{mut_{ij}}, se\ rand > P_c, i = 1, \dots, n \end{cases} \quad \text{(Eq.5)}$$

A equação V é utilizada na recombinação em DE, onde $v_{rec_{ij}}$ representa o componente do vetor experimental, $v_{cor_{ij}}$ é o componente do vetor corrente, $v_{mut_{ij}}$ é o componente do vetor doador. A variável $rand$ é um número gerado aleatoriamente no intervalo [0,1] (Oliveira, 2006; Souza, 2008). P_c é a taxa de cruzamento, ela define a probabilidade do vetor doador transmitir seus valores para o vetor experimental, essa variável é definida pelo usuário.

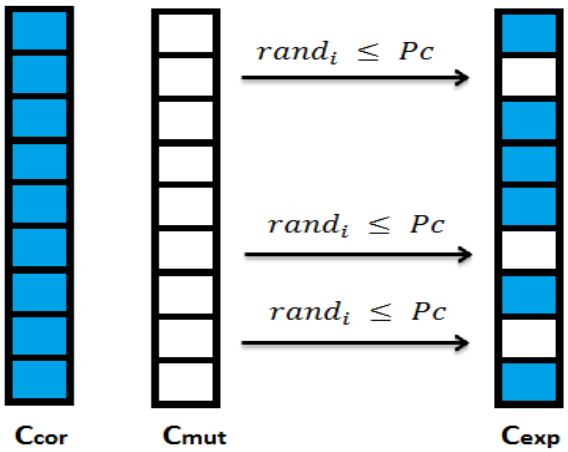

Figura 7. Ilustração de recombinação em DE.
(Adaptado de Oliveira, 2006)

A figura 7 ilustra o processo de recombinação em DE. Para cada locus dos vetores corrente e mutante (doador), é gerado um número aleatório $rand$ que será comparado à taxa de cruzamento para definir qual dos dois vetores irá contribuir para o vetor experimental. Para os casos em que o valor $rand$ for maior ou igual que P_c, o valor a ser atribuído ao vetor experimental será o do vetor corrente, caso seja menor, o locus do vetor doador irá contribuir para o vetor experimental, este tipo de recombinação é o tipo mais básico, conhecido na literatura como recombinação discreta.

2.3.4 Seleção

"A seleção é o processo de produzir filhos melhores. Diferentemente de outros algoritmos evolutivos, a evolução diferencial não usa hierarquia (elitismo) nem seleção proporcional." (OLIVEIRA, 2006, p 31). A estratégia utilizada para a seleção em DE é a do custo do vetor. Após a ação do operador de recombinação é calculada o custo do vetor experimental $Vexp$ e do vetor corrente (alvo) $Vcor$, se o custo do vetor corrente for maior ou igual ao custo do vetor experimental, então o vetor experimental será o vetor corrente da próxima geração. Caso contrário, mantém-se o vetor corrente da geração atual para a próxima geração.

O DE é finalizado até que se alcance o critério de parada ou o número máximo de gerações definido.

2.4 Problemas de otimização

O processo de otimização consiste em melhorar algo, onde têm-se um conceito inicial que sofre variações experimentais para melhorar as informações que o caracterizam (Oliveira ,2006).

"A Otimização é o campo de conhecimentos cujas técnicas visam determinar os extremos (máximos ou mínimos) de funções, em domínios determinados." (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2002, p. 1)

2.4.1 Otimização Mono-Objetivo

De forma geral, em problemas de otimização deseja-se a minimização ou maximização de uma $função\ objetivo$, também chamada de função de adaptação ou função custo, de n variáveis, chamadas de variáveis de projeto ou decisão, que podem ou não ser sujeitas a restrições do tipo igualdade, desigualdade e restrições laterais. O objetivo da utilização destas restrições é delimitar o espaço de busca onde deseja-se encontrar a melhor solução, chamada de $solução\ ótima$ (Oliveira, 2006).

> A otimização visa determinar a melhor configuração de projeto de um dado sistema sem ter que testar todas as possibilidades possíveis. Um problema de otimização pode apresentar várias soluções, assim, a definição de "melhor" solução é relativa ao problema considerado, ao método de solução e às tolerâncias adotadas. (OLIVEIRA, 2006, p. 1)

Costa (2003) define um problema de minimização genérico da seguinte forma:

$$\min f(x) \text{ onde } x \in \Omega \qquad \text{(Eq.6)}$$

Sujeito a

$$\begin{gathered} g_j(x) \geq 0 \text{ com } j = 1, \dots, m \\ h_i(x) = 0 \text{ com } i = m+1, \dots, m+p \end{gathered} \qquad \text{(Eq.7)}$$

Considerando:

- $\boldsymbol{x}$ é o vetor das variáveis de decisão, são variáveis que se modificam durante o processo de otimização, alterando o valor da função objetivo, sendo representadas por x_i, para $i = 1, \dots, n$ (Oliveira, 2006).
- $\boldsymbol{f(x)}$ é a função objetivo, função que será minimizada ou maximizada;
- $\boldsymbol{x*}$ é o ponto ótimo;

- Ω é o espaço das variáveis, todos os valores possíveis para as variáveis de decisão, também chamado de espaço viável ou espaço de busca.
- $\boldsymbol{g(x)}$ é o vetor de restrições do tipo $desigualdade$ que devem ser satisfeitas cujos valores variam de 1 a m;
- $\boldsymbol{h(x)}$ é o vetor de restrições do tipo $igualdade$ que devem ser satisfeitas cujos valores variam de 1 a $m + p$.

Oliveira (2006) complementa a definição de um problema de minimização os limites laterais:

$$x_i^{inf} \leq x_i \leq x_i^{sup}, \qquad i = 1, \dots, n \tag{Eq.8}$$

As restrições laterais limitam as variáveis de projeto, definidas como: limite inferior $x^{inf} = (x_1^{inf}, x_2^{inf}, \dots, x_n^{inf})^T$ e limite superior $x^{sup} = (x_1^{sup}, x_2^{sup}, \dots, x_n^{sup})^T$, onde T representa a transposição do vetor (Oliveira, 2006).

2.4.2 Otimização Local e Global

Seguem abaixo as definições de mínimo local e global segundo Costa (2003):

2.4.2.1 Mínimo Global

Um ponto $x*$ é mínimo global se $f(x*) \leq f(x)$ para todo o $x \in \Omega$. Por outro lado, muitas vezes, os algoritmos apenas permitem encontrar um mínimo local que é o ponto que na sua vizinhança apresenta o menor valor da função objetivo. Define-se uma vizinhança de x como sendo um conjunto aberto que contém x e que está contido no domínio de $f(x)$ (Costa, 2003).

2.4.2.2 Mínimo Local Fraco

Um ponto $x*$ é mínimo local fraco se existe uma vizinhança V de $x*$ tal que $f(x*) \leq f(x)$ para todo o $x \in V$ (Costa, 2003).

2.4.2.3 Mínimo Local Forte

Um ponto $x*$ é mínimo local forte se existe uma vizinhança V de $x*$ tal que $f(x*) < f(x)$ para todo o $x \in V$ e $x \neq x*$ (Costa, 2003).

2.5 Funções de testes para problemas de otimização

A qualidade de procedimentos de otimização é medida através de funções padrões. Existem vários classes destas funções, todas são contínuas, segue abaixo as definições de algumas classes de funções (Molga e Smutnicki, 2005):

a) Unimodal, convexo e multidimensional: contém funções que causam convergência pobre ou lenta para um mínimo global;
b) Multimodal, bidimensional com um pequeno número de mínimos locais: muito utilizadas para testar procedimentos de otimização em ambientes com poucos mínimos locais e com um único mínimo global;
c) Multimodal, bidimensional com grande número de mínimos locais;
d) Multimodal, multidimensional com grande número de mínimos locais.

As classes de funções *c* e *d* são recomendadas para testes de qualidade em métodos inteligentes de otimização como AEs e são consideradas funções muito difíceis (Molga e Smutnicki, 2005).

Nos tópicos a seguir serão apresentadas algumas funções comumente conhecidas na literatura.

2.5.1 Rosenbrock's valley

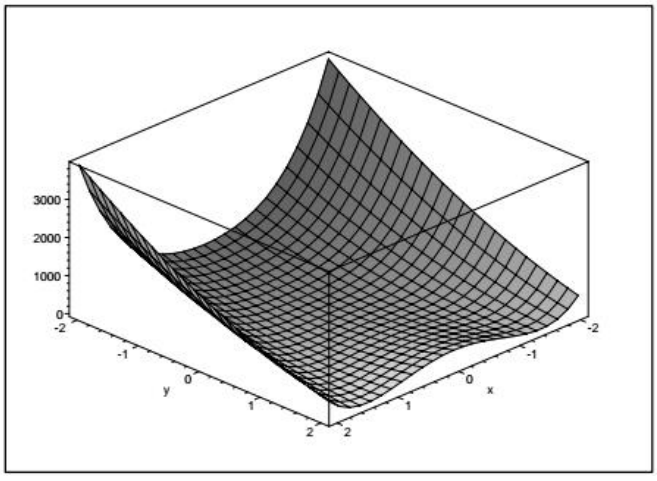

Figura 8. Rosenbrock's valley em 2D.
(Molga e Smutnicki, 2005)

Rosenbrock's valley é um problema de otimização clássico, conhecido também como função banana ou segunda função de De Jong. O vale é plano, longo, estreito e parabólico. O ótimo global localiza-se dentro deste plano, a tarefa de localizar o plano

não é relativamente fácil, porém a localização do ótimo global se torna muito difícil, por conta disso, este problema tem sido utilizado para testar algoritmos de otimização (Molga e Smutnicki, 2005). A função tem a seguinte definição:

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - {x_i}^2)^2 + (1 - x_i)^2]. \qquad \text{(Eq.9)}$$

O mínimo global igual a $f(x) = 0$ é obtido para $x_i = 0, i = 1, \dots, n$.

2.5.2 Ackley's function

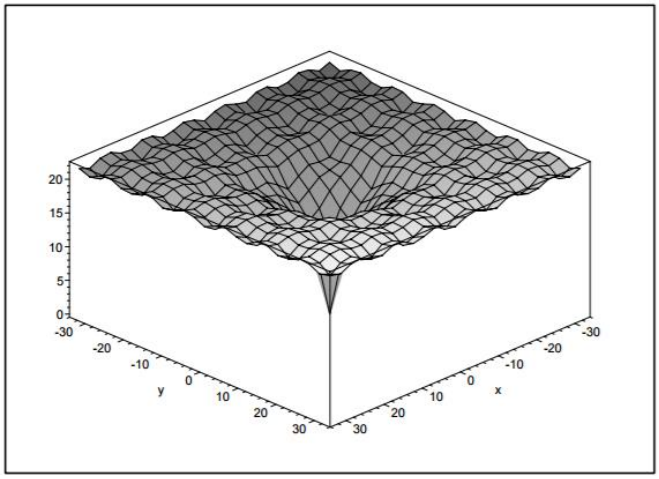

Figura 9. Ackley's function em 2D.
(Molga e Smutnicki, 2005)

É uma função multimodal de teste muito utilizada. A função tem a seguinte definição (Molga e Smutnicki, 2005):

$$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} \mathrm{x}^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)\right) + a + \exp(1) \qquad \text{(Eq.11)}$$

Molga e Smutnicki (2005) recomendam os seguintes valores como parâmetros: $a = 20, b = 0{,}2, c = 2\pi$. A área de teste é geralmente restrita ao hipercubo $-32{,}768 \leq x_i \leq 32{,}768, i = 1, \dots, n$. O mínimo global $f(x) = 0$ é obtido por $x_i = 0, i = 1, \dots, n$.

2.5.3 Drop Wave function

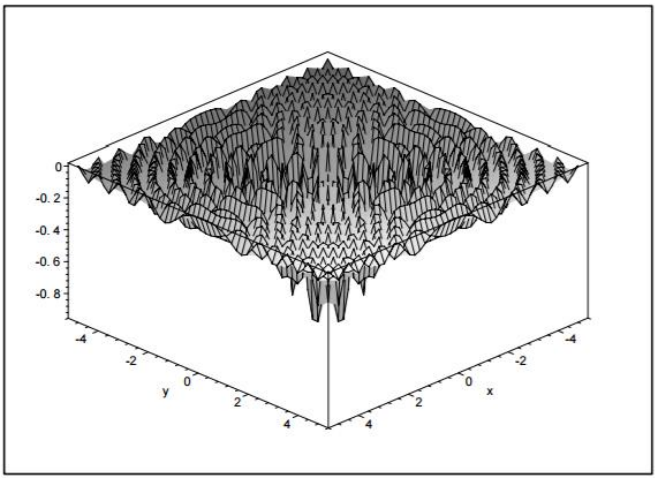

Figura 10. Drop Wave function em 2D.
(Molga e Smutnicki, 2005)

Função de teste multimodal a qual dispõe de somente duas variáveis com a seguinte definição:

$$f(x_1, x_2) = -\frac{1 + \cos(12\sqrt{x^2 + x_2^2})}{\frac{1}{2}(x_1^2 + x_2^2) + 2} \qquad \text{(Eq.12)}$$

A área de teste é geralmente restrita ao quadrado $-5{,}12 \leq x_1 \leq 5{,}12, -5{,}12 \leq x_2 \leq 5{,}12$ (Molga e Smutnicki, 2005).

2.5.4 Easom's function

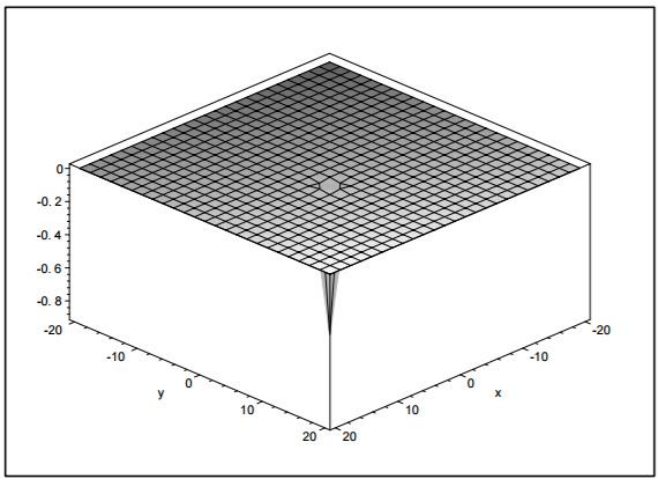

Figura 11. Easom's function em 2D.
(Molga e Smutnicki, 2005)

É uma função de teste unimodal, onde o mínimo global tem uma pequena área relativa ao espaço de procura. A função contém duas variáveis e tem a seguinte definição (Molga e Smutnicki, 2005):

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) \qquad \text{(Eq.13)}$$

O espaço de busca é frequentemente restrito ao quadrado $-100 \leq x_1 \leq 100, -100 \leq x_2 \leq 100$. O mínimo global $f(x) = -1$ é obtido por $(x_1, x_2) = (\pi, \pi)$. (Molga e Smutnicki, 2005).

2.5.5 De Jong's function

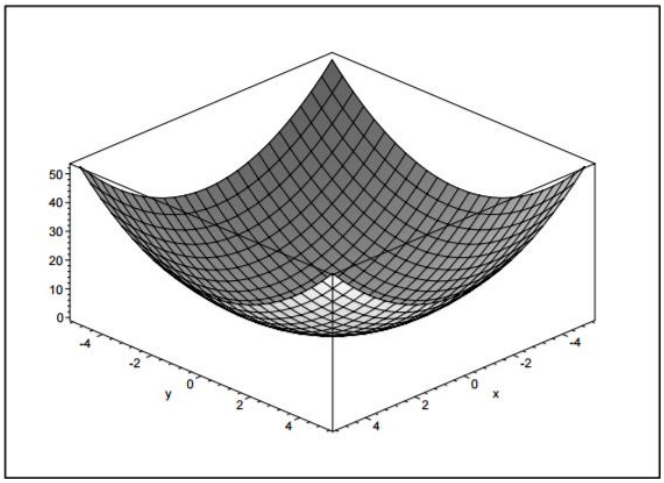

Figura 12. De Jong's function em 2D.
(Molga e Smutnicki, 2005)

Também chamada de primeira função de De Jong's é uma das funções de benchmark mais simples, é uma função contínua, convexa e unimodal e tem a seguinte definição (Molga e Smutnicki, 2005):

$$f(x) = \sum_{i=1}^{n} {x_i}^2. \qquad \text{(Eq.14)}$$

O espaço de busca é frequentemente restrito ao hipercubo $-5{,}12 \leq x_1 \leq 5{,}12, i = 1, \ldots, n$. O mínimo global $f(x) = 0$ é obtido por $x_i = 0,\ i = 1, \ldots, n$. (Molga e Smutnicki, 2005).

2.5.6 Griewangk's function

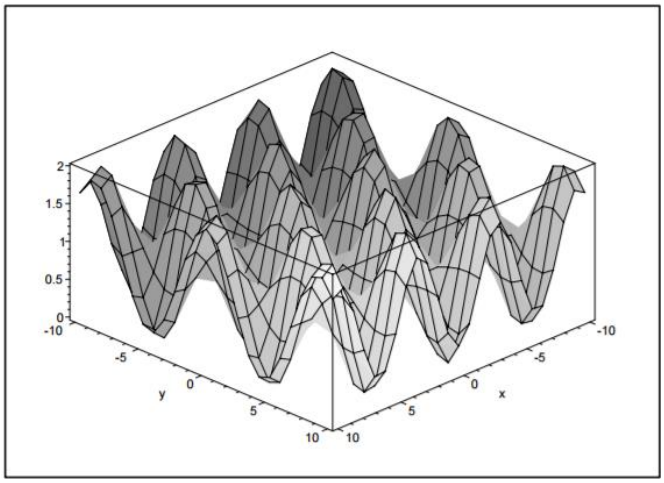

Figura 13. Griewangk's function em 2D vista em média escala.
(Molga e Smutnicki, 2005)

Esta função tem um grande número de mínimos locais difundidos regularmente. A função tem a seguinte definição (Molga e Smutnicki, 2005):

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} {x_i}^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1. \qquad \text{(Eq.15)}$$

O espaço de busca é frequentemente restrito ao hipercubo $-600 \leq x_i \leq 600, i = 1, \dots, n.$ O mínimo global $f(x) = 0$ é obtido por $x_i = 0,\ i = 1, \dots, n.$ A interpretação da função depende da escala de visualização, em escala média percebe-se a existência de locais extremos, em visualização geral sugere-se que é uma função convexa, como na figura 12 (Molga e Smutnicki, 2005).

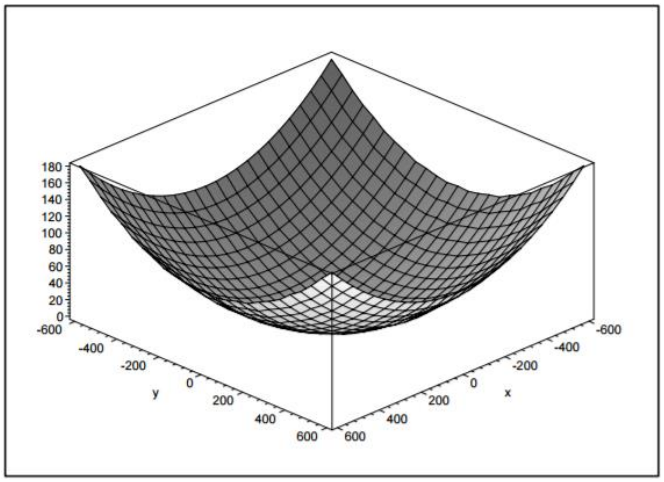

Figura 14. Griewangk's function em 2D.
(Molga e Smutnicki, 2005)

2.5.7 Goldstein-Price's function

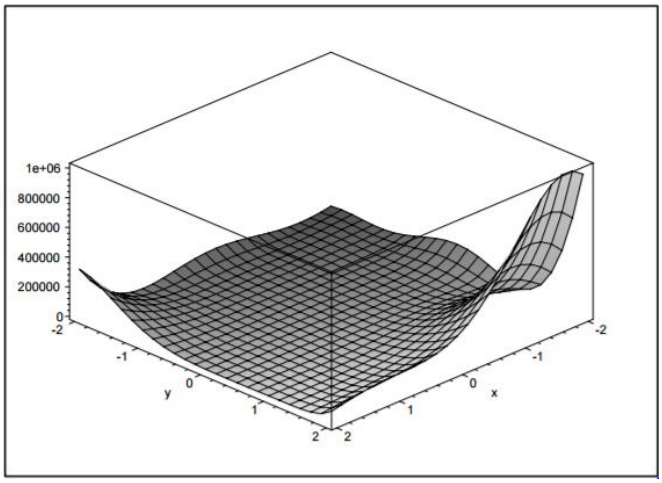

Figura 15. Goldstein-Price's function em 2D.
(Molga e Smutnicki, 2005)

É uma função de teste de otimização global, contém duas variáveis e a seguinte definição (Molga e Smutnicki, 2005):

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]. \quad \text{(Eq.16)}$$

O espaço de busca é frequentemente restrito ao quadrado $-2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2$. O mínimo global $f(x) = 3$ é obtido por $(x_1, x_2) = (0, -1)$ (Molga e Smutnicki, 2005).

2.5.8 Shubert's function

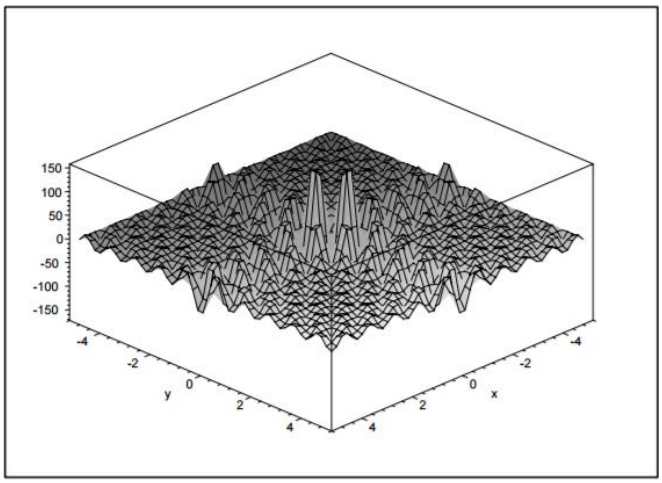

Figura 16. Shubert's function em 2D.
(Molga e Smutnicki, 2005)

É uma função de teste multimodal, contém duas variáveis e a seguinte definição (Molga e Smutnicki, 2005):

$$f(x_1, x_2) = -\sum_{i=1}^{5} i \cos\big((i+1)x_1 + 1\big) \sum_{i=1}^{5} i \cos((i+1)x_2 + 1). \qquad \text{(Eq.17)}$$

O espaço de busca é frequentemente restrito ao quadrado $-5.12 \leq x_1 \leq 5{,}12, -5{,}12 \leq x_2 \leq 5{,}12$ (Molga e Smutnicki, 2005). O valor do mínimo global é $f(x) = -186{,}7309$.

3 METODOLOGIA

3.1 Materiais

Para o desenvolvimento dos algoritmos e realização dos testes de desempenho dos mesmos, utilizou-se o software MATLAB[1], na versão R2013b. Esta ferramenta é muito utilizada para resolver problemas científicos e de engenharia. Tem alta performance no processamento de cálculos matemáticos, além de gráficos embutidos que facilitam a análise e interpretação dos dados (MathWorks, 2017).

Para o tratamento dos dados gerados nos testes de desempenho dos algoritmos no MATLAB, utilizou-se o R[2], uma linguagem e ambiente para computação estatística e gráficos (R-Project, 2017), além da interface gráfica RStudio[3]. Utilizou-se também o Microsoft Excel 2016[4] que consiste em um editor de planilhas amplamente utilizado, devido à quantidade de recursos que o compõe que vão desde sua interface interativa, desempenho na resolução de cálculos e recursos de desenvolvimento de gráficos.

Para a viabilidade do trabalho, serão utilizadas as seguintes ferramentas de hardware e software:

Tabela 1. Materiais utilizados**.**

DESCRIÇÃO	FABRICANTE	VERSÃO/MODELO
Notebook	Sony Vaio	VPCSE15FB
SO Windows[5]	Microsoft	7 – 64 bits
Processador	Intel	i7-2640M 2,80GHz

[1] Disponível em: <https://www.mathworks.com/products/matlab.html>
[2] Disponível em: <https://cran.r-project.org/mirrors.html>
[3] Disponível em: <https://www.rstudio.com/products/rstudio/download/>
[4] Disponível em: <https://products.office.com/pt-BR/business/get-the-most-secure-office-with-excel-2016>
[5] Disponível em: <https://www.microsoft.com/pt-br/software-download/windows7>

Memória RAM	Kingston	6 Gb, DDR3
Disco rígido	Seagate	500Gb, 5400 rpm
MATLAB	MathWorks	R2013b
R	R-Project	3.4.1
RStudio	RStudio	Desktop 1.0.143
Excel	Microsoft	2016

3.2 Implementação e ajuste dos métodos

Implementou-se os algoritmos Evolução Diferencial e Algoritmo Genético Simples e testou-se nas funções definidas na seção 2.5. De modo a permitir a comparação entre os algoritmos supracitados, executou-se cada método 33 vezes em cada umas das funções, esse valor foi definido de forma determinística.

Após a execução dos algoritmos desenvolvidos nas funções de testes, considerou-se a média dos 33 resultados do fitness (cálculo da função objetivo) da melhor solução de cada geração, assim pôde ser avaliada a média de fitness da melhor solução, para cada uma das gerações na resolução de cada uma das funções. Esse procedimento facilitou a análise e comparação gráfica entre os algoritmos. Buscou-se ainda encontrar uma configuração genérica para cada algoritmo, que fosse capaz de resolver a maioria das funções de maneira satisfatória.

Calculou-se ainda a média, mediana, variância, desvio padrão, para o conjunto de testes, em cada uma das funções, considerando um nível de confiança de 95%. Os algoritmos foram automatizados para realizar os 33 testes de maneira consecutiva para tornar o processo menos oneroso, os dados gerados foram salvos em arquivo texto. Realizou-se o tratamento dos dados no RStudio e posteriormente exportou-os para o Excel para que fossem gerados os gráficos.

3.2.1 Especificações dos algoritmos

A codificação utilizada para os cromossomos nos dois algoritmos, SGA e DE, foi a codificação real, onde cada indivíduo da população é composto por duas variáveis com representação em ponto flutuante.

3.2.1.1 Especificações SGA

- Tipo de Seleção: Seleção por torneio binário, apresentado na seção 2.2.2.2;
- Tipo de Mutação: Mutação Gaussiana, esta consiste em perturbar o cromossomo com uma pequena variação de seu valor (Linden, 2008), é uma das mutações

mais populares utilizadas em algoritmos com codificação em ponto flutuante (Von Zuben, 2000);

- Tipo de Recombinação: Crossover Aritmético (Michalewicz, 1996), este tipo de recombinação consiste em gerar descendentes no intervalo entre os progenitores, garantindo que o processo de recombinação não gere indivíduos que não representem uma solução válida.

3.2.1.2 Especificações DE

- Estratégia de Mutação: DE/rand/1/bin, método apresentado na seção 2.3.2;
- Estratégia de Recombinação: Recombinação discreta, apresentada na seção 2.3.3.

3.2.2 Pseudocódigos

Segue abaixo o pseudocódigo do SGA desenvolvido:

Algoritmo 1 : SGA

Input: Nome da função objetivo a ser minimizada *FOBJ*, limites inferiores das variáveis de otimização *LB*, limites superiores das variáveis de otimização *UB*, tamanho da população de soluções *N*, geração atual g, número máximo de gerações $gMax$, probabilidade de cruzamento *Pc*, probabilidade de mutação *Pm*.

Output: Vetor $fvec$ contendo o melhor $f(x)$ de cada geração.

```
1  begin
2      P = generatePopulation(LB,UB,N);          /*Gera população inicial*/
3      jP = rating(FOBJ,P);                      /* Avalia a população gerada */
4      fMin = min(jP);                           /* Resgata a melhor solução encontrada */
5      fVec (g) = fMin;                          /* Salva a melhor solução da primeira geração */
6      while (g < gMax)
7          C = crossover(P, Pc);                 /* Recombinação */
8          M = mutation(C, LB, UB, Pm);          /* Mutação */
9          jM = rating(FOBJ, M);                 /* Avaliação */
10         [U, jU] = selection([P M], [jP jM], N);   /* Seleção */
11         [P, jP, fMin] = elitism([P M],[jP jM],U,jU)]   /* Elitismo */
12         g = g + 1;
13         fVec(g) = fMin;                       /* Armazena a melhor solução da geração corrente
14     End
15     plot(fVec);                               /* Imprime as melhores soluções por geração */
16 end
```

As entradas do algoritmo são o nome da função a qual deseja-se que o algoritmo resolva (Rosenbrock's Valley, Drop Wave e etc.) representado no pseudocódigo como $FOBJ$, os limites inferiores LB e superiores UB das variáveis de otimização, o tamanho da população N de soluções (quantidade de cromossomos utilizados por população), número máximo de gerações $gMax$ e as probabilidades de cruzamento Pc e mutação Pm.

Após informadas as variáveis de entrada, o algoritmo gera a população inicial P de forma aleatória respeitando os limites informados e o tamanho da população. Após a geração da população inicial, é realizada a avaliação de cada cromossomo da população de acordo com a função a ser calculada e retorna-se o vetor jP que é o vetor de custo de

P. A melhor solução do vetor jP é armazenada em um vetor $fVec$ que é utilizado para armazenar a melhor solução de cada geração de cromossomos.

Inicia-se o laço de repetição do algoritmo para que sejam realizados os passos do SGA pelo valor máximo $gMax$ de gerações. Sobre a geração corrente atua-se o operador de recombinação (crossover) por coordenada, ou seja, as soluções são recombinadas 2 a 2, conforme a probabilidade de cruzamento Pc, na ordem em que aparecem em P. A função de recombinação recebe como parâmetros P e Pc e têm-se como saída um vetor de soluções C. Sobre as soluções candidatas geradas pela recombinação, representadas pelo vetor C, atua-se o operador de mutação que recebe como parâmetros C, LB, UB e Pm. A função de mutação irá atuar sobre o vetor C de acordo com Pm e os limites LB e UB a fim de garantir que os cromossomos gerados estejam de acordo com os limites das funções, ao fim retornando um vetor de soluções M.

Após a atuação dos operadores genéticos de recombinação e mutação o vetor de soluções obtido, M, é avaliado através da função de avaliação que recebe como parâmetros Me $FOBJ$ e têm-se como retorno o vetor jM com o custo dos cromossomos de M. Após a atuação dos operadores genéticos, atua-se sobre P e M a função de seleção, que utiliza o método de torneio binário, recebendo como parâmetros a concatenação dos vetores P e M, jP e jM e o tamanho da população N. O retorno do método são dois vetores, o vetor de soluções U referente a atuação sobre a concatenação de P e M e o vetor jU de custos de U.

A fim de garantir que a melhor solução esteja na próxima geração de soluções candidatas, aplica-se o método de elitismo que recebe como parâmetros a concatenação de P e M, jP e jM além de U e jU. O retorno do método são os vetores P de soluções e jP com seus respectivos custos, além da variável $fMin$ com a melhor solução gerada. Armazena-se $fMin$ no vetor $fVec$ e inicia-se o laço de repetição novamente até que se atinja o $gMax$.

Os primeiros testes do SGA foram realizados com população de 10 soluções, critério de parada em 100 gerações, taxa de cruzamento em 60% e taxa de mutação em 10% e 30 % de elitismo. Com esses valores, a convergência na maioria dos testes ficou por volta da 50º geração, o que tornava o critério de parada em 100 gerações desnecessário.

Foi reduzido o critério de parada do algoritmo para 50 gerações. Para evitar possíveis resultados de convergência acima de 50 gerações, foi alterada a porcentagem da taxa de cruzamento para 90%.

Segue abaixo o pseudocódigo do DE desenvolvido:

Algoritmo 2 : DE

Input: Nome da função objetivo a ser minimizada *FOBJ*, limites inferiores das variáveis de otimização *LB*, limites superiores das variáveis de otimização *UB*, tamanho da população de soluções *N*, geração atual *g*, número máximo de gerações *gMax*, probabilidade de recombinação Pc, fator de perturbação *F*.

Output: Vetor b contendo o melhor $f(x)$ de cada geração.

```
1  Begin
2      P = generatePopulation(UB,LB,N);          /* gera a população inicial */
3      cP = rating(FOBJ, P);                     /* avalia a população gerada */
4      while (g < gMax)
5          b(g) = min (cP);                      /* armazena a melhor solução da geração */
6          [V, vPos] = mutation(P,F);            /* mutação diferencial */
7          vExp = crossover(P,V,vPos,Pc);        /* recombinação */
8          vExp = reflection(vExp,LB,UB);        /* reflexão para manter os resultados nos limites */
9          cExp = rating(FOBJ,vExp);             /* avaliação */
10         X = P(:,vPos);
11         jX = cP(vPos);
12         [P, cP] = selection(X, jX, vExp, cExp);   /* seleção */
13         g = g + 1;
14     end
15     plot(b);                                  /* imprime as melhores soluções por geração */
16 end
```

As entradas do algoritmo são o nome da função a qual deseja-se que o algoritmo resolva (Rosenbrock's Valley, Drop Wave e etc.) representado no pseudocódigo como $FOBJ$, os limites inferiores LB e superiores UB das variáveis de otimização, o tamanho da população N de soluções (quantidade de cromossomos utilizados por população), número máximo de gerações $gMax$ e as probabilidades de cruzamento Pc e fator de perturbação F.

Após informadas as variáveis de entrada, o algoritmo gera a população inicial P de forma aleatória respeitando os limites informados e o tamanho da população. Após a geração da população inicial, é realizada a avaliação de cada cromossomo da população de acordo com a função a ser calculada e retorna-se o vetor cP com o custo de cada cromossomo.

Inicia-se o laço de repetição do algoritmo para que sejam realizados os passos do DE pelo valor máximo $gMax$ de gerações. A melhor solução do vetor cP é armazenada em um vetor b que é utilizado para armazenar a melhor solução de cada geração de cromossomos. Posteriormente é realizada a mutação, esta função recebe como

parâmetros o vetor de soluções da geração corrente P e o fator de perturbação F e tem como retorno o vetor doador V e o vetor $vPos$ que tem em cada gene representado um locus do vetor P, mas de forma desordenada, posteriormente utilizado para perturbar os processos de recombinação e seleção com a finalidade de inibir a convergência prematura.

O próximo passo do algoritmo é realizar o processo de recombinação, esta função recebe como parâmetro $P, V, vPos, e\ Pc$. Na recombinação, serão comparados um número aleatório $rand$ e a constante Pc, caso $rand$ seja menor ou igual a Pc o próximo locus do vetor de soluções experimental $vExp$ receberá o valor do mesmo locus em V, caso contrário, é utilizado o vetor $vPos$ para que o locus de $vExp$ receba o valor de um locus aleatório de P. A saída da função é o vetor de soluções experimental $vExp$. Após gerado o vetor de soluções experimental, realiza-se o processo de reflexão para analisar se os cromossomos estão dentro dos limites UB e LB. Posteriormente é realizada a avaliação do custo de $vExp$ através da função de avaliação. Ela recebe como parâmetro $vExp$ e $FOBJ$ e retorna o vetor $cExp$ que representa o custo de cada cromossomo da população $vExp$.

Para o processo de seleção, foram criados os vetores auxiliares X e cX que recebem P e cP respectivamente, porém os locus são embaralhados com a ajuda de $vPos$, já que no processo de seleção os dois vetores são comparados locus a locus, isso dificulta que a comparação seja entre locus de mesmos valores, já que $vExp$ provavelmente recebeu algum cromossomo de P no processo de recombinação. A função de seleção recebe como parâmetros $X, cX, vExp$ e $cExp$ e retorna o vetor P que será a geração corrente na próxima geração e cP que é o vetor custo de P.

Os primeiros testes do DE foram realizados com população de 10 soluções, critério de parada em 100 gerações, probabilidade de recombinação em 60% e fator de perturbação em 0,9. Com esses valores, a convergência na maioria dos testes ficou por volta da 50º geração, o que tornava o critério de parada em 100 gerações desnecessário.

Foi reduzido o critério de parada do algoritmo para 50 gerações. Para evitar possíveis resultados de convergência acima de 50 gerações, foi alterada a probabilidade de recombinação para 90%.

4. RESULTADOS

4.1 Introdução

Neste capítulo serão discutidos os resultados obtidos para cada uma das funções utilizadas. A princípio são descritos os resultados obtidos pelos algoritmos DE e SGA para o conjunto de funções apresentadas na sessão 2.5. Além disso, é feita análise estatística nos resultados dos testes realizados para cada uma das funções avaliadas.

4.2 Resultados dos testes das funções

Após a obtenção dos dados resultantes dos testes das funções para cada algoritmo, parte-se para uma análise descritiva com o objetivo de verificar a existência de diferenças significativas entre o que observa-se pelo DE e pelo SGA para a resolução de cada função.

Os dados dos 33 testes foram gerados e exportados para o RStudio para realizar os cálculos estatísticos, cálculo da média, mediana, variância, desvio padrão e intervalo de confiança para cada uma das funções para posteriormente serem importados para o Excel para a geração dos gráficos.

A tabela 2 mostra a média, para cada geração, nos 33 testes realizados pelo SGA para cada uma das funções:

Tabela 2. Média dos resultados por geração nos 33 testes realizados pelo SGA.

MÉDIA DOS RESULTADOS DE CADA GERAÇÃO PARA OS 33 TESTES PELO SGA								
Geração	**Ackley**	**Drop Wave**	**Goldstein Price**	**Griewangk**	**Rosenbrock's Valley**	**De Jong**	**Shubert**	**Easom**
1	1,3795	-0,8907	123,8533	4,8589	11,0548	1,3696	-52,8547	0,0000
2	0,9499	-0,9184	54,9612	3,1813	6,4652	1,0919	-62,0400	0,0000
3	0,7192	-0,9259	18,1445	2,1934	2,7618	0,5197	-71,9061	-0,0232
4	0,4817	-0,9411	15,6112	1,6831	1,8022	0,3340	-90,5175	-0,0266
5	0,4308	-0,9433	13,6903	1,4081	1,2639	0,2775	-94,9257	-0,0463
6	0,3799	-0,9447	13,0395	0,9759	0,8660	0,2213	-101,2828	-0,0550
7	0,3532	-0,9473	12,4496	0,8218	0,7118	0,1551	-110,8284	-0,0842
8	0,3181	-0,9507	10,4268	0,7154	0,5652	0,1459	-112,3043	-0,0989
9	0,2962	-0,9523	9,7806	0,6133	0,5193	0,1394	-115,5410	-0,1271
10	0,2440	-0,9545	9,2866	0,5401	0,5157	0,1107	-128,2887	-0,1573
11	0,2109	-0,9551	8,3367	0,4769	0,5039	0,0976	-133,0755	-0,1792
12	0,1692	-0,9553	7,9421	0,4552	0,4973	0,0697	-134,1129	-0,2207
13	0,1559	-0,9558	7,8623	0,4234	0,4846	0,0687	-136,7799	-0,2533
14	0,1521	-0,9567	7,7169	0,3812	0,4824	0,0546	-138,9251	-0,2533
15	0,1366	-0,9575	7,0814	0,3709	0,4818	0,0518	-144,0837	-0,2849
16	0,1266	-0,9575	7,0580	0,3656	0,4750	0,0437	-146,1182	-0,2862
17	0,1245	-0,9579	6,5219	0,3369	0,4737	0,0422	-151,9877	-0,2886
18	0,1213	-0,9582	5,7984	0,2895	0,4375	0,0405	-155,0213	-0,3224
19	0,1157	-0,9582	5,2838	0,2733	0,4221	0,0392	-157,4899	-0,3840
20	0,1048	-0,9593	5,2293	0,2547	0,4080	0,0308	-160,0406	-0,4416

21	0,0822	-0,9593	4,9570	0,2418	0,3202	0,0297	-163,4738	-0,4724
22	0,0753	-0,9593	4,4728	0,2407	0,3151	0,0250	-166,5099	-0,4787
23	0,0678	-0,9593	4,3692	0,2403	0,2955	0,0231	-169,4714	-0,5587
24	0,0643	-0,9604	4,3329	0,2400	0,2857	0,0230	-169,5899	-0,5603
25	0,0637	-0,9604	4,3049	0,2377	0,2842	0,0224	-174,4218	-0,5753
26	0,0636	-0,9605	4,0889	0,2377	0,2695	0,0222	-174,4333	-0,5941
27	0,0635	-0,9605	4,0737	0,2288	0,2695	0,0214	-175,8017	-0,6016
28	0,0634	-0,9605	3,7756	0,2229	0,2605	0,0196	-176,0779	-0,6163
29	0,0634	-0,9607	3,7653	0,2227	0,2605	0,0188	-176,6469	-0,6175
30	0,0628	-0,9607	3,6955	0,2171	0,2599	0,0167	-176,8220	-0,6265
31	0,0604	-0,9609	3,6577	0,2170	0,2598	0,0151	-179,3811	-0,6491
32	0,0556	-0,9622	3,6378	0,2169	0,2530	0,0151	-179,6821	-0,6612
33	0,0498	-0,9622	3,6026	0,1956	0,2351	0,0146	-180,2560	-0,6826
34	0,0468	-0,9628	3,5993	0,1953	0,2261	0,0146	-181,0140	-0,6960
35	0,0443	-0,9628	3,2880	0,1952	0,2261	0,0145	-181,5784	-0,6960
36	0,0396	-0,9628	3,2685	0,1952	0,2252	0,0144	-181,8497	-0,7247
37	0,0371	-0,9628	3,2474	0,1948	0,2244	0,0090	-182,0619	-0,7613
38	0,0352	-0,9628	3,2471	0,1948	0,2244	0,0082	-182,1888	-0,7640
39	0,0331	-0,9644	3,1992	0,1864	0,2236	0,0082	-182,3929	-0,7798
40	0,0330	-0,9644	3,1992	0,1668	0,2234	0,0082	-182,4863	-0,8481
41	0,0329	-0,9648	3,1992	0,1657	0,2234	0,0081	-182,5178	-0,8652
42	0,0296	-0,9657	3,1980	0,1656	0,2234	0,0081	-183,6465	-0,8765
43	0,0295	-0,9657	3,1904	0,1631	0,2223	0,0079	-183,7309	-0,8833
44	0,0295	-0,9657	3,1904	0,1631	0,2187	0,0060	-184,4386	-0,8847
45	0,0295	-0,9657	3,1892	0,1631	0,2187	0,0048	-184,4422	-0,9054
46	0,0295	-0,9657	3,1834	0,1631	0,2187	0,0048	-184,5003	-0,9064
47	0,0292	-0,9657	3,1795	0,1619	0,2183	0,0041	-184,7581	-0,9064
48	0,0287	-0,9658	3,1330	0,1584	0,1758	0,0040	-184,8669	-0,9067
49	0,0287	-0,9658	3,1313	0,1571	0,1758	0,0040	-185,0402	-0,9067
50	0,0287	-0,9659	3,1005	0,1567	0,1753	0,0029	-185,1359	-0,9081

Como verificou-se na tabela 2, na geração 50 a média dos 33 testes resultou em um valor bem próximo ao mínimo global em todas as funções, conseguindo cravá-lo nas funções Ackley e De Jong. Para estas duas funções, percebe-se que o algoritmo tem facilidade em encontrar o ótimo global, uma vez que conseguem encontrá-lo muito antes da geração 50, na vigésima primeira geração para a função Ackley e na décima primeira para a De Jong.

Tabela 3. Resultados dos testes estatísticos do SGA para os dados da tabela 2.

	Ackley	**Drop Wave**	**Goldstein Price**	**Griewangk**	**Rosenbrock's Valley**	**De Jong**	**Shubert**	**Easom**
Média	0,1674	-0,9565	9,1510	0,5305	0,7681	0,1060	-155,5000	-0,5089
Mediana	0,0637	-0,9604	4,1970	0,2377	0,2769	0,0223	-174,4000	-0,5847
Variância	0,0641	0,0002	335,2462	0,7072	3,1236	0,0639	1288,3540	0,0942
Desvio Padrão	0,2531	0,0134	18,3097	0,8409	1,7674	0,2527	35,8937	0,3069
Intervalo confiança	0,0955 - 0,2394	-0,9603 - -0,9527	3,9475 - 14,3546	0,2915 - 0,7695	0,2658 - 1,2704	0,0342 - 0,1779	-165,7477 - -145,3460	-0,5961 - -0,4217

A tabela 3 apresenta os dados estatísticos aplicados aos resultados da tabela 2 para cada uma das funções. Analisando a média, observou-se que a função Ackley resultou em valores bem próximos ao ótimo global em todo o percurso de 50 gerações, onde iniciou-se com $f(x) = 1{,}3795$ e finalizou em $f(x) = 0{,}0287$, com média de

0,1674. Situação similar ocorre também com as funções De Jong e Drop Wave, esta última ainda mais perceptível, pois os resultados das gerações foi ainda mais próximos entre si, com média $f(x) = -0{,}9565$, mediana $f(x) = -0{,}9604$ e desvio padrão $f(x) = 0{,}0133$.

A tabela 4 mostra a média, para cada geração, nos 33 testes realizados pelo DE para cada uma das funções:

Tabela 4. Média dos resultados por geração nos 33 testes realizados pelo DE.

MÉDIA DOS RESULTADOS DE CADA GERAÇÃO PARA OS 33 TESTES PELO DE								
Geração	**Ackley**	**Drop Wave**	**Goldstein Price**	**Griewangk**	**Rosenbrock's Valley**	**De Jong**	**Shubert**	**Easom**
1	1,6456	-0,8462	204,9511	7,6797	16,8804	1,9966	-45,6942	-0,0186
2	1,2957	-0,8649	106,7809	5,8071	10,1008	1,5532	-47,4977	-0,0186
3	1,1574	-0,8861	58,0803	4,8200	5,5096	1,0560	-52,0702	-0,0186
4	0,9124	-0,9055	56,7037	4,2346	3,2474	0,8091	-56,6336	-0,0186
5	0,7730	-0,9113	42,1707	3,4315	2,5533	0,5780	-63,8375	-0,0186
6	0,6492	-0,9232	38,6909	2,1725	1,8277	0,4646	-68,1739	-0,0191
7	0,5440	-0,9295	31,7363	1,7359	1,6021	0,3353	-75,9628	-0,0191
8	0,4556	-0,9331	26,4024	1,5232	1,3243	0,2533	-81,3824	-0,0191
9	0,3647	-0,9383	22,0932	1,1589	1,0514	0,2238	-87,8308	-0,0191
10	0,2904	-0,9437	19,1783	0,9241	0,9199	0,1694	-90,3846	-0,0203
11	0,2410	-0,9497	16,0862	0,7912	0,7718	0,0918	-94,7698	-0,0203
12	0,1948	-0,9567	15,5908	0,6885	0,7517	0,0677	-104,2346	-0,0203
13	0,1766	-0,9613	14,1707	0,6191	0,4967	0,0472	-108,9951	-0,0203
14	0,1496	-0,9615	12,8557	0,5765	0,4518	0,0457	-117,1537	-0,0203
15	0,1322	-0,9631	11,1276	0,4781	0,3672	0,0339	-122,7887	-0,0248
16	0,0927	-0,9663	10,5578	0,4616	0,3158	0,0188	-123,8483	-0,0445
17	0,0676	-0,9673	9,4886	0,4091	0,2963	0,0137	-125,8107	-0,0472
18	0,0573	-0,9705	9,1314	0,3751	0,2929	0,0102	-130,3756	-0,0609
19	0,0448	-0,9721	8,5675	0,3546	0,2767	0,0066	-131,7602	-0,0609
20	0,0407	-0,9736	8,0802	0,3301	0,2624	0,0049	-134,4664	-0,0679
21	0,0374	-0,9736	7,8677	0,3152	0,2383	0,0046	-140,7808	-0,0769
22	0,0324	-0,9778	7,7127	0,2667	0,2248	0,0028	-146,6095	-0,0899
23	0,0283	-0,9793	6,6362	0,2388	0,2121	0,0016	-148,6882	-0,0992
24	0,0245	-0,9812	6,5976	0,2184	0,2009	0,0009	-151,0656	-0,1400
25	0,0222	-0,9812	6,4203	0,2123	0,1990	0,0007	-159,3095	-0,2153
26	0,0169	-0,9814	6,3504	0,2017	0,1775	0,0006	-159,9030	-0,2755
27	0,0153	-0,9814	5,6656	0,1864	0,1166	0,0005	-161,1818	-0,2905
28	0,0125	-0,9819	5,3296	0,1829	0,1123	0,0004	-161,9908	-0,3227
29	0,0118	-0,9825	4,8729	0,1767	0,0983	0,0003	-164,6609	-0,3784
30	0,0096	-0,9825	4,8266	0,1606	0,0983	0,0002	-167,2455	-0,4250
31	0,0075	-0,9825	4,0191	0,1452	0,0944	0,0002	-171,7686	-0,4439
32	0,0052	-0,9827	3,7174	0,1428	0,0869	0,0001	-172,0111	-0,4660
33	0,0043	-0,9829	3,4368	0,1424	0,0845	0,0000	-174,7619	-0,4984
34	0,0040	-0,9829	3,4033	0,1402	0,0700	0,0000	-177,5344	-0,5173
35	0,0033	-0,9835	3,3541	0,1345	0,0659	0,0000	-179,5917	-0,5467
36	0,0028	-0,9845	3,2985	0,1277	0,0631	0,0000	-181,1659	-0,5833
37	0,0022	-0,9847	3,2879	0,1268	0,0593	0,0000	-181,4043	-0,5865
38	0,0020	-0,9856	3,0843	0,1124	0,0565	0,0000	-181,5350	-0,6098
39	0,0015	-0,9856	3,0520	0,1073	0,0503	0,0000	-182,7376	-0,6602
40	0,0014	-0,9857	3,0403	0,1067	0,0455	0,0000	-183,2101	-0,7030
41	0,0012	-0,9858	3,0359	0,1057	0,0451	0,0000	-183,3750	-0,7605
42	0,0009	-0,9860	3,0236	0,0945	0,0432	0,0000	-183,5365	-0,7649
43	0,0008	-0,9861	3,0219	0,0905	0,0425	0,0000	-183,7398	-0,7915

44	0,0007	-0,9861	3,0146	0,0868	0,0371	0,0000	-183,9515	-0,8195
45	0,0007	-0,9862	3,0080	0,0846	0,0359	0,0000	-184,3707	-0,8667
46	0,0005	-0,9862	3,0061	0,0794	0,0301	0,0000	-184,4585	-0,8907
47	0,0004	-0,9863	3,0053	0,0774	0,0262	0,0000	-185,1109	-0,9106
48	0,0004	-0,9864	3,0038	0,0759	0,0235	0,0000	-185,1311	-0,9116
49	0,0003	-0,9864	3,0031	0,0753	0,0224	0,0000	-185,7434	-0,9221
50	0,0003	-0,9864	3,0016	0,0728	0,0216	0,0000	-185,7540	-0,9365

Como verificou-se na tabela 4, na geração 50 a média dos 33 testes resultou em um valor bem próximo ao mínimo global em todas as funções, conseguindo cravá-lo nas funções Ackley, De Jong, Goldstein Price, Griewangk e Rosenbrock's Valley. Para estas funções, percebe-se que o algoritmo tem facilidade em encontrar o ótimo global, uma vez que conseguem encontrá-lo muito antes da geração 50, na décima sexta geração para a função Ackley, trigésima oitava para a Goldstein Price, quadragésima segunda, vigésima nona para Rosenbrock's Valley e décima primeira para De Jong.

Tabela 5. Resultados dos testes estatísticos do DE para cada função

	Ackley	**Drop Wave**	**Goldstein Price**	**Griewangk**	**Rosenbrock's Valley**	**De Jong**	**Shubert**	**Easom**
Média	0,1907	-0,9646	16,9708	0,8572	1,0396	0,1559	-141,1200	-0,3416
Mediana	0,0196	-0,9813	6,3854	0,2070	0,1882	0,0007	-159,6063	-0,2454
Variância	0,1334	0,0011	1068,9861	2,4935	7,8500	0,1556	1993,4363	0,1095
Desvio Padrão	0,3653	0,0326	32,6954	1,5791	2,8018	0,3944	44,6479	0,3309
Intervalo confiança	0,0859 - 0,2956	-0,9739 - -0,9552	7,5846 - 26,3571	0,4039 - 1,3105	0,2353 - 1,8440	0,0426 - 0,2691	-153,9376 - -128,3023	-0,4366 - -0,2466

A tabela 5 apresenta os dados estatísticos aplicados aos resultados da tabela 4 para cada uma das funções. Analisando a média, observou-se que a função De Jong resultou em valores bem próximos ao ótimo global em todo o percurso de 50 gerações, onde iniciou-se com $f(x) = 1{,}9966$ e finalizou em $f(x) = 0{,}0000$, com média de 0,1559. Situação similar ocorre também com a função Drop Wave, onde os resultados das gerações foi ainda mais próximos entre si, com média $f(x) = -0{,}9646$, mediana $f(x) = -0{,}9813$ e desvio padrão $f(x) = 0{,}0326$.

A seguir é apresentada a análise sobre a comparação entre os dois algoritmos para cada uma das funções.

4.2.1 Função Rosenbrock's Valley

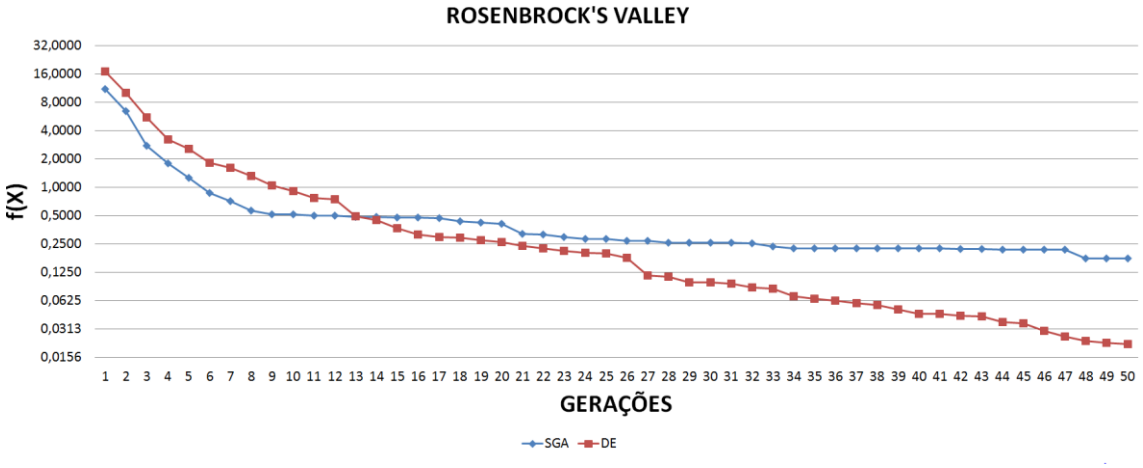

Figura 17. Resultados da função Rosenbrock's para o DE e SGA.

A figura 17 mostra o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 9. A escala dos resultados $f(x)$ foi alterada à medida que os resultados ficaram mais específicos. Essa abordagem facilitou a visualização dos resultados dos algoritmos principalmente por um deles ter apresentado um resultado ótimo muito mais específico, embora os dois tenham convergido para o mínimo global da função, $f(x) = 0$.

Observou-se que o DE mantém uma trajetória de resolução com variação entre as gerações de maneira uniforme desde a primeira geração até a quinquagésima geração, tendo algumas variações mais significativas visíveis entre a décima segunda e décima terceira e entre a vigésima sexta e vigésima sétima geração. O valor de $f(x)$ iniciou-se com o valor $f(x) = 16{,}8804$ e convergiu ao fim do teste, na geração 50 para o valor mínimo $f(x) = 0{,}0216$.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora nas primeiras nove gerações o algoritmo tenha conseguido uma melhor aproximação do mínimo, variando de $f(x) = 11{,}0548$ na primeira geração para $f(x) = 0{,}5193$ na nona, a partir daí, pôde notar variações significativas somente entre a vigésima e vigésima primeira geração e entre a quadragésima sétima e quadragésima oitava geração, finalizando com o mínimo global $f(x) = 0{,}1753$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, no SGA a convergência para o valor $f(x) = 0$ se dá de maneira satisfatória com mínimo global $f(x) = 0{,}1753$, embora quando comparado ao DE, mínimo global

$f(x) = 0{,}0216$ percebe-se que a diferença de desempenho em relação ao mínimo global encontrado é muito grande.

4.2.2 Função Ackley

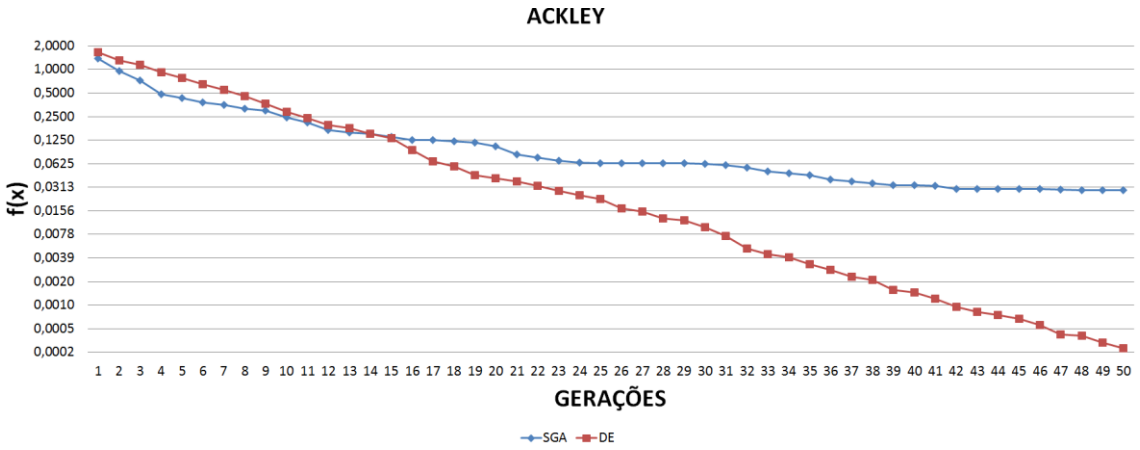

Figura 18. Resultados da função Ackley para o DE e SGA.

A figura 18 descreve o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 11. A escala dos resultados $f(x)$ foi alterada à medida que os resultados ficaram mais específicos, essa abordagem facilitou a visualização dos resultados dos algoritmos principalmente por um deles ter apresentado um resultado ótimo muito mais específico, embora os dois tenham convergido para o mínimo global da função, $f(x) = 0$.

Observou-se que o DE mantém uma trajetória de resolução uniforme desde a primeira geração até a quinquagésima geração, tendo pequenas variações visíveis por volta da décima quinta e trigésima segunda geração. Essa trajetória uniforme iniciou-se com o valor $f(x) = 1{,}6456$ e convergiu ao fim do teste, na geração 50 para o valor mínimo $f(x) = 0{,}0003$.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora nas primeiras nove gerações o algoritmo tenha conseguido uma melhor aproximação do mínimo, variando de $f(x) = 1{,}3795$ na primeira geração para $f(x) = 0{,}2962$ na nona. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = 0{,}0287$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém no SGA a convergência para o valor $f(x) = 0$ se dá de maneira

satisfatória com mínimo global $f(x) = 0{,}0287$, embora quando comparado ao DE, mínimo global $f(x) = 0{,}0003$ percebe-se que a diferença de desempenho em relação ao mínimo global encontrado é muito grande.

4.2.3 Função Drop Wave

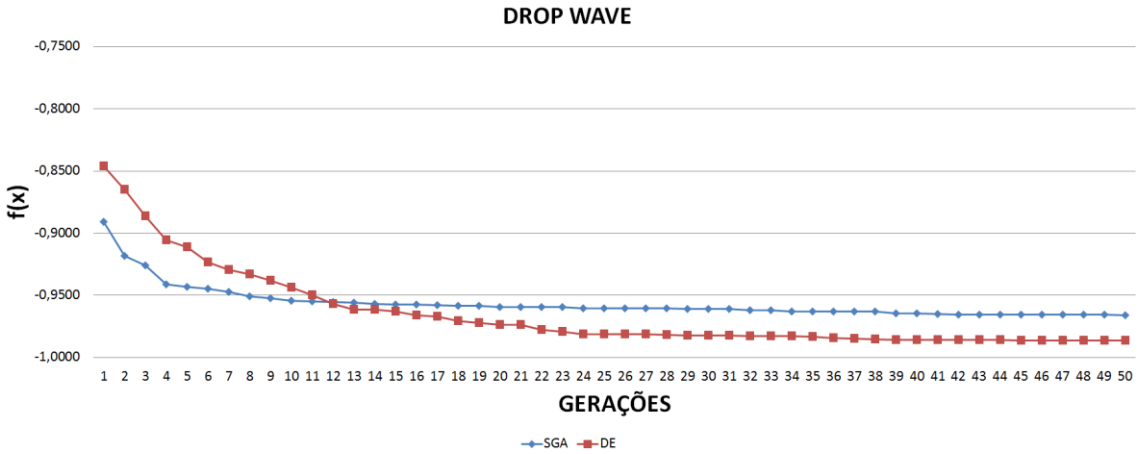

Figura 19. Resultados da função Drop Wave para o DE e SGA.

A figura 19 apresenta o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 12.

Observou-se que o DE apresentou, em sua convergência para o ótimo global, uma variação considerável da primeira geração até a décima terceira, deste intervalo, a variação mais notável pode ser vista da primeira para a quarta geração, onde o salto foi de $f(x) = -0{,}8462$ para $f(x) = -0{,}9055$, respectivamente. Após a décima terceira geração, os valores de $f(x)$ sofreram pouca variação até fechar com o ótimo global $f(x) = -0{,}9864$.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora nas primeiras dez gerações o algoritmo tenha conseguido uma melhor aproximação do mínimo, variando de $f(x) = -0{,}8907$ na primeira geração para $f(x) = -0{,}9545$ na décima, neste intervalo a variação mais notável pode ser observada da primeira à quarta geração, onde variou-se de $f(x) = -0{,}8907$ a $f(x) = -0{,}9411$, respectivamente. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = -0{,}9659$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém no SGA a convergência para o valor $f(x) = -1$ se dá de maneira satisfatória e relativamente rápida, com mínimo global $f(x) = -0{,}9659$, já o DE conseguiu um resultado final ligeiramente melhor, $f(x) = -0{,}9864$, embora tenha demorado mais gerações para convergir para valores próximos ao global encontrado.

4.2.4 Função De Jong's

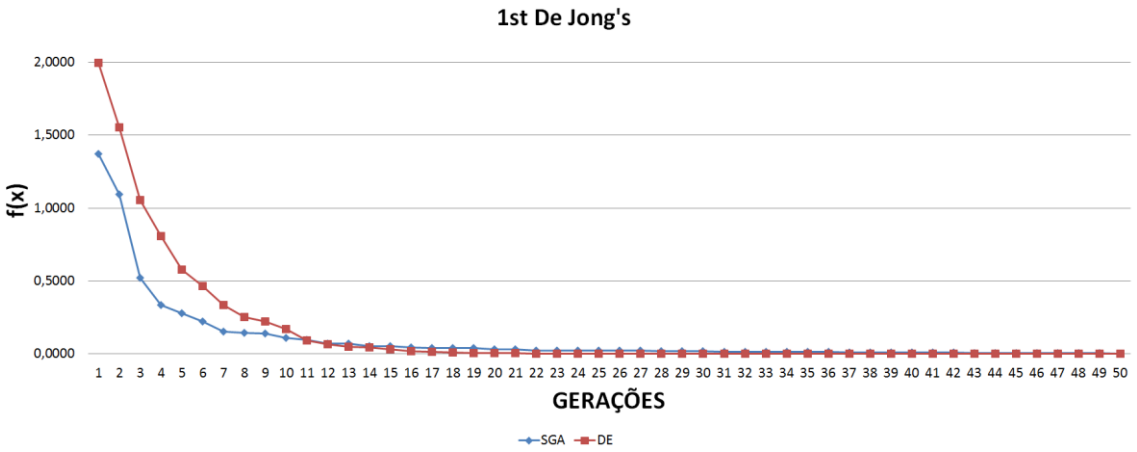

Figura 20. Resultados da função De Jong's para o DE e SGA.

A figura 20 ilustra o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 14.

Observou-se que o DE apresentou, em sua convergência para o ótimo global, uma variação considerável da primeira geração até a décima primeira, deste intervalo, a variação mais notável pode ser vista da primeira para a oitava geração, onde o salto foi de $f(x) = 1{,}9966$ para $f(x) = 0{,}2533$, respectivamente. Após a décima primeira geração, os valores de $f(x)$ sofreram pouca variação até fechar com o ótimo global $f(x) = 0{,}0000$.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora nas primeiras sete gerações o algoritmo tenha conseguido uma melhor aproximação do mínimo, variando de $f(x) = 1{,}3696$ na primeira geração para $f(x) = 0{,}1551$ na sétima. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = 0{,}0029$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém no SGA a convergência para o valor $f(x) = 0$ se dá de maneira

satisfatória e relativamente rápida, com mínimo global $f(x) == 0{,}0029$, já o DE conseguiu um resultado final ligeiramente melhor, $f(x) = 0{,}0000$, embora tenha demorado mais gerações para convergir para valores próximos ao global encontrado.

4.2.5 Função Griewangk's

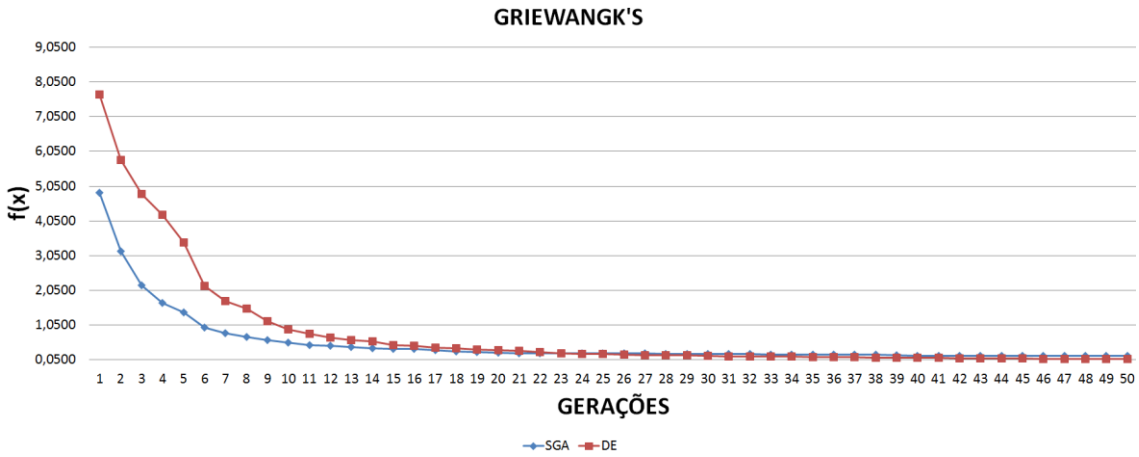

Figura 21. Resultados da função Griewangk's para o DE e SGA.

A figura 21 mostra o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 15.

Observou-se que o DE apresentou, em sua convergência para o ótimo global, uma variação considerável da primeira geração até a décima quinta, deste intervalo, a variação mais notável pode ser vista da primeira para a sexta geração, onde o salto foi de $f(x) = 7{,}6797$ para $f(x) = 2{,}1725$, respectivamente. Após a décima quinta geração, os valores de $f(x)$ sofreram pouca variação até fechar com o ótimo global $f(x) = 0{,}0728$.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora nas primeiras dez gerações o algoritmo tenha conseguido uma melhor aproximação do mínimo, variando de $f(x) = 4{,}8589$ na primeira geração para $f(x) = 0{,}5401$ na décima. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = 0{,}1567$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém no SGA a convergência para o valor $f(x) = 0$ se dá de maneira satisfatória e relativamente rápida, com mínimo global $f(x) = 0{,}1567$, já o DE

conseguiu um resultado final ligeiramente melhor, $f(x) = 0{,}0728$, embora tenha demorado mais gerações para convergir para valores próximos ao global encontrado.

4.2.6 Função Goldstein-Price's

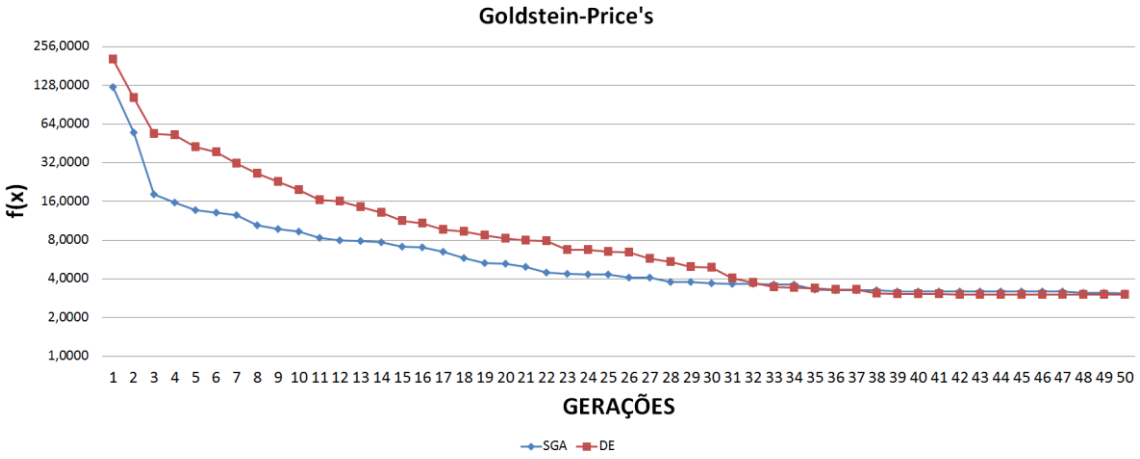

Figura 22. Resultados da função Goldstein-Price's para o DE e SGA.

A figura 22 apresenta o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 16.

Observou-se que o DE apresentou uma convergência uniforme da terceira à trigésima terceira geração. Neste intervalo, a variação foi de $f(x) = 53{,}7624$ para $f(x) = 4{,}0509$, da terceira para a trigésima primeira, respectivamente. A partir dali, a variação até o mínimo global $f(x) = 3{,}0017$ foi pouca entre as gerações.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora percebeu-se uma convergência uniforme da terceira à vigésima segunda geração. Neste intervalo, a variação foi de $f(x) = 18{,}1445$ para $f(x) = 4{,}4728$, respectivamente. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = 3{,}1005$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém no SGA a convergência para o valor $f(x) = 3$ se dá de maneira satisfatória e relativamente rápida, com mínimo global $f(x) = 3{,}1005$, já o DE conseguiu um resultado final ligeiramente melhor, $f(x) = 3{,}0017$, embora tenha demorado bem mais gerações para convergir para valores próximos ao global encontrado.

4.2.7 Função Shubert's

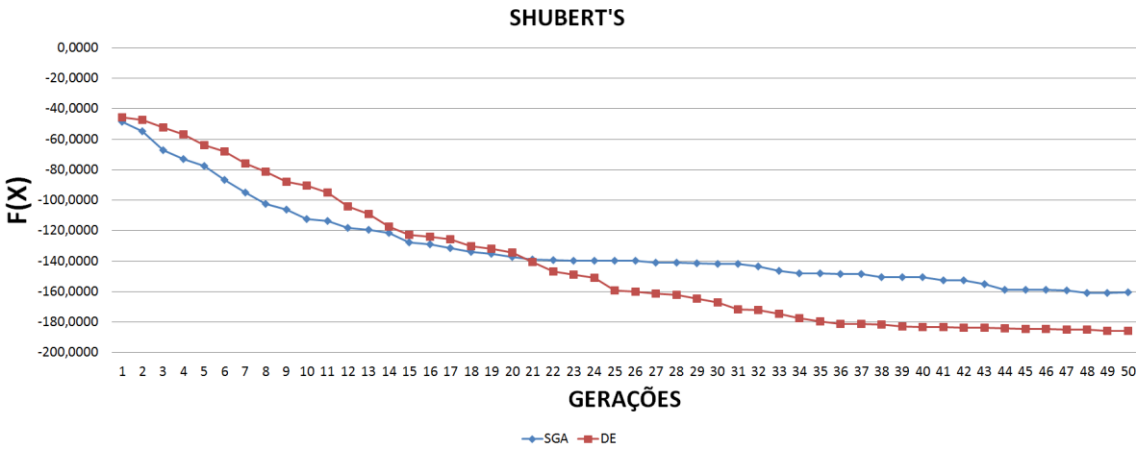

Figura 23. Resultado para a função Shubert's para o DE e SGA.

A figura 23 apresenta o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 17.

Observou-se que o DE apresentou uma convergência uniforme da primeira à trigésima sexta geração. Neste intervalo, a variação foi de $f(x) = -45{,}6942$ para $f(x) = -181{,}1659$, da terceira para a trigésima primeira, respectivamente. A partir dali, a variação até o mínimo global $f(x) = -185{,}7540$ foi pouca entre as gerações.

Para o SGA, observou-se que o algoritmo, no geral, teve uma variação menor entre as gerações, embora percebeu-se uma convergência uniforme da primeira à vigésima primeira geração. Neste intervalo, a variação foi de $f(x) = -48{,}3808$ para $f(x) = -139{,}0280$, respectivamente. Da vigésima primeira a trigésima primeira os resultados pouco se alteraram, chegando a $f(x) = -141{,}8148$. Da trigésima primeira a quadragésima quarta geração, pôde-se observar uma evolução uniforme de geração pra geração, chegando ao valor de $f(x) = -158{,}7176$. A partir dali, houve menor variação entre as gerações, com o mínimo global encontrado $f(x) = -160{,}3340$.

Note que a média dos 33 resultados para a geração 50 foi de $f(x) = -160{,}3340$, valor bem abaixo do mínimo global da equação 17, $f(x) = -186{,}7309$, utilizando a configuração genérica proposta na seção 3.2. Por esta razão, foi necessário realizar novos testes para ajustar o algoritmo para a resolução desta equação de forma mais satisfatória, convergindo para o valor próximo ao ótimo global da mesma. Após

realizados os testes, ajustou-se a taxa de mutação para 0,40, com este valor a função foi capaz de convergir para resultados mais satisfatório, como pode ser visto na figura 25.

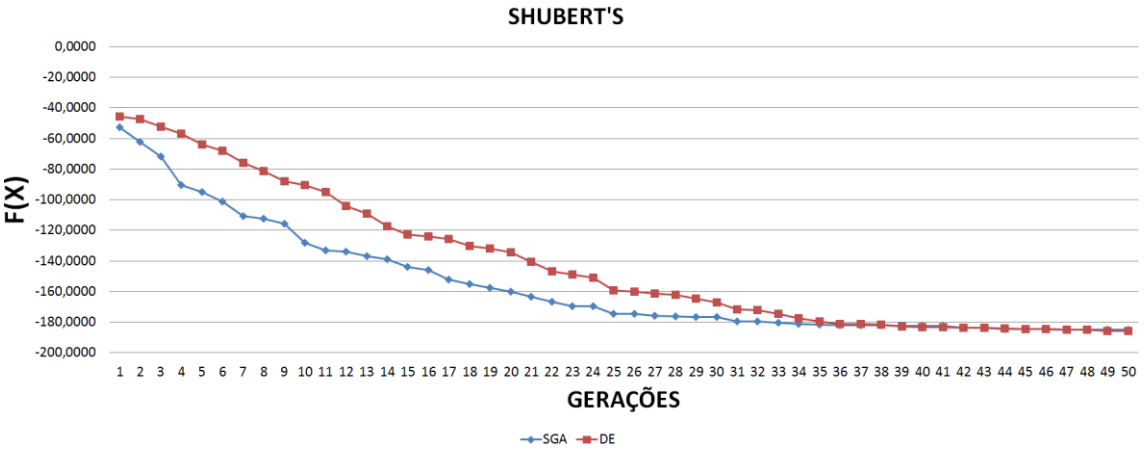

Figura 24. Resultado da função Shubert's para o SGA e DE após ajuste no SGA.

Analisando os novos resultados do algoritmo SGA , ilustrados na Figura 24, percebe-se que com o ajuste na taxa de mutação, o algoritmo conseguiu convergir para um valor satisfatório. A variação de $f(x)$ da primeira para a décima quarta geração foi bem acentuada, variando de $f(x) = -52{,}8547$ para $f(x) = -138{,}9251$, respectivamente. A partir dali, a variação foi uniforme até a vigésima quinta geração, resultando em $f(x) = -174{,}4218$. Da vigésima quinta até a quinquagésima geração, a variação foi pequena entre as gerações, finalizando com ótimo global $f(x) = -185{,}1359$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém para o SGA foi necessário aumentar a taxa de mutação para que fosse possível o resultado convergir para um valor mais próximo ao mínimo global da função.

No SGA, a convergência para o valor $f(x) = -186{,}7309$ se deu de maneira satisfatória e relativamente rápida, com mínimo global $f(x) = -185{,}1359$ encontrado.

4.2.8 Função Easom's

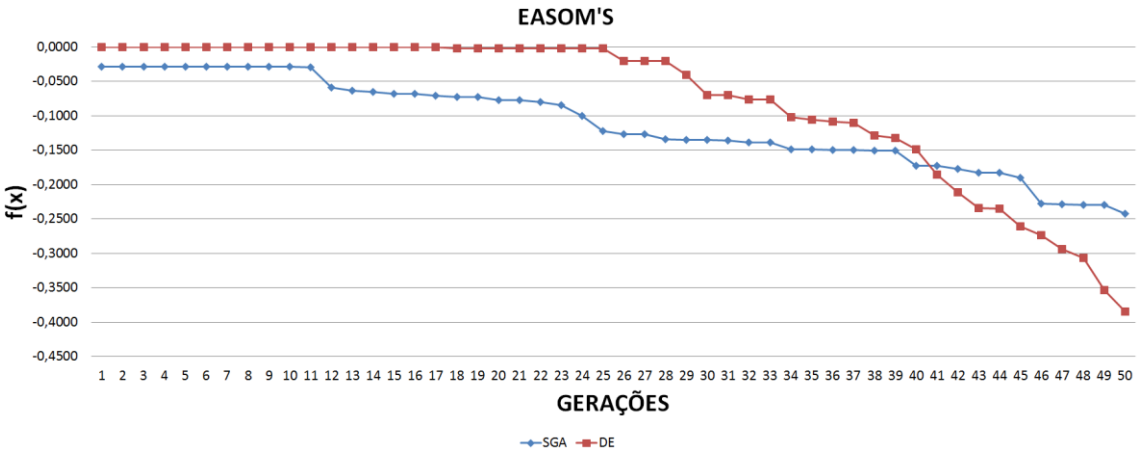

Figura 25. Resultado da função Easom's para o DE e SGA.

A figura 25 mostra o resultado da média dos resultados dos 33 testes para cada uma das cinquenta gerações para os dois algoritmos, DE e SGA na resolução da equação 13.

Percebeu-se que o DE mantém o valor $f(x) = 0$ da primeira até a vigésima quinta geração. A partir dali, o algoritmo começa o desenrolar para a o mínimo de maneira acentuada e contínua, finalizando com valor $f(x) = -0{,}3842$.

O SGA começou com um resultado $f(x) = -0{,}0286$ e não teve alteração até a décima geração. Observou-se uma variação uniforme da décima segunda à vigésima terceira geração, variando de $f(x) = -0{,}0586$ até $f(x) = -0{,}0850$. Houve outro trecho de variação uniforme, pôde ser observado da vigésima quinta à quadragésima sexta quinta, variação de $f(x) = -0{,}1226$ até $f(x) = -0{,}1904$. O algoritmo finaliza com o valor $f(x) = -0{,}2427$.

Pôde-se notar até agora que os dois algoritmos encontraram valores muito distantes do mínimo global da equação 13, $f(x) = -1$, utilizando a configuração genérica proposta no seção 3.2. Diante disto, foi necessário realizar novos testes para ajustar os algoritmos para a resolução desta equação de forma mais satisfatória, convergindo para o valor próximo ao ótimo global da mesma.

Para o SGA, depois de realizados os testes, ajustou-se a taxa de mutação para 0,40, com este valor a função foi capaz de convergir para resultados mais satisfatórios. Para o DE, após realizados os testes, alterou-se o fator de perturbação de 0,9 para 0,4. Com essa alteração, o algoritmo foi capaz de convergir para valores de $f(x)$ mais

próximas de $f(x) = -1$, como pode ser observado na figura 27 que mostra o resultado dos dois algoritmos em teste da equação 13, após os ajustes comentados.

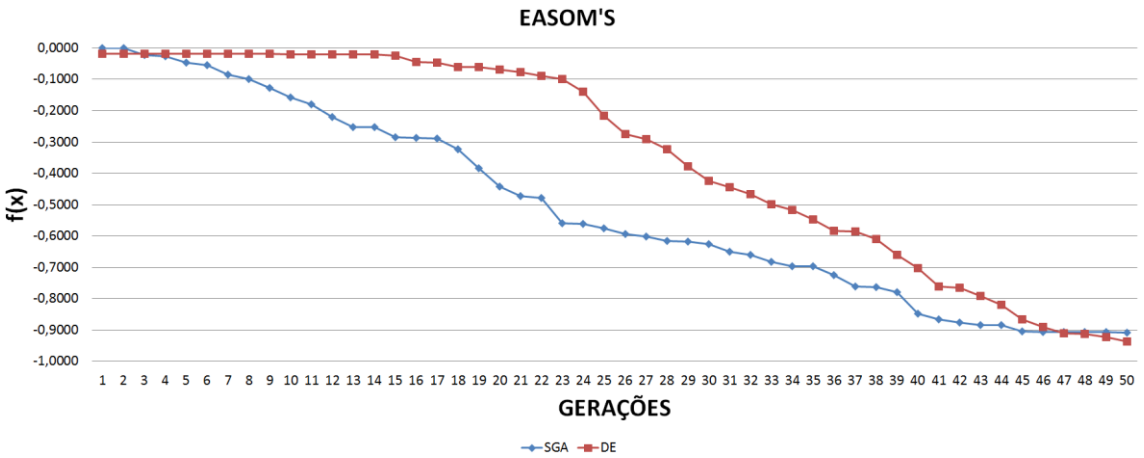

Figura 26. Resultado da função Easom's para DE e SGA após ajustes dos algoritmos.

Analisando os novos resultados do algoritmo SGA, percebeu-se que com o ajuste na taxa de mutação o algoritmo conseguiu convergir para um valor satisfatório. Os valores de $f(x)$ alteraram-se de maneira uniforme desde a primeira à quadragésima geração, partindo de $f(x) = 0{,}0000$ até $f(x) = -0{,}8481$. Dali à quinquagésima geração a variação entre o $f(x)$ entre as gerações é pequena, finalizando no valor $f(x) = -0{,}9081$.

Analisando os novos resultados do algoritmo DE, verificou-se que houve uma variação muito pequena da primeira até a décima quinta geração, variando de $f(x) = -0{,}0186$ à $f(x) = -0{,}0248$. Dali até a vigésima terceira geração foi possível visualizar uma variação uniforme de $f(x)$ entre as gerações, ficando em $f(x) = -0{,}0992$ na vigésima terceira. Da vigésima terceira para a vigésima sétima geração houve uma variação bem acentuada, de $f(x) = -0{,}0992$ para $f(x) = -0{,}2905$, respectivamente. Dali em diante a variação percebida foi uniforme, finalizando em $f(x) = -0{,}9365$.

Conclui-se que os dois algoritmos conseguem convergir para o mínimo global da função, porém foi necessário aumentar a taxa de mutação no SGA e diminuir o fator de perturbação no DE para que fosse possível o resultado convergir para um valor mais próximo ao mínimo global da função.

5. CONCLUSÃO

Conforme apresentado na Seção 1, os algoritmos evolucionários se mostram um conjunto de métodos bastante eficientes para a resolução de problemas de otimização, conseguindo encontrar resultados mínimos globais em problemas que métodos tradicionais não conseguem ou a solução se torna dispendiosa. A diferença de abordagem e funcionamento dos algoritmos SGA e DE justificou a contextualização do trabalho.

O intuito deste trabalho foi avaliar o desempenho dos algoritmos evolucionários SGA e DE na resolução de funções de benchmark irrestritas. A escolha por funções de benchmark ao invés de problemas reais de otimização para a comparação entre os algoritmos se deu devido às funções terem características distintas entre si, o que facilita analisar e inferir de acordo com o comportamento dos mesmos, qual se destaca para variados tipos de problemas. A escolha das funções utilizadas no trabalho, apresentadas na Seção 2.5, se deu de maneira aleatória, porém teve-se o cuidado de selecionar funções que tivessem comportamentos mais distintos, para garantir melhor aferição da análise. Conforme pôde ser conferido na Seção 2, para o desenvolvimento dos algoritmos foi necessário um estudo detalhado do funcionamento dos mesmos, uma vez que, mesmo se tratando de algoritmos evolucionários, os dois algoritmos tem abordagens bastante diferentes.

Após desenvolvidos os algoritmos, buscou-se ajustá-los de modo que houvesse uma configuração genérica que conseguisse resolver a maioria das equações de maneira satisfatória, além de uma determinada quantidade de gerações, comum aos dois algoritmos, para que fosse possível uma melhor comparação entre os mesmos. As configurações propostas para cada algoritmo na Seção 3.2 se mostraram bastante satisfatórias, uma vez que foi possível encontrar o mínimo global da maioria das funções em apenas 50 gerações para os dois.

Na Seção 4 pôde-se avaliar e comparar o desempenho dos algoritmos função por função. A metodologia aplicada ao SGA se mostrou adequada, uma vez que foi possível convergir para o valor mínimo global de todas as funções testadas, embora para as funções Easom e Shubert foi necessário realizar alguns ajustes no algoritmo para alcançar resultados melhores como foi apresentado nas Seções 4.2.7 e 4.2.8. A

metodologia aplicada ao DE também se mostrou adequada, foi possível convergir para o valor mínimo global em todas as funções com ressalvas à função Easom, onde foi necessário ajustar o algoritmo para que fosse possível convergir para o valor mínimo global da função, como pôde ser observado na Seção 4.2.8.

Através da análise estatística dos resultados dos 33 testes aplicados a cada função por algoritmo, detalhado na seção 4.2, permitiu comparar o desempenho do SGA e DE na resolução das funções. Embora os dois algoritmos tenham conseguido convergir para os mínimos globais das funções, o DE se mostrou mais eficaz pois teve resultados melhores em relação ao SGA em todas as funções. Na média aritmética dos resultados da quinquagésima geração nos 33 testes, o DE conseguiu cravar o resultado em 5 das 8 funções testadas, em comparação às 2 do SGA.

5.1 Trabalhos futuros

Pretende-se como trabalhos futuros:

- Realizar estudo da resolução das funções utilizadas neste trabalho utilizando métodos escalares como Elipsoidal, Soma Ponderada, dentre outros;
- Avaliar o desempenho dos métodos escalares na resolução de funções de benchmark restritas e irrestritas;
- Avaliar os métodos desenvolvidos em funções com restrições e com mais de um objetivo.

6. REFERÊNCIAS

COSTA, L. A. A. F. Algoritmos evolucionários em optimização uni e multi-objectivo. 2003.

DA COSTA, Marcos Fabio Nobrega et al. Computação evolutiva para minimização de perdas resistivas em sistemas de distribuição de energia elétrica. 1999.

DE JONG, Kenneth Alan. Analysis of the behavior of a class of genetic adaptive systems. 1975.

FOGEL, Lawrence J.; OWENS, Alvin J.; WALSH, Michael J. Artificial intelligence through simulated evolution. 1966.

GASPAR-CUNHA, António; TAKAHASHI, Ricardo; ANTUNES, Carlos Henggeler. **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

GUIMARÃES, Frederico G.; RAMALHO, Marcelo C. Implementação de um Algoritmo Genético. **Trabalho referente à disciplina "Otimização", junho de** 2001.

GUIMARÃES, Frederico Gadelha. Algoritmos de evolução diferencial para otimização e aprendizado de máquina. In: **IX Congresso Brasileiro de Redes Neurais**. 2009.

HOLLAND, John H. Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. **Ann Arbor, MI: University of Michigan Press**, 1975.

LINDEN, Ricardo. **Algoritmos genéticos (2a edição)**. Brasport, 2008.

MARQUES, Marcel Augustino, "Charles Darwin". Disponível em: <http://www.pucsp.br/pos/cesima/schenberg/alunos/marcelmarques/index.htm>. Acesso em 12 de Março de 2017.

MATHWORKS. MATLAB, The Language of Technical Computing. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em 15 de Abril de 2017.

MICHALEWICZ, Z. "Genetic algorithms + Data Structures = Evolution Programs", 3rd edition, Springer-Verlag, 1996.

OLIVEIRA, Giovana Trindade da Silva et al. Estudo e aplicações da evolução diferencial. 2006.

PACHECO, André. "O algoritmo evolutivo Differential Evolution". Disponível em:<http://www.computacaointeligente.com.br/algoritmos/o-algoritmo-evolutivo-differential-evolution>. Acesso em 03 de Abril de 2017.

PARREIRAS, Roberta Oliveira. **Algoritmos evolucionários e técnicas de tomada de decisão em análise multicritério**. 2006. Tese de Doutorado. Universidade Federal de Minas Gerais.

PRICE, K.; STORN, R. Differential Evolution-a simple and efficient adaptive scheme for global optimization over continuous space. **Technical Report, International Computer Science Institute**, 1995.

R-PROJECT. What is R. Disponível em: <https://www.r-project.org/about.html>. Acesso em 15 de Abril de 2017.

RECHENBERG, Ingo. **Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution**. frommann-holzbog, Stuttgart, 1973.

REZENDE, Solange Oliveira. **Sistemas inteligentes: fundamentos e aplicações**. Editora Manole Ltda, 2003.

SOUZA, RCT. Previsão de séries temporais utilizando rede neural treinada por filtro de Kalman e evolução diferencial. **Pontifícia Universidade Católica do Paraná, Curitiba**, 2008.

SYSWERDA, Gilbert. Uniform crossover in genetic algorthms. In: **Proc. Third International Conference of Genetic Algorithms**. Morgan Kaufmann, 1989.

VON ZUBEN, Fernando J. Computação evolutiva: uma abordagem pragmática. **Faculdade de Engenharia Elétrica e de Computação-Universidade Estadual de Campinas**, 2000.