UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

ENG07053 - TRABALHO DE DIPLOMAÇÃO EM ENGENHARIA QUÍMICA

Comparação Sistemática entre Métodos de Otimização Estocástica

Autor: Paula Scherer Servat

Orientador: Prof. Dr. Marcelo Farenzena

Porto Alegre, novembro de 2020

Sumário

Agradecimentos

Primeiramente, agradeço aos meus pais, Paulo e Maristela, pelo apoio incondicional dedicado a mim, pela compreensão nos meus momentos de ausência, pelo incentivo constante e por todas as oportunidades que me proporcionaram. A eles dedico este trabalho, pois são minha fonte de inspiração.

Agradeço ao meu namorado, Rodrigo, pela compreensão diante das minhas escolhas, pelo suporte e incentivo em todos os momentos de dificuldade e por todo apoio, amor e carinho dedicados a mim neste último ano.

Agradeço às minhas amigas Amanda, Anna, Clara, Moema e Laura e ao meu amigo Robert por todo o apoio e parceria nesses anos de engenharia. Não teria sido tão legal sem vocês.

Por fim, agradeço ao meu orientador, Marcelo, por todo o suporte, orientação e ensinamentos concedidos a mim nesse último ano, pela disponibilidade e paciência e pelo excelente trabalho que realiza como professor da Engenharia Química da UFRGS.

Resumo

Os algoritmos de otimização surgiram do desejo de utilizar os recursos disponíveis da melhor maneira possível e, ao longo dos últimos anos, foram responsáveis por muitos avanços tecnológicos alcançados pela humanidade. Os algoritmos estocásticos, desenvolvidos no final do século *XX*, são uma importante classe de algoritmos de otimização que, ao contrário da classe de algoritmos determinísticos, visam contemplar a aleatoriedade na busca pelo ótimo. Desde sua criação, vem sendo apresentados inúmeros estudos com novos algoritmos, dentre os quais destacam-se os chamados meta heurísticos, que imitam comportamento da natureza. Dessa forma, quando se deseja resolver um problema de otimização, há, atualmente, inúmeras possibilidades de algoritmos para escolha, porém a literatura diverge quanto ao desempenho desses algoritmos, o que dificulta a seleção do algoritmo para resolver o problema. Portanto, este trabalho focou em realizar uma comparação sistemática entre os algoritmos clássicos Genético, Evolução Diferencial e *Particle Swarm Optimization* e algoritmos recentemente publicados*, Bat Algorithm* e *Flower Pollination Algorithm*, realizando uma análise alicerçada em três critérios de avaliação, a robustez (η), a eficiência (γ) e a eficácia (θ), os quais avaliaram os algoritmos em termos de quantidade de problemas resolvidos, consistência em relação ao melhor resultado e consistência em relação ao melhor tempo de convergência, respectivamente. Para isso, dividiu-se a análise em duas etapas, a primeira teve como objetivo realizar o ajuste de parâmetros específicos de cada algoritmo, visto que esses influenciam no seu desempenho, e a segunda realizou a comparação entre os algoritmos a partir do seu melhor desempenho individual observado na etapa um. Os resultados demonstraram que os algoritmos clássicos Evolução Diferencial (ED), Exame de Partículas (PSO) e Algoritmo Genético (AG) apresentaram os melhores desempenhos, em ordem de maior para menor. Os mais recentes, *Bat Algorithm* (BA) e *Flower Pollination Algorithm* (FPA), tiveram desempenho inferior aos clássicos, embora o FPA tenha sido significativamente superior ao BA. Os resultados foram de encontro aos apresentados pelo autor do FPA e BA, mas estão em linha com outros estudos publicados. Com isso, verificou-se que os algoritmos podem ter o seu desempenho influenciado pelos parâmetros escolhidos, pela maneira como foram codificados e pelo tipo de problema utilizado, comprovou-se também a importância de se estabelecer um método estruturado de comparação entre os algoritmos.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

N^*	Número de simulações que convergiram
N_{fes}	Número de avaliações da função objetivo
N_{fes}	Número de avaliações da função objetivo
P_g	Melhor posição do enxame (PSO)
P_i	Melhor posição individual (PSO)
$\bar{T}$	Tempo médio de convergência de um algoritmo
T_j	Tempo de convergência de uma simulação
T_j^*	Melhor tempo de uma simulação
X_G	Representação do enxame de partículas (PSO)
c_1	Parâmetros componente cognitivo (PSO)
c_2	Parâmetro componente social (PSO)
c_k	Restrição k a qual a função objetivo é submetida
f_G	Frequência na geração G (PSO)
$f_{máx}$	Parâmetro de frequência máxima (BA)
$f_{mín}$	Parâmetro de frequência mínima (BA)
r_j	Índice aleatório da população (ED)
$u_{j,G}$	Vetor teste j na geração G (ED)
$v_{j,G}$	Velocidade j na geração G (PSO; BA)
$\bar{x}$	Média das soluções
x^*	Melhor solução encontrada
$x_{j,G}$	Solução j na geração G gerada por um algoritmo
$x_{k,G}$	Solução k na geração G (FPA)
$z_{j,G}$	Vetor mutante j na geração G (ED)
G	Índice de geração de uma população
j	Índice da solução de uma população
$\Gamma(\lambda)$	Função gamma (FPA)

A	Parâmetro Amplitude (BA)
CR	Parâmetro taxa de cruzamento (AG; ED)
D	Dimensão de uma função objetivo
DP	Diversidade
F	Fator de escala (ED)
L	Distribuição de Lévy (FPA)
N	Número total de simulações realizadas
NP	Tamanho da população
NP	Tamanho da população de um algoritmo
TM	Taxa de mutação (AG)
TP	Parâmetro percentual de novos indivíduos na nova população (AG)
TS	Parâmetro número de indivíduos selecionados (AG)
$f(x)$	Função objetivo
k	Índice de solução (FPA)
p	Probabilidade de troca entre a busca local e global (FPA)
r	Parâmetro taxa de pulsos (BA)
s	Tamanho do passo (FPA)
x	Variável de projeto
β	Parâmetro aleatório (BA)
γ	Eficiência
η	Robustez
θ	Eficácia
λ	Parâmetro da função de distribuição de Lévy (FPA)
μ	Distância média do ótimo
ω	Parâmetro de inércia (PSO)
ϵ	Parâmetro aleatório da busca local (FPA)

Lista de Abreviaturas e Siglas

AG	Algoritmo Genético
BA	*Bat Algorithm*
ED	Algoritmo Evolução Diferencial
FPA	*Flower Pollination Algorithn*
PSO	Algoritmo *Particle Swarm Optimization*

1 Introdução

Os problemas de otimização estão presentes nas mais diversas áreas de estudo e permeiam o nosso dia-a-dia, como por exemplo, quando decidimos como fazer o nosso itinerário até o trabalho, ou quando organizamos nosso tempo, estamos sempre buscando o mínimo custo, ou a maior economia, seja ela de tempo, dinheiro, ou energia para realizar as tarefas. Esses são exemplos de problemas simples, que conseguimos resolver mentalmente, no entanto problemas mais complexos como por exemplo a otimização de elevação de petróleo em plataformas ou a determinação da velocidade mínima necessária para que um foguete escape da atração gravitacional da terra são exemplos de problemas muito difíceis de resolver sem a utilização de um *software*. Nesse contexto, surgem os algoritmos de otimização que visam resolver de uma maneira mais fácil e rápida os mais variados problemas.

Esses algoritmos vêm sendo um importante objeto de estudo de pesquisadores e muitos avanços foram alcançados nas últimas décadas. A otimização estocástica surge nesse cenário para resolver problemas complexos que possuam muitas variáveis, funções-objetivo não-diferenciáveis ou um conjunto de restrições fortes e seu principal diferencial é a facilidade para encontrar o ótimo global (MATTOS CAVALCANTE, 2009). Os primeiros algoritmos estocásticos foram desenvolvidos ainda no final do século XX e, desde então, uma ampla gama de novos algoritmos foram propostos, dentre os quais se destacam principalmente aqueles chamados meta heurísticos, que imitam o comportamento da natureza.

Dentre os algoritmos mais antigos destacam-se o Algoritmo Genético, o Algoritmo de Evolução Diferencial e o *Particle Swarm Optimization*. O Algoritmo Genético e o Algoritmo de Evolução Diferencial são algoritmos baseados na teoria de evolução das espécies de Darwin, enquanto o *Particle Swarm Optimization* é um algoritmo com mecanismo de funcionamento baseado em inteligência de enxames. Esses três algoritmos possuem eficácia comprovada e, desde a sua criação, vem sendo amplamente utilizados para a resolução de problemas complexos (SERAPIÃO, 2009).

Nos últimos anos, inúmeros novos algoritmos foram propostos com o intuito de superar a performance oferecida pelos clássicos AG, ED e PSO. Exemplos destes novos algoritmos são o *Bat Algorithm,* que possui mecanismo de funcionamento baseado na inteligência de enxames, e o *Flower Pollination Algorithm* que é baseado no processo de polinização das flores. No entanto, embora o criador do BA e FPA, YANG (2010) e YANG (2012), aponte que ambos algoritmos possuem desempenho muito superior aos clássicos, as buscas na literatura mostram que ainda não há consenso sobre seu desempenho, sendo encontrados poucos estudos que demonstrem sua capacidade de otimização em aplicações para resolução de problemas reais.

Dessa forma, quando se faz necessário realizar a otimização de um problema, há inúmeras possibilidades para a escolha do algoritmo, entretanto não ficam claras quais as vantagens que cada um possui e como se diferenciam em termos de desempenho. Portanto, este trabalho tem como objetivo mensurar o desempenho de cinco diferentes algoritmos, por meio de uma comparação sistemática dos mesmos, de modo a permitir que os algoritmos sejam avaliados em relação a critérios específicos e assim sejam mais facilmente selecionados.

Para atingir o objetivo geral, os seguintes objetivos específicos devem ser atingidos:

1. Levantamento de algoritmos estocásticos e funções de *benchmark* utilizadas para testar seu desempenho;

2. Levantamento de bibliotecas que disponibilizem os algoritmos e funções utilizadas;

3. Definição do melhor conjunto de parâmetros a ser utilizado por cada algoritmo, a fim de compará-los entre si posteriormente;

4. Aplicação dos critérios de avaliação estabelecidos a partir dos resultados obtidos por cada um dos algoritmos.

O presente trabalho é dividido como segue: no capítulo 2 são apresentados conceitos gerais de otimização e dos algoritmos utilizados; no capítulo 3 é apresentada a metodologia de comparação dos algoritmos, trazendo detalhadamente o mecanismo de funcionamento dos mesmos, os parâmetros utilizados, os critérios de avaliação e as funções de benchmark; no capítulo 4 é apresentada a discussão sobre os resultados de desempenho obtidos a partir dos critérios de avaliação; e, por fim, no capítulo 5 são apresentadas as conclusões geradas a partir da análise dos resultados, bem como as sugestões para próximos trabalhos.

2 Revisão Bibliográfica

2.1 Introdução à otimização

A busca pelo resultado ótimo, seja em processos, projetos ou problemas está presente desde o surgimento da humanidade e vem evoluindo com ela. As teorias de otimização surgem nesse cenário para definir métodos de busca pelo ótimo, ou seja, métodos para encontrar a solução que mais se adequa ao objetivo desejado. Como sintetizado por GLAND; KLEIN; THOMAS (2001) a otimização é o uso de métodos específicos para determinar a solução mais econômica e eficiente para um problema ou desenho de um processo. Essa técnica é uma das principais ferramentas quantitativas na tomada de decisão industrial.

Os problemas de otimização estão presentes em inúmeras áreas, como economia, transporte, engenharia, medicina, entre outros. Nesses casos o que geralmente se busca é a minimização dos custos e a maximização da receita, no entanto os problemas de otimização não estão restritos apenas a benefícios monetários. Nas operações de plantas industriais, por exemplo, a otimização gera um desempenho aprimorado da planta, através do aumento de rendimento de produtos, do consumo reduzido de energia, de taxas de processamento mais altas e tempos mais longos entre as paradas. Além disso, surgem benefícios intangíveis a partir da sua utilização, pois é extremamente útil identificar sistematicamente o objetivo, as restrições e os graus de liberdade em um processo ou em uma planta, levando a benefícios como melhoria na qualidade do projeto, resolução de problemas e tomada de decisões de maneira mais rápida (GLAND; KLEIN; THOMAS, 2001).

Para utilizar as ferramentas de otimização primeiramente é necessário identificar um objetivo, uma medida quantitativa do desempenho do sistema em estudo. Esse objetivo pode ser lucro, tempo, energia ou qualquer quantidade ou combinação de quantidades que possa ser representada numericamente. O objetivo depende das características do sistema e essas características são representadas pelas suas variáveis as quais podem ou não conter restrições. O que se busca, então, é encontrar valores das variáveis que minimizam ou maximizam, dependendo do caso, a função que se definiu como objetivo (NOCEDAL; WRIGHT, 1951).

2.1.1 Conceitos básicos

De acordo com REY NARIÑO (2014) os termos do problema de otimização podem ser definidos como:

- Função objetivo ($f(x)$): é a representação matemática do critério de eficiência adotado no problema de otimização. É influenciada pelas variáveis de projeto, conhecidas também como variáveis de controle do problema.
- Variáveis de projeto (x): as variáveis de projeto são os parâmetros variáveis no processo de otimização que definem as características do modelo analisado. Em outras palavras, são as variáveis que se deseja determinar dentro de um intervalo pré-definido e para as quais o problema é resolvido. De forma geral, as variáveis podem adotar qualquer valor, mas isso poderia produzir soluções inviáveis no processo de otimização. Por isso, é necessário introduzir as variáveis do projeto em um intervalo cujo limite inferior e superior delimitem a variação das mesmas. Igualmente importante é definir o tipo de variável, seja ela discreta ou contínua, podendo existir problemas

onde possam ser utilizados ambos os tipos de variáveis. Uma variável contínua é aquela que pode assumir qualquer valor dentro de um intervalo de variação, enquanto uma variável discreta é aquela que pode assumir valores específicos.

- Restrições (c_k): as restrições são equações de igualdade ou de desigualdade e determinam os limites de viabilidade para a solução do problema, expressando uma condição desejável do comportamento do sistema. Geralmente, as restrições estão relacionadas com a geometria, os esforços admissíveis, os recursos disponíveis, os custos envolvidos, dentre outros.
- Espaço de busca: o espaço de busca é a região do domínio que satisfaz às restrições do problema. É também conhecido como domínio viável do problema. É delimitado pelas restrições impostas ao sistema e pelo intervalo de variação das variáveis do projeto. Podem existir "melhores soluções" fora do espaço de busca do problema, porém, tecnicamente, elas são inviáveis pelo fato de violarem as restrições.
- Ponto ótimo: o ponto ótimo é o vetor das variáveis de projeto que minimiza (ou maximiza) a função objetivo e satisfaz as restrições do problema. O valor da função objetivo correspondente a essas variáveis é denominado valor ótimo. Consequentemente, o par formado pelo ponto ótimo e o valor ótimo é chamado de solução ótima, podendo ser local, se for um ponto de mínimo (ou máximo) em relação a uma vizinhança desse ponto, ou global, se for um valor mínimo (ou máximo) em relação a todo o espaço de busca do problema.

Assim, pode-se escrever o problema como:

$$\min_{x \in R^n} f(x) \quad \text{restrito à} \quad c_k(x) = 0 \quad \text{e} \quad c_k(x) \geq 0 \tag{2.1}$$

O problema de otimização descrito pelas Equações 2.2 e 2.3 e ilustrado na Figura 2.1 exemplifica os conceitos apresentados anteriormente. Para este caso, f é a função objetivo, x o vetor de variáveis e c_k as restrições. A figura mostra os contornos da função objetivo, ou seja, o conjunto de pontos para os quais f tem um valor constante. Também ilustra a região viável/factível, que é o conjunto de pontos que satisfazem todas as restrições (a área entre os dois limites da restrição) e o ponto x^* que é a solução do problema.

$$\min(x_1 - 2)^2 + (x_2 - 1)^2 \tag{2.2}$$

$$\text{Restrito à} \quad x_1{}^2 - x_2 \leq 0 \ \text{ e } \ x_1 + x_2 \leq 2 \tag{2.3}$$

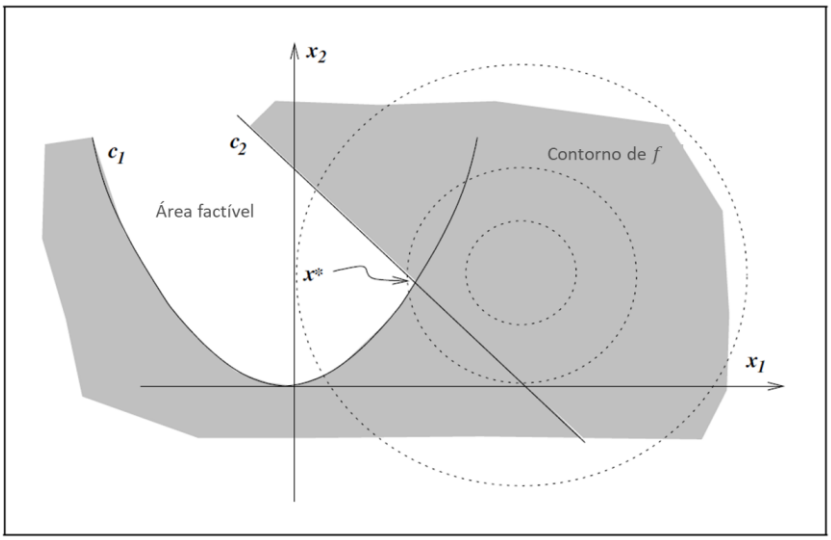


Figura 2.1: Representação Geométrica do Problema 1. Adaptada de NOCEDAL; WRIGHT, (1951).

2.1.2 Formulação do problema

Os modelos matemáticos são uma parte importante do problema de otimização, pois são eles que definem o problema a ser resolvido. Modelos matemáticos são definidos por EYKHOFF (1974) como *"uma representação dos aspectos essenciais de um sistema existente (ou um sistema a ser construído) que apresenta o conhecimento desse sistema em uma forma utilizável".* Na otimização é necessário desenvolver modelos da função objetivo e das restrições de igualdade e desigualdade, sempre levando em conta que um modelo apenas imita a realidade e não pode incorporar todas as características do processo real que está sendo modelado (GLAND; KLEIN; THOMAS, 2001).

Para resolver esses problemas, surgem então os algoritmos de otimização, que podem ser classificados como determinísticos ou estocásticos. Nas Seções 2.2 e 2.3 é apresentada uma análise dos principais pontos relacionados a essas classes de algoritmos, bem como suas diferenças.

2.2 Otimização determinística

A otimização determinística é o ramo clássico da otimização e é fortemente baseada na álgebra linear para obtenção de resultados, pois comumente utiliza o cálculo de gradientes e algumas vezes a Hessiana das variáveis de resposta (CAVAZZUTI, 2003). O método determinístico sempre leva as mesmas respostas a partir das mesmas variáveis de entrada e essa característica acarreta vantagens e desvantagens em relação a sua utilização.

A principal vantagem desse método é a sua rápida convergência em comparação com uma convergência mais lenta quando utilizado o método de otimização estocástica. De acordo com CAVAZZUTI (2003) isso se deve ao fato de que a otimização determinística necessita de um número menor de avaliações de função, para chegar à solução. A avaliação de uma função envolve um experimento ou simulação a ser realizada, portanto, o número de estimativas requeridas por um algoritmo de otimização para chegar a uma solução é uma medida do tempo requerido pelo próprio processo de otimização. Por estar baseado em uma formulação matemática rigorosa que não envolve elementos estocásticos, os resultados de um processo de otimização determinística são inequívocos e replicáveis.

No entanto, apesar de os problemas convergirem rapidamente com esse método, caso o algoritmo determinístico utilizado não seja de busca global, a solução ótima eventualmente encontrada poderia ser um ótimo local e não global (SALATTA; DOS SANTOS, 2017). Além disso, embora haja algoritmos determinísticos para otimização global, os principais estudos realizados neste tema ainda estão focados na classe estocástica, pois os determinísticos, mesmo que de busca global, apresentam desempenho inferior aos estocásticos em termos de tempo de convergência (LIBERTI; KUCHERENKO, 2005).

2.3 Otimização estocástica

A probabilidade e estatística há muito tempo nos ensinou que o futuro não pode ser perfeitamente previsto e que, ao contrário disso, deve ser considerado aleatório ou incerto (BIRGE; LOUVEAUX, 2006). Nesse sentido, surge a programação estocástica, objeto de estudo deste trabalho, que tem como principal diferencial contemplar a aleatoriedade no processo de busca, diferentemente dos algoritmos determinísticos que utilizam dados fornecidos com antecedência e levam aos mesmos resultados a partir das mesmas variáveis de entrada. Com isso, a otimização estocástica permite encontrar o ótimo global mais facilmente para problemas que podem ter uma resolução muito complexa quando utilizado o método determinístico, visto que para esse método simplesmente encontrar todos os ótimos locais já é quase impraticável (BIRGE; LOUVEAUX, 2006).

Os estudos sobre o tema otimização estocástica teve início nos anos 1950, no entanto o uso de algoritmos estocásticos se intensificou somente a partir dos anos 1990 com o avanço da tecnologia. Esses algoritmos são geralmente utilizados para resolver problemas que apresentam algumas características específicas como por exemplo, muitas variáveis, funções-objetivo não-diferenciáveis e, possivelmente, um conjunto de restrições impostas pelo problema que dificultem a otimização. Problemas com essas características são comuns em situações reais e frequentemente não podem ser resolvidos de maneira satisfatória - qualidade da solução e tempo de processamento razoáveis - por métodos determinísticos (MATTOS CAVALCANTE, 2009).

Os algoritmos de otimização estocástica oferecem uma alternativa às dificuldades encontradas pelo método determinístico. Por exemplo, algoritmos estocásticos em lugar de encontrar e comparar todos os ótimos locais para identificar o ótimo global, amostram e avaliam as soluções candidatas e aproximam-nas ao ótimo global. Isso significa que a solução final obtida a partir dos métodos de otimização global estocástica converge para o ótimo global em um sentido probabilístico (MONTAZ ALI; KHOMPATRAPORN; ZABINSKY, 2005).

Em comparação com os métodos determinísticos, a otimização estocástica apresenta a vantagem de ser menos complicada matematicamente e de contemplar a aleatoriedade no processo de busca, entretanto a sua maior desvantagem está na convergência mais lenta para a solução ideal (CAVAZZUTI, 2003).

2.4 Algoritmos para otimização estocástica

Como mencionado, este trabalho busca estudar e analisar uma série de algoritmos estocásticos e assim concluir quais são mais adequados de acordo com os critérios que serão apresentados no Capítulo 3. Um maior detalhamento dos mesmos será apresentado também no Capítulo 3.

2.4.1 Algoritmo Genético (AG)

Os algoritmos genéticos são uma classe de algoritmos evolutivos que propõem uma solução para problemas de otimização inspirada nos conceitos da teoria de seleção natural e genética, apresentada por Charles Darwin. A teoria propõe que quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes. O primeiro pesquisador a introduzir o conceito de algoritmo genético foi John Holland em 1975, entretanto, esse método só se popularizou por um de seus alunos, David Goldberg (LACERDA; CARVALHO, 1999).

Analogamente à teoria de seleção natural para os seres vivos, a abordagem dos algoritmos genéticos gera uma população (conjunto de possíveis soluções para o problema) onde cada indivíduo na população, denominado cromossomo, representa uma solução do problema codificado. Os indivíduos são compostos por genes que são os valores da solução. Estes genes podem sofrer alterações de uma geração para outra, através da aplicação de operadores de seleção, cruzamento (crossover), mutação e elitismo (FALCONE, 2004).

De acordo com CÁRDENAS (2007), para encontrar a solução desejada, a primeira população é gerada de forma aleatória para garantir que haja diversidade nas soluções e cada indivíduo é avaliado de acordo com a função *fitness* que mede a qualidade da solução analisando o valor obtido na função objetivo. A partir daí, o método vai utilizar o operador seleção para selecionar os melhores indivíduos para que estes, posteriormente, passem pelo cruzamento, operador que cruza o material genético de dois indivíduos aleatórios, pela mutação, que seleciona indivíduos aleatórios e os modifica e pelo elitismo, que insere os melhores indivíduos da população anterior na nova população sem altera-los e assim cria-se uma nova população com soluções melhores que a anterior.

Os algoritmos genéticos surgiram há mais de 40 anos e continuam hoje representando uma ótima alternativa para problemas de otimização em diversas áreas, por isso ainda são amplamente estudados e melhorados. Algumas de suas aplicações são descritas como segue:

- MAGNANI (2019) otimizou o gerenciamento dos lotes de Nafta recebidos por uma indústria petroquímica através de operações de blending para viabilizar o recebimento de lotes que apresentam contaminantes e o enquadramento destes nos limites de processamento das unidades. Assim foi possível flexibilizar a compra de matérias-primas, diminuindo os gastos e aumentando o lucro da empresa.
- METAWA; HASSAN; ELHOSENY (2017), a partir da utilização do AG, forneceram uma estrutura para otimizar a construção da carteira de empréstimos de um banco, maximizando o lucro e minimizando a probabilidade de inadimplência.

2.4.2 Evolução Diferencial (ED)

O algoritmo Evolução Diferencial (ED), foi proposto por STORN; PRICE (1997), como uma alternativa aos algoritmos genéticos. Seu funcionamento é muito similar ao AG, pois o método também é baseado em criar populações as quais representam o conjunto de soluções possíveis para o problema que devem ‘evoluir’ a partir dos pontos fortes da

geração anterior, ou seja, a nova população deve estar sempre mais próxima do ótimo global (HEGERTY; HUNG; KASPRAK, 2009).

O método utilizado pelo ED é eficaz na obtenção da solução mais adequada para problemas complexos devido à sua estratégia de competição rigorosa. O algoritmo faz a seleção da solução, ou soluções, para o problema a partir da comparação dessas com uma solução padrão pré-determinada, o vetor mutante. A estratégia do ED é dita rigorosa, pois até mesmo o vetor mutante pode ser substituído caso seja encontrado um candidato de melhor desempenho (ESSIET; SUN; WANG, 2019).

De acordo com RAMALHO (2016) este algoritmo possibilita a escolha de diferentes estratégias para a evolução diferencial. Essas estratégias podem ser classificadas de acordo com o tipo de indivíduo (vetor alvo) a ser modificado, o número de indivíduos utilizados para gerar uma perturbação e o tipo de cruzamento a ser utilizado. As etapas de mutação e cruzamento são as mais críticas para esse processo pois há inúmeras combinações possíveis de vetores lineares que podem ser utilizadas. Assim, o algoritmo ED continua sendo objeto de interesse e tem seu desempenho constantemente melhorado, como é possível verificar nas aplicações apresentadas:

- SOUZA et al. (2016) otimizou o planejamento de redes locais de internet sem fio por meio da definição do posicionamento e o balanceamento da carga entre pontos de acesso da rede, de modo a elevar o nível de sinal nas estações clientes, maximizar o balanceamento de carga dos pontos de acesso e minimizar as interferências na rede.
- DE OLIVEIRA VASCONCELOS (2016) otimizou a configuração de sistemas de distribuição de energia elétrica para que houvesse uma rede com mínimas perdas e número mínimo de cargas desenergizadas, reduzindo assim as perdas de energia por efeito joule e falhas nas redes devido a fatores externos.

2.4.3 Particle Swarm Optimization (PSO)

O algoritmo *Particle Swarm Optimization* (PSO), ou otimização por enxame de partículas em português, pertence a uma classe de algoritmos chamada de Inteligência de Enxames. Essa classe, também referenciada como Inteligência de Colônias ou Inteligência Coletiva, é um conjunto de técnicas baseadas no comportamento coletivo de sistemas auto-organizados, distribuídos, autônomos, flexíveis e dinâmicos (SERAPIÃO, 2009b).

O PSO foi proposto por Kennedy e Eberhart em 1995 e surgiu da implementação de uma analogia ao comportamento social da interação entre indivíduos (partícula) de um grupo (enxame) a partir da observação de bandos de pássaros e cardume de peixes em busca de alimento (MEDEIROS, 2005). Cada indivíduo de uma população possui sua própria experiência e é capaz de estimar a qualidade dessa experiência e, como os indivíduos são sociais, eles também possuem conhecimento sobre como seus vizinhos comportam-se. Esses dois tipos de informação correspondem à aprendizagem individual (cognitiva) e à transmissão cultural (social), respectivamente. Dessa forma as chances de um determinado indivíduo tomar uma certa decisão será uma função de seu desempenho individual no passado e do desempenho de alguns de seus vizinhos (SERAPIÃO, 2009b). Nesse sentido, pode-se dizer que a principal força motriz do PSO é a interação social e a troca de conhecimento sobre o espaço de busca (PESSIN; OSÓRIO, 2009).

Nesse método, de acordo com PESSIN; OSÓRIO (2009), as partículas representam as possíveis soluções e o enxame é o conjunto de todas as possíveis soluções. Inicialmente, o primeiro enxame de partículas é gerado de maneira aleatória e, a partir disso, as partículas vão interagindo entre si. A cada interação serão atualizadas, seguindo dois melhores valores: o melhor *fitness* da população e o melhor *fitness* encontrado pela partícula (considerando suas gerações passadas).

O PSO é tido como uma abordagem inovadora e possui inúmeras aplicações como as apresentadas:

- CUI et al. (2020) otimizou a produção de carvão verde que tem como requisitos para ser considerado verde o equilíbrio entre os benefícios econômicos, energéticos e ecológicos da sua produção. Para equilibrar esses fatores é proposto um modelo de otimização de multiobjetivos.
- EBADIFARD; BABAMIR (2017) otimizaram o provisionamento dinâmico de recursos sob demanda do processo de scheduling de tarefas de computação em nuvem de forma a aumentar a utilização e o desempenho de recursos, reduzir o tempo de resposta e manter todo o sistema equilibrado.

2.4.4 Bat Algorithm (BA)

O *Bat Algorithm*, ou Algoritmo do Morcego em português, foi desenvolvido por YANG (2010) e é inspirado no mecanismo de eco localização dos morcegos. Este mecanismo permite que, por meio da emissão de um som de alta frequência e inaudível ao ouvido humano, o morcego consiga detectar presas, evitar obstáculos e localizar suas fendas no escuro. Durante a busca por presas, os morcegos emitem pulsos curtos, entretanto, quando uma presa potencial se aproxima, suas taxas de emissão de pulso aumentam e a frequência é ajustada. O aumento da frequência juntamente com a aceleração da emissão de impulsos encurta o comprimento de onda do som emitido e, assim, aumenta a precisão da detecção (YANG; GANDOMI, 2013).

Para implementar o algoritmo GONZÁLEZ et al. (2010) propõe algumas simplificações como por exemplo:

- Todos os morcegos usam a eco localização e 'sabem' a diferença entre comida / presa e barreiras de fundo;
- Os morcegos voam aleatoriamente com velocidade $\mathrm{v_i}$ na posição $\mathrm{x_i}$ com uma frequência fixa f_{min}, variando o comprimento de onda λ e a amplitude da onda A_o para procurar uma presa e podem ajustar automaticamente o comprimento de onda (ou frequência) dos pulsos emitidos, bem como a taxa de emissão de pulso, dependendo da proximidade de seu alvo;
- Embora a amplitude da onda emitida possa variar, presume-se que ela varia de um A_o grande (positivo) a um valor constante mínimo A_{min}.

Dessa forma, as posições (x) são o conjunto de soluções possíveis para o problema e, uma vez que a melhor solução ($\mathrm{x^*}$) é localizada, depois de ter sido comparada com todas as soluções, a velocidade e o comprimento de onda serão ajustados. O range de frequência ($\mathrm{f_{mín}}$, $\mathrm{f_{máx}}$) depende do tamanho do domínio do problema de interesse.

O autor, GONZÁLEZ et al. (2010), considera que a atualização das velocidades e posições dos morcegos são similares ao procedimento na otimização de enxame de partículas

padrão e que o BA pode ser considerado como uma combinação balanceada da otimização de enxame de partículas padrão e a busca local intensiva controlada pelo volume e taxa de pulso.

O algoritmo é uma proposta recente de meta heurística para busca local e global, no entanto já vem sendo utilizado para algumas aplicações, como podemos ver nos exemplos:

- ADARSH et al. (2016) resolveu o problema de Economic Dispatch, utilizando o Chaotic Bat Algorithm, uma adaptação do algoritmo original (BA). A otimização do problema buscou atingir valores ótimos de geração de energia de modo a minimizar os custos e satisfazer as restrições estabelecidas.
- BAHMANI-FIROUZI; AZIZIPANAH-ABARGHOOEE (2014) determinaram o tamanho ótimo para uma bateria de armazenamento de energia de fontes renováveis, baseado nos custos de produção e armazenamento.

2.4.5 Flower Pollination Algorithm (FPA)

O algoritmo *Flower Pollination*, ou Polinização de Flores em português é um recente algoritmo meta heurístico proposto por YANG (2012). O algoritmo, como o nome já diz, é inspirado no mecanismo de polinização das flores. As flores dominam a paisagem desde o período Cretáceo e esse domínio se deve ao seu mecanismo de polinização, que normalmente está associado à transferência de pólen. Essa transferência geralmente depende de polinizadores como insetos, pássaros, morcegos e outros animais. Dessa forma, algumas flores e insetos evoluíram para uma parceria flor-polinizadora muito especializada (YANG, 2012).

O mecanismo de polinização pode ser realizado de três formas: por polinização cruzada abiótica, por autopolinização ou por polinização biótica. A polinização cruzada abiótica, ou alogamia, é a polinização que ocorre a partir do pólen de uma flor de uma planta diferente, enquanto a autopolinização é a fertilização de uma flor a partir do pólen da mesma flor ou de flores diferentes da mesma planta. Já a polinização biótica é a polinização que depende de animais para ocorrer e é considerada a polinização global, já que os animais podem percorrer longas distâncias para realizar a polinização (DURAND-LOSE; JONOSKA, 2012).

Para implementação do algoritmo YANG (2012) propôs algumas simplificações:

- A polinização biótica é considerada um processo de polinização global, enquanto a abiótica e autopolinização são consideradas polinização local.
- A constância da flor pode ser considerada como a probabilidade de reprodução é proporcional à semelhança das duas flores envolvidas.
- A ocorrência de polinização local ou global é controlada por uma probabilidade de alternância $p \in [0,1]$. A probabilidade está em função de fatores como a proximidade física e o vento, a polinização local pode ter uma fração p significativa devido a isso.

O FPA possui então dois passos principais, a polinização global (busca global) e a local. Na polinização global o polinizador que tiver sucesso será o mais adequado, ou seja, será a solução mais aderente para o problema. No passo de polinização local (busca local) são considerados pólens de diferentes flores da mesma planta que cruzam e geram um novo pólen, ou seja, duas soluções anteriores são consideradas para gerar a nova solução. Em

ambos os casos se as soluções recém-geradas forem melhores em termos de minimização da função objetivo, a matriz de soluções é atualizada (BEKDAŞ; NIGDELI; YANG, 2015).

Assim como o *Bat Algorithm*, o FPA é um algoritmo recente que vendo sendo estudado e utilizado para algumas aplicações:

- BEKDAŞ; NIGDELI; YANG (2015) utilizou o FPA para minimizar o peso das estruturas de treliça, incluindo variáveis de projeto de dimensionamento.
- ABDELAZIZ; ALI; ABD ELAZIM (2016) otimizou um problema de Economic Load Dispatch que combinava a alocação da carga necessária entre as unidades de geração disponíveis, de modo que o custo total de geração seja minimizado e a minimização dos impactos ambientais por emissões de poluentes em usinas de energia fóssil.

3 Metodologia

Este capítulo contém a descrição detalhada da metodologia de avaliação de desempenho dos algoritmos, do seu mecanismo de funcionamento, bem como seus parâmetros, funções de *benchmark* e critérios de avaliação.

3.1 Metodologia de avaliação

Para que fosse possível comparar o desempenho dos algoritmos selecionados, cada um deles foi testado em relação às funções de *benchmark* listadas na seção 3.3. Para realizar as simulações utilizou-se a biblioteca *NiaPy* versão 2.0.0rc2, disponibilizada pelo site *PyPi*, que forneceu tanto os algoritmos quanto as funções. Os algoritmos foram implementados no *IDE Spyder* na linguagem *Python.*

Como os algoritmos possuem parâmetros específicos que precisam ser ajustados para que se obtenha um melhor desempenho e não há na literatura um consenso sobre os valores destes parâmetros, dividiram-se as análises em duas etapas. Na primeira foram realizadas simulações afim de encontrar o melhor conjunto de parâmetros para cada um dos algoritmos e na segunda os algoritmos foram comparados entre si por meio de simulações que mantiveram o conjunto de parâmetros específicos, definidos na etapa 1, e o número de gerações fixos, variando apenas o tamanho da população e as funções de *benchmark* utilizadas. As atividades realizadas em cada uma das etapas de análise são descritas na Figura 3.1.

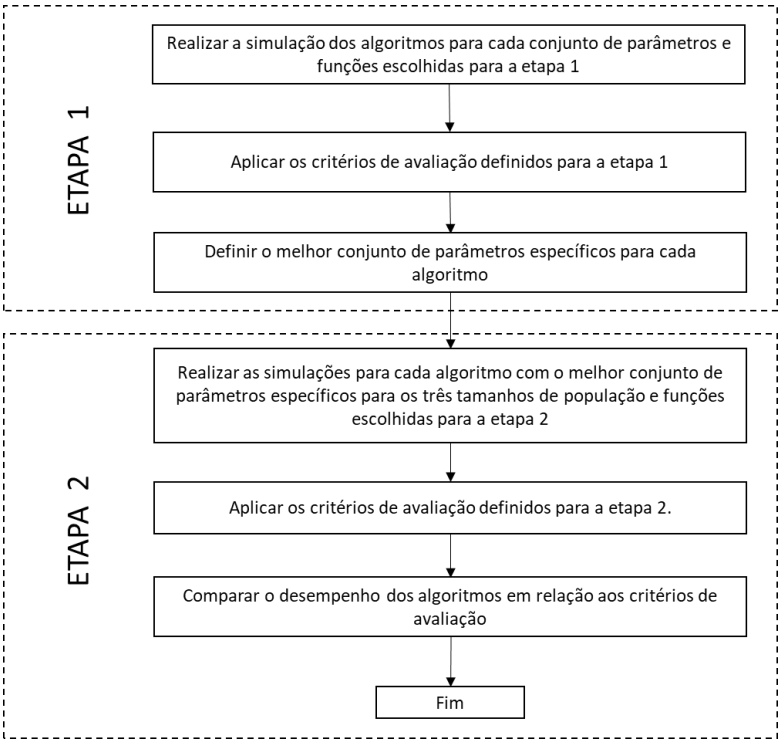


Figura 3.1: Atividades realizadas para comparação dos algoritmos

Dessa forma, na etapa 1, buscou-se encontrar o melhor conjunto de parâmetros para cada um dos algoritmos realizando 50 simulações com cada um dos cinco conjuntos de parâmetros específicos para as três funções de *benchmark* escolhidas, *Ackley, Rosenbrock e Schwefel*, ilustradas na Tabela 3.7. Os parâmetros tamanho da população (NP) e número de avaliações da função objetivo ($\mathrm{N_{fes}}$) foram mantidos fixos em 30 e 10000, respectivamente. Após realizadas as simulações, os resultados foram avaliados a partir dos

critérios distância média do ótimo (μ) e robustez (η), descritos na Seção 3.4. A escolha do melhor conjunto a ser utilizado nas análises da etapa 2 levou em consideração, prioritariamente, o critério de robustez (η), pois esse traduz a capacidade otimização do algoritmo, e utilizou o critério distância média do ótimo (μ) como critério de desempate. Os conjuntos de parâmetros específicos utilizados na etapa 1 para cada algoritmo estão descritos na Seção 3.2.

Na etapa 2, os algoritmos tiveram seu desempenho testado em relação à cinco funções de *benchmark,* mantendo fixo o seu melhor conjunto de parâmetros específicos e variando o tamanho da população. Para isso, foram realizadas 100 simulações para os 3 tamanhos de população, descritos na Tabela 3.1, com as funções de *benchmark Ackley, Rosenbrock, Salomon, Schwefel e Alpine,* ilustradas na Tabela 3.7.

Tabela 3.1: Variação do parâmetro tamanho da população para a etapa 2 de análise.

Tamanho da população (NP)
10
50
100

Uma vez realizadas as simulações da etapa 2, calculou-se a distância média do ótimo (μ) e o tempo médio de otimização ($\bar{T}$) para cada algoritmo e, a partir disso, calcularam-se então os critérios de robustez (η), eficiência (γ), eficácia (θ) e diversidade (D).

3.2 Algoritmos utilizados

3.2.1 Algoritmo Genético (AG)

O Algoritmo Genético é uma versão computacional da seleção natural aplicada à otimização e tem seu mecanismo de funcionamento descrito pela Figura 3.2.

Figura 3.2: Fluxograma do Algoritmo Genético

Desta forma, descrevem-se brevemente seus operadores como segue:

Representação dos Indivíduos

Uma das primeiras atividades a ser realizada na implementação do AG é a definição da maneira de representação das características dos cromossomos para descrever o indivíduo (solução). Esta definição determina como o problema será estruturado para a otimização e quais operadores genéticos poderão ser utilizados. Geralmente as representações cromossômicas se dão por representação binária (0-1), por ponto flutuante ou por inteiros e quantificam as qualidades necessárias às características das soluções.

A definição do tipo de representação a ser utilizado varia de acordo com as necessidades, características do problema a ser resolvido e a natureza das variáveis. Neste trabalho foi utilizada a representação de ponto flutuante que tem como principais características uma maior precisão, consistência entre os resultados e velocidade de processamento.

Geração da população inicial

Neste trabalho a população inicial foi gerada aleatoriamente assumindo uma distribuição uniforme e um tamanho pré-definido NP. O tamanho da população influencia diretamente na capacidade de o algoritmo encontrar a solução ideal. Populações muito pequenas podem fazer com o que o algoritmo não encontre a solução ideal e populações muito grandes podem fazer com que o tempo de busca e o custo computacional sejam muito altos.

Após gerada a primeira população, as populações subsequentes serão geradas a partir dos operadores genéticos (seleção, mutação, cruzamento e elitismo) para que, progressivamente, possam obter soluções melhores do que as das populações anteriores.

Avaliação da solução (*fitness*)

Uma vez gerada uma população, esta é submetida a uma avaliação para medir a qualidade das soluções geradas. O mecanismo de avaliação utilizado neste trabalho é a simples comparação entre os indivíduos, onde os mais bem avaliados serão aqueles que geram o menor valor para a função objetivo. Uma vez avaliados, os indivíduos são organizados em ordem de aptidão.

Operadores Genéticos

Os operadores genéticos são utilizados com o objetivo de criar populações cada vez melhores. Há uma ampla gama de operadores genéticos e, neste trabalho, os operadores implementados serão o de seleção, cruzamento, mutação e elitismo.

O primeiro operador aplicado é o de seleção que, de forma probabilística, tem como objetivo escolher os indivíduos da população atual para o desenvolvimento da geração subsequente. O método escolhido para o algoritmo utilizado neste trabalho é o *Tournament Selection*, que realiza um "torneio" entre um número de indivíduos *TS,* selecionados aleatoriamente entre a população, organiza-os do melhor para o pior em relação a sua aptidão e seleciona os dois melhores indivíduos para passarem pelos

operadores de cruzamento e mutação. O número de indivíduos selecionados na população (TS) é um dos parâmetros a ser escolhido no momento de implementação do algoritmo.

O operador de cruzamento, ou *crossover,* se dá a partir da recombinação de 2 indivíduos pais gerados pela seleção o que acarreta a geração de 2 novos indivíduos (filhos). O cruzamento pode ocorrer de diversas formas a depender do tipo de problema. No presente trabalho este operador recombina uma parcela dos indivíduos pais resultantes do operador de seleção. Para isso é necessário definir a parcela dos indivíduos a ser recombinada por meio do parâmetro CR no momento de implementação.

A mutação é o operador que altera aleatoriamente os indivíduos da população. Para esse operador é possível utilizar diferentes mecanismos. Neste trabalho, a mutação pode ou não ocorrer e isso dependerá do parâmetro taxa de mutação (TM) definido. Uma vez que a mutação seja realizada, os indivíduos modificados são os indivíduos filhos gerados pelo cruzamento.

O operador elitismo utilizado visa manter os melhores indivíduos da população anterior na população subsequente. Para isso manteve-se fixo um percentual da população anterior e substituiu-se o restante dos indivíduos antigos – indivíduos com pior avaliação - pelos melhores indivíduos gerados pelos operadores de seleção, cruzamento e mutação e assim formou-se a nova população. Logo, a nova população será composta pelos indivíduos originais, pelos indivíduos mais bem avaliados, pelos indivíduos que sofreram cruzamento e, por fim, por aqueles que sofreram a mutação. A soma dos últimos representará TP% da população. Os operadores genéticos serão aplicados até que o critério de parada seja atingido. Neste caso o critério de parada definido é um número pré-definido de gerações ($\mathrm{N_{fes}}$).

Parâmetros do AG

Como mencionado na seção 3.1, não há consenso sobre os valores dos parâmetros específicos para os algoritmos e, para o caso do algoritmo genético, esses valores divergem ainda mais, visto que há inúmeras maneiras de implementá-lo. Dessa forma, testaram-se cinco diferentes conjuntos de parâmetros específicos, descritos na Tabela 3.2. No entanto, apesar de não haver consenso sobre os valores dos parâmetros estes seguem algumas premissas: devem-se manter os valores dos parâmetros taxa de mutação, taxa de crossover e indivíduos novos na próxima geração entre 0 e 1 (0 e 100%). A partir de testes preliminares, percebeu-se que é necessário variar o valor do parâmetro Indivíduos selecionados do operador de seleção conforme o tamanho da população, mantendo-o em 1/3 do tamanho da população.

Tabela 3.2: Parâmetros específicos utilizados para o Algoritmo Genético.

Algoritmo	**Parâmetros específicos -**			
AG	**Elitismo - Indivíduos novos na próxima população (TP)**	**Seleção -Indivíduos selecionados (TS)**	**Mutação - Taxa de mutação (TM)**	**Cruzamento - Taxa de crossover (CR)**
Conjunto 1	100%	10	0,1	0,65
Conjunto 2	25%	10	0,1	0,8
Conjunto 3	90%	10	0,05	0,8
Conjunto 4	33%	10	0,1	0,7
Conjunto 5	100%	10	0,1	0,85

3.2.2 Algoritmo Evolução Diferencial (ED)

O mecanismo de funcionamento do algoritmo de Evolução Diferencial (ED) utilizado neste trabalho pode ser definido conforme os passos que seguem na Figura 3.3.

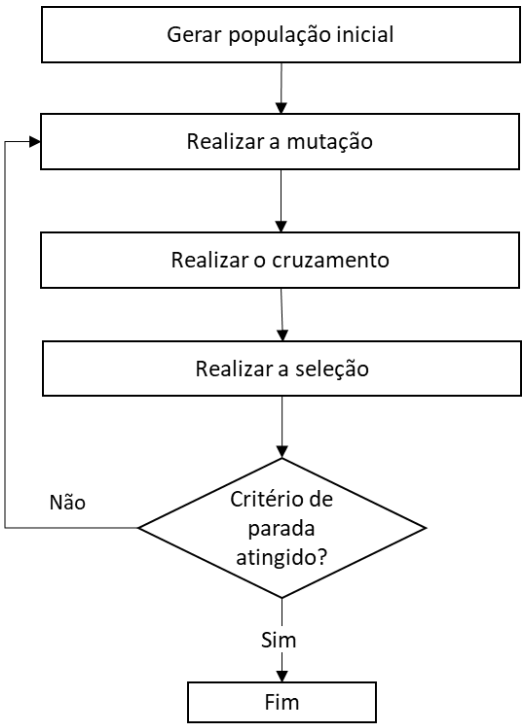


Figura 3.3: Fluxograma do Algoritmo Evolução Diferencial

A estratégia do ED, assim como algoritmo genético, baseia-se na utilização de operadores genéticos para criar populações cada vez melhores, porém o mecanismo de funcionamento desses operadores difere, conforme descrito a seguir.

Geração da População

A população inicial no ED, assim como para os outros algoritmos implementados neste trabalho, é gerada aleatoriamente, assumindo uma distribuição de probabilidade uniforme e um tamanho pré-definido NP. Após a geração da população o ED gera novos indivíduos (soluções) para as novas gerações a partir dos operadores mutação, cruzamento e seleção. Os vetores solução terão a formato da equação 3.1.

$$\mathrm{X_{j,G}} = \left[\mathrm{x_{1,G}}, \mathrm{x_{2,G}}, \ldots \mathrm{x_{j,G}}\right] \qquad \mathrm{j} = 1, 2, \ldots, \mathrm{NP} \tag{3.1}$$

onde G é a geração em questão e j o índice da população.

Mutação

O objetivo da mutação é expandir o espaço de busca e ela pode ser realizada de diversas maneiras. O mecanismo de mutação utilizado neste trabalho segue a equação 3.2, onde para cada vetor alvo $x_{j,G}$ um vetor mutante é gerado. O vetor mutante ($z_{j,G+1}$) tem como objetivo servir para uma posterior comparação com o vetor alvo em termos de aptidão da solução.

$$z_{j,G+1} = x_{r_1,G} + F\left(x_{r_2,G} - x_{r_3,G}\right), \quad r_1 \neq r_2 \neq r_3 \text{ (3.2)}$$

onde os índices r_1, r_2 *e* r_3 dados por $rn(j) \in \{1, 2, \ldots NP\}$ são aleatórios, inteiros e diferentes entre si e representam soluções da população. O vetor mutante é então a soma de um vetor alvo com a diferença entre dois outros vetores alvo, com uma amplificação controlada pelo parâmetro F que deve ser maior que zero. A partir da geração do vetor mutante o ED passa então para a operação de cruzamento.

Cruzamento

O operador de cruzamento é aplicado para garantir a diversidade das soluções. O seu funcionamento mescla os parâmetros do vetor mutante ($z_{j,G+1}$) e do vetor alvo ($x_{j,G}$) e assim gera o vetor teste ($u_{j,G+1}$).

$$u_{j,G+1} = \begin{cases} z_{j,G+1}, & se\ rand_j \leq CR\ ou\ j = rn(j) \\ x_{j,G}, & se\ rand_j > CR\ ou\ j \neq rn(j) \end{cases} \quad (3.3)$$

Nesta equação, j $= 1, 2, 3 \ldots\ldots \mathrm{NP}$, $rand_j$ é um número aleatório entre 0 e 1, CR é a taxa de cruzamento que deve ser definida na implementação do algoritmo e j é o índice aleatório $\mathrm{rn(j)} \epsilon\ (1, 2, 3, \ldots \mathrm{NP})$ das soluções que garante que $u_{j,G+1}$ tenha pelo menos um elemento $z_{j,G+1}$. Em linhas gerais, o vetor teste será igual ao vetor mutante se o número aleatoriamente gerado for menor do que a taxa CR definida ou se o índice da solução for igual as índice aleatório $\mathrm{rn(j)}$, caso contrário, o vetor teste será igual ao vetor alvo.

Seleção

A seleção é a operação que decide se o vetor teste deve ou não estar na próxima geração como denota a Equação 3.4. Para isso, o vetor $u_{j,G+1}$ irá competir com o vetor alvo $x_{j,G}$ e o vencedor dessa competição será o vetor que gera o menor valor da função objetivo dado por $x_{j,G+1}$. Ou seja, se o vetor teste gerar uma solução melhor para o problema do que o vetor alvo, ele estará presente na próxima geração, do contrário, o vetor alvo continuará o mesmo da população inicial.

$$x_{j,G+1} = \begin{cases} u_{j,G+1}, & f\left(u_{j,G+1}\right) < f\left(x_{j,G}\right) \\ x_{j,G}, & f\left(u_{j,G+1}\right) \geq f\left(x_{j,G}\right) \end{cases} \quad (3.4)$$

Critério de Parada

Os operadores de Mutação, Cruzamento e Seleção devem ser repetidos até que o critério de parada do algoritmo seja atingido. Neste trabalho o critério de parada adotado foi um número pré-definido de gerações (N_{fes}).

Parâmetros do ED

Os parâmetros específicos do ED, F e CR, podem ter valores entre 0 e 2 e 0 e 1, respectivamente. Como não há consenso sobre a escolha dos valores para estes parâmetros na literatura, foram testados cinco conjuntos de parâmetros específicos, descritos na Tabela 3.3.

Tabela 3.3: Parâmetros específicos utilizados para o algoritmo Evolução Diferencial.

Algoritmo	**Parâmetros**	
ED	**Mutação - Fator de escala (F)**	**Cruzamento - Taxa de *crossover* (CR)**
Conjunto 1	0,3	0,35
Conjunto 2	0,4	0,65
Conjunto 3	0,3	0,4
Conjunto 4	0,5	0,65
Conjunto 5	0,3	0,3

3.2.3 Algoritmo Particle Swarm Optimization (PSO)

O PSO, algoritmo baseado na inteligência de enxames, utilizado neste trabalho tem mecanismo de funcionamento conforme o mostrado na Figura 4.

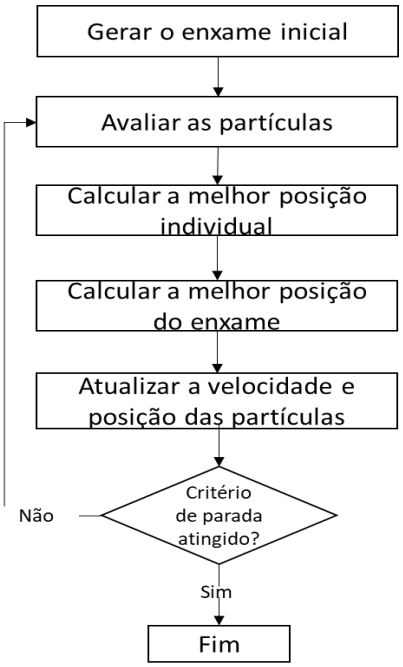


Figura 3.4: Fluxograma do algoritmo *Particle Swarm Optimization*

Neste algoritmo, o enxame de partículas possui NP partículas, ou seja, NP é o tamanho total da população. As soluções potenciais são representadas pela posição de cada uma das partículas (x_j) em um espaço de dimensão D com velocidade v_j. O movimento das partículas pode ser alterado por três principais razões: manter sua inércia, alterar a condição de acordo com a sua própria posição mais otimista ou alterar a condição de acordo com a posição mais otimista do enxame.

<u>Geração da população</u>

A população inicial do algoritmo é gerada aleatoriamente, assumindo uma distribuição de probabilidade uniforme e tem seu tamanho (NP) como um dos parâmetros a ser definido. Cada partícula do enxame representa uma possível solução para o problema e terá seus valores de posição (x_j) e velocidade (v_j) alterados para que o enxame se movimente em direção à solução ótima.

Avaliação da população

As partículas do enxame são avaliadas em relação à minimização da função objetivo. Considerando que a população terá tamanho NP e velocidade v e será representada por $X_G = (x_{Gj}, x_{Gj+1}, \dots, x_{GNP})$, onde X é a representação de todas as soluções do enxame, G a geração e j $=1, 2, 3 \dots$ NP o índice da partícula. As partículas do enxame terão a sua posição comparada entre si e assim será calculada a melhor posição individual, P_i, e a melhor posição do enxame, P_g, (que será a melhor posição dentre todos os indivíduos). A partir disso, a posição da partícula e sua velocidade são atualizadas. Para que este mecanismo funcione, as partículas manterão gravadas as suas melhores soluções (posições) individuais, representadas por $P_i = (p_{ij}, p_{ij+2,} \dots, p_{iNP})$, e as melhores soluções do enxame, $P_g = (p_{gj}, p_{gj+1}, \dots, p_{gNP})$.

Desta forma, dada a função objetivo f, a posição individual da partícula é atualizada para a dimensão D conforme pode ser observado na Equação 3.5.

$$\mathrm{p}^{\mathrm{D}}_{\mathrm{ij,\,G+1}} = \begin{cases} \mathrm{x}^{\mathrm{D}}_{\mathrm{j,\,G+1}}, & \text{se } \mathrm{f}(\mathrm{x}^{\mathrm{D}}_{\mathrm{j,\,G+1}}) < \mathrm{f}(\mathrm{p}_{\mathrm{i,\,G}}) \\ \mathrm{p}^{\mathrm{D}}_{\mathrm{ij,\,G}}, & \text{se } \mathrm{f}(\mathrm{x}^{\mathrm{D}}_{\mathrm{j,\,G+1}}) \geq \mathrm{f}(\mathrm{p}_{\mathrm{i,\,G}}) \end{cases} \tag{3.5}$$

Assim, a nova posição individual da partícula ($x^D_{j,G+1}$) é comparada com a melhor posição individual da geração anterior ($P_{i,G}$) e é atualizada se esta gerar uma solução melhor que a comparação, caso contrário, sua melhor posição individual se mantem a mesma. Por consequência o conjunto de posições ideais do enxame (P_g) será formado pela melhor posição individual de cada indivíduo.

Atualização da velocidade e da posição das partículas

Uma vez que as posições foram avaliadas, as velocidades e a posições individuais, $\mathrm{v_j}$ e $\mathrm{x_j}$, serão atualizadas conforme demonstrado nas Equações 3.6 e 3.7. Desta forma, é criado então um novo enxame com soluções melhores que as anteriores.

$$\mathrm{v}^{\mathrm{D}}_{\mathrm{j,\,G+1}} = \omega\, \mathrm{v}^{\mathrm{D}}_{\mathrm{j,\,G}} + \mathrm{c_1 rand}\left(\mathrm{p}^{\mathrm{D}}_{\mathrm{ij,\,G}} - \mathrm{x}^{\mathrm{D}}_{\mathrm{j,\,G}}\right) + \mathrm{c_2 rand}\left(\mathrm{p}^{\mathrm{D}}_{\mathrm{gj,\,G}} - \mathrm{x}^{\mathrm{D}}_{\mathrm{j,\,G}}\right) \tag{3.6}$$

$$\mathrm{x}^{\mathrm{D}}_{\mathrm{j,G+1}} = x^D_{j,G} + v^D_{j,G+1} \tag{3.7}$$

A Equação 3.6 atualiza a velocidade levando em conta os parâmetros de inércia (ω), componente cognitivo (c_1) e componente social (c_2). O parâmetro ω pesa o quanto a inércia contribuirá para atualização da velocidade e consequente atualização da posição da partícula, o parâmetro c_1 pesa o quanto a melhor posição individual da partícula contribuirá e, por fim, o parâmetro c_2 pesa o quanto a melhor posição do enxame contribuirá para a atualização. Nesta equação, rand é um número aleatório com distribuição uniforme entre [0,1]. A atualização da posição da partícula, descrita pela Equação 3.7, é dada pela soma da solução da geração anterior com a nova velocidade.

Critério de Parada

A atualização da velocidade e posição das partículas deve ser repetida até que o critério de parada do algoritmo seja atingido. Neste trabalho o critério de parada adotado foi um número pré-definido de gerações de enxames ($\mathrm{N_{fes}}$).

Parâmetros do PSO

Como também não há consenso na literatura para a escolha dos parâmetros c_1, c_2 e ω, os conjuntos de parâmetros específicos testados para o algoritmo *Particle Swarm Optimization* estão descritos na Tabela 3.4. Os parâmetros testados nos conjuntos 4 e 5 são parâmetros que aparecem mais frequentemente na literatura, seguindo a premissa de que deve haver equilíbrio entre a inércia e os componentes social e cognitivo e por isso foram incluídos nos testes realizados. Para este algoritmo, o código padrão exigia uma definição de velocidade mínima e máxima para as partículas, o que limitava o espaço de busca do algoritmo e dificultava a convergência, por isso o código padrão foi alterado para e passou a não considerar esta condição.

Tabela 3.4: Parâmetros específicos utilizados no algoritmo *Particle Swarm Optimization*

Algoritmo	**Parâmetros**		
PSO	**Componente cognitivo (c_1)**	**Componente social (c_2)**	**Inécia (ω)**
Conjunto 1	2,05	2,05	0,4
Conjunto 2	1,5	1,5	0,9
Conjunto 3	2,5	1,5	0,2
Conjunto 4	2	2	0,7
Conjunto 5	1,5	1,5	0,9

3.2.4 Bat Algorithm (BA)

O *Bat Algorithm*, algoritmo de otimização com mecanismo de funcionamento inspirado na eco localização de morcegos utilizado neste trabalho, tem seu passo a passo mostrado na Figura 3.5.

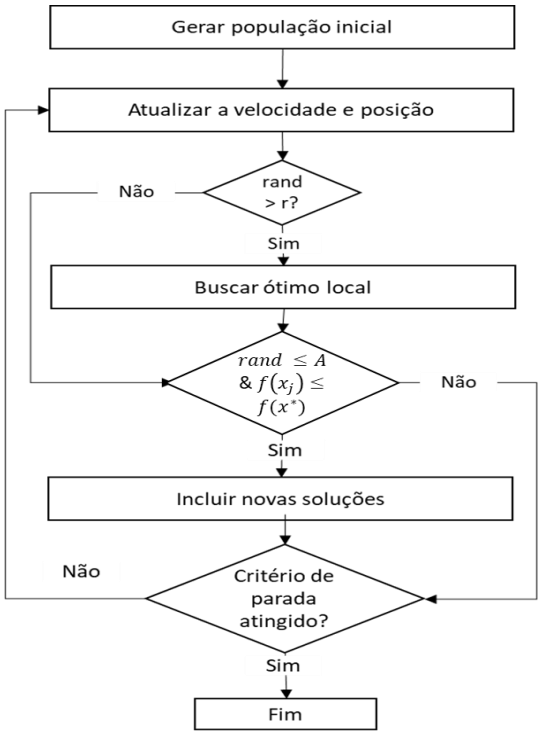


Figura 3.5: Fluxograma do *Bat Algorithm*

Para este trabalho, o algoritmo utilizado considerou que os morcegos voam aleatoriamente com velocidade v_j , posição x_j e uma frequência f_j ajustando sua frequência

a partir de um fator aleatório. Uma vez atualizada a frequência, são atualizadas, então, a velocidade e posição, para assim se aproximar do ótimo global. A taxa de emissão de pulsos $r \in [0,1]$ e amplitude A são mantidas fixas.

Geração da população

A primeira população de morcegos é gerada aleatoriamente, assumindo uma distribuição de probabilidade uniforme e tem seus valores de solução representados pela posição x_j e pela velocidade v_j. O tamanho da população é dado pelo parâmetro NP e sua frequência por f_G.

Atualização da velocidade

A atualização da velocidade $v_{j,G}$ e posição $x_{j,G}$ se inicia com a atualização da frequência f_G, como pode ser observado nas equações 3.8 e 3.9.

$$f_{G+1} = f_{mín} + (f_{máx} - f_{mín})\beta \tag{3.8}$$

$$v_{j,G+1} = v_{j,G} + (x_{j,G} - x^*)f_{G+1} \tag{3.9}$$

Assim, tem-se que f_{G+1}, onde G é a geração atual do enxame, será atualizada de acordo com a soma entre $f_{mín}$ e a diferença entre o seu valor máximo e mínimo, que terá influência controlada pelo parâmetro aleatório $\beta \in [0,1]$. A partir disso, então, a velocidade $v_{j,G+1}$, onde $j = 1, 2, 3, \ldots NP$, é atualizada considerando pela soma da sua velocidade anterior com a diferença entre a posição atual e a posição ótima, que é a melhor posição dentre todas as posições atuais, controladas pela frequência f_{G+1}.

Atualização da posição

O algoritmo BA poderá ter a posição dos morcegos atualizadas de duas diferentes maneiras. A primeira por meio da busca global, que se dá pela Equação 3.10, e é realizada sempre que a velocidade for atualizada. A nova posição gerada pela busca global ($x_{j,G+1}$) será o resultado da soma da solução anterior ($x_{j,G}$) e a velocidade atualizada ($v_{j,G+1}$).

$$x_{j,\,G+1} = x_{j,G} + v_{j,\,G+1} \tag{3.10}$$

A busca local, diferentemente da global que ocorre para todas as gerações, será realizada somente se um número aleatório $rand \in [0,1]$com distribuição uniforme for maior que a taxa de emissões de pulsos r. Logo, se $rand_1 > r$, o algoritmo realizará a busca local, dada pela Equação 3.11. A solução gerada pela busca local ($x_{j,G+1}$) é dada pela soma da solução anterior ($x_{j,G}$) com a amplitude (A), onde a influência da amplitude é controlada pelo fator ε que será um número aleatório entre -1 e 1. Caso contrário, se $rand_1 < r$, não haverá busca local, e será realizada diretamente a avaliação da solução.

$$x_{j,\,G+1} = x_{j,\,G} + \varepsilon A \tag{3.11}$$

Avaliação das soluções

A avaliação da solução neste algoritmo, assim como para os algoritmos já mencionados aqui, identifica qual a melhor solução gerada até o momento em termos de minimização da função objetivo e faz com que o algoritmo siga nesta direção. Para este caso, as soluções,

tanto globais quanto locais, se houver, serão incluídas na nova população se seguirem o seguinte critério: o número aleatório $\mathrm{rand}_2 \in [0,2]$ deve ser menor do que a amplitude A e a nova solução $f(x_{j,G+1})$ deve ser melhor do que a melhor solução anterior $f(x_G^*)$ ou seja, deve satisfazer $rand_2 \leq \mathrm{A}\ \&\ f(x_{j,G+1}) \leq f(x_G^*)$.

Critério de Parada

O critério de parada estabelecido para esse algoritmo é um número pré-definido de iterações ($\mathrm{N_{fes}}$). A avaliação, atualização da velocidade e atualização da posição das partículas será repetida até que este critério seja atingido.

Parâmetros

O autor do *Bat Algorithm*, YANG (2010), sugere que o parâmetro taxa de pulso r tenha valores entre 0 e 1. Para os demais parâmetros não são sugeridos valores específicos. Dessa forma foram testados 5 conjuntos de parâmetros específicos para este algoritmo que podem ser observados na Tabela 3.5.

Tabela 3.5: Parâmetros específicos utilizados para o *Bat Algorithm.*

Algoritmo	**Parâmetros Específicos**			
BA	**Amplitude (A)**	**Taxa de pulso (r)**	**Frequência Mínima ($\mathrm{f_{mín}}$)**	**Frequência máxima ($\mathrm{f_{máx}}$)**
Conjunto 1	0,5	0,1	1	10
Conjunto 2	0,9	0,9	1	10
Conjunto 3	0,9	0,4	1	10
Conjunto 4	0,4	0,1	1	100
Conjunto 5	0,9	0,9	1	100

3.2.5 Flower Polination Algorithm (FPA)

O algoritmo FPA, é um algoritmo de otimização com mecanismo de funcionamento baseado na polinização das flores, este mecanismo é apresentado na Figura 3.6.

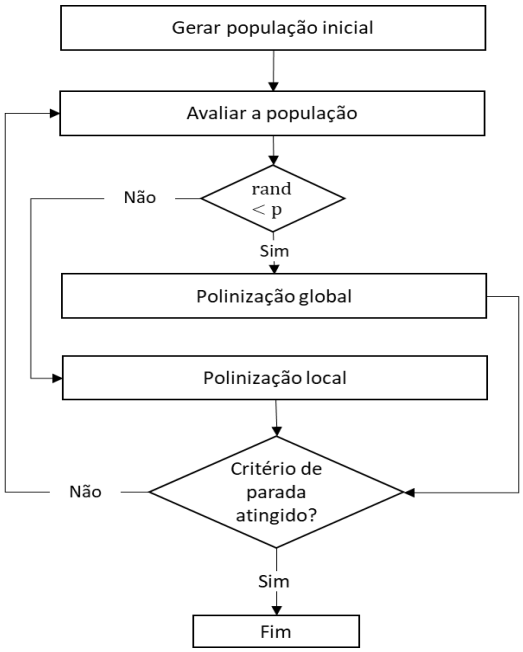


Figura 3.6: Fluxograma *Flower Pollination Algorithm*

Para fins de simplificação, o algoritmo utilizado, considera que cada flor é passível de gerar apenas uma nova flor, não sendo necessário então diferenciar flores de gametas de pólen, ou seja, cada flor gerada é uma solução em potencial que quando polinizada, tende a gerar uma solução (ou flor) melhor que a anterior.

Gerar a população inicial

A população inicial de flores, ou gametas de pólen, deve ser gerada aleatoriamente, assumindo uma distribuição de probabilidade uniforme, com tamanho definido pelo parâmetro NP. Após a inicialização, as soluções são avaliadas para que se encontre a melhor solução x^* e assim, posteriormente, serão geradas novas flores/gametas de pólen com soluções mais adequadas. Isso se dará por meio da polinização global ou local, que tem sua alternância controlada por uma probabilidade de troca entre os tipos de polinização definida pelo parâmetro $\mathrm{p} \in [0,1]$.

Avaliação da população

Todas as soluções geradas deverão ser avaliadas quanto à minimização da função objetivo. Uma nova solução só será aceita e fará parte da nova população uma vez que esta gere resultados melhores para função objetivo do que a solução anterior com a qual é comparada, do contrário, a solução se manterá a mesma para a próxima população.

Polinização global

A polinização global só ocorre quando um número gerado aleatoriamente ($\mathrm{rand} \in [0,1]$) é maior do que p, que é o parâmetro de probabilidade de troca entre a polinização global e local, do contrário, caso rand seja menor ou igual a p ocorrerá apenas a polinização local.

Na polinização global, o pólen é carregado por insetos polinizadores, que podem viajar longas distâncias, garantindo que ocorra a polinização das melhores flores. Esse tipo de polinização é representado pela Equação 3.12, onde a nova solução $\mathrm{x}_{\mathrm{j,G+1}}$ é resultado da soma entre a solução da geração anterior $\mathrm{x}_{\mathrm{j,\,G}}$ e diferença entre a solução anterior e a melhor solução x^* da população anterior, controlada pelo fator L, que é o parâmetro da força da polinização, dado pela Equação 3.14.

$$\mathrm{x}_{\mathrm{j,G+1}} = \mathrm{x}_{\mathrm{j,G}} + L\left(\mathrm{x}_{\mathrm{j,G}} - \mathrm{x}^*\right) \tag{3.12}$$

$$\mathrm{L} \sim \frac{\frac{\lambda\Gamma(\lambda)\mathrm{sem}(\lambda\pi)}{2}}{\pi}\ \frac{1}{\mathrm{s}^{1+\lambda}}, \qquad (\mathrm{s} \gg \mathrm{s}_0 > 0) \tag{3.13}$$

A Equação 3.13 é a distribuição de Lévy, onde $\Gamma(\lambda)$ é a função gama padrão e s é o tamanho do passo. Esta distribuição imita o comportamento dos polinizadores ao viajar para realizar a polinização das flores, visto que estes as viajam em largos passos, ou seja, percorrem grandes distâncias para polinizar as flores. Logo a distribuição de Lévy é válida para passos grandes e maiores que zero. As simulações realizadas neste trabalho consideraram $\lambda = 1{,}5$, assim como foi proposto pelo autor do algoritmo, YANG (2010).

Polinização local

Uma vez que $\mathrm{rand} \leq \mathrm{p}$ é realizada a polinização local. Esta imita o mecanismo de reprodução das flores quando elas são polinizadas pelo pólen da mesma flor ou de outras flores da mesma planta, esse processo acontece quando não há outros polinizadores disponíveis. Este mecanismo é representado pela Equação 3.14.

$$\mathrm{x_{j,\,G+1}} = \mathrm{x_{j,\,G}} + \epsilon\left(\mathrm{x_{j,\,G}} - \mathrm{x_{k,\,G}}\right) \tag{3.14}$$

Na Equação 3.14, $x_{j,G}$ e $x_{k,G}$ são pólens de diferentes flores da mesma espécie. Matematicamente, $x_{j,G}$ e $x_{k,G}$ são duas soluções retiradas aleatoriamente da mesma população. O parâmetro ϵ é o parâmetro de distribuição uniforme que transforma essa busca em aleatória. A nova solução $x_{j,G+1}$, será dada pela soma da solução anterior com a diferença entre a solução anterior e uma solução aleatória da mesma geração, controlada pelo fator ϵ.

Critério de parada

Para as simulações realizadas com este algoritmo, o critério de parada estabelecido foi o número de populações geradas N_{fes}, portanto, o algoritmo continuará realizando a polinização até que este critério seja atingido.

Parâmetros

O autor do FPA, YANG (2012), sugere que sejam testados diferentes valores para o parâmetro de probabilidade de troca (p) para verificar qual proporciona o melhor desempenho ao algoritmo. Os valores testados são descritos na Tabela 3.6.

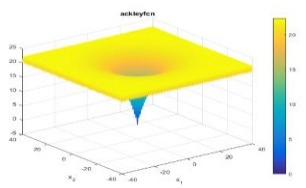
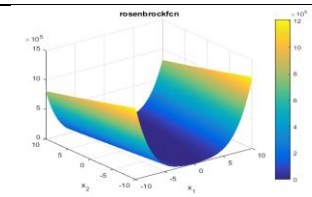
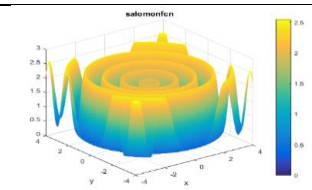
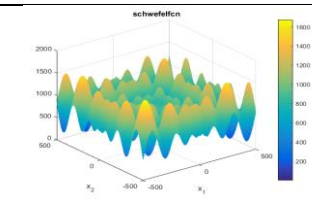
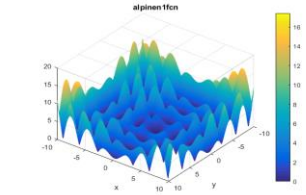
Tabela 3.6: Parâmetros específicos utilizados para o *Flower Pollination Algorithm*

Algoritmo	**Parâmetros**
FPA	**Probabilidade de troca (p)**
Conjunto 1	0,4
Conjunto 2	0,5
Conjunto 3	0,6
Conjunto 4	0,56
Conjunto 5	0,3

3.3 Funções de *benchmark*

Para que fosse possível avaliar e comparar os algoritmos, foram utilizadas cinco funções de *benckmark*, mostradas na Tabela 3.7. Estas funções já possuem seus máximos e mínimos conhecidos, o que torna possível avaliar a aderência dos resultados obtidos pelos algoritmos, bem como comparar o tempo e sua eficácia quando testadas para cada um dos algoritmos. Conforme mencionado anteriormente, na etapa 1 de análise foram testadas as funções Ackley, Rosenbrock e Schwefel, enquanto na etapa 2 foram testadas todas as 5 funções descritas na Tabela 3.7. Todas as funções descritas Tabela 3.7 possuem mínimo global em $\mathrm{f(x)} = 0$ e dimensão 10.

Tabela 3.7: Funções de *benchmark* testadas

Nome	Fórmula	Representação 3D
Ackley	$f(x)= 20.\exp^{*}\left(-0{,}2\sqrt{\frac{1}{D}\sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D}\sum_{j=1}^{D}\cos(2\pi x_j)\right) + 20 + e$	
Rosenbrock	$f(x) = \sum_{j=1}^{D-1}\left((x_j - 1)^2 + 100\,(x_{j+1} - x_j^2)^2\right)$	
Salomon	$f(x)= 1 - \cos\left(2\pi\sqrt{\sum_{j=1}^{D} x_j^2}\right) + 0{,}1\sqrt{\sum_{i=j}^{D} x_j^2}$	
Schwefel	$f(x)= -\sum_{j=1}^{D} x_j \sin\left(\sqrt{\lvert x_j\rvert}\right)$	
Alpine 1	$f(x) = \sum_{j=1}^{D} \lvert x_j \sin(x_j) + 0{,}1\, x_j\rvert$	

3.4 Critérios de avaliação

A análise comparativa realizada neste trabalho se deu através da utilização de métricas que visam comparar o desempenho dos algoritmos em relação a sua eficácia, eficiência, robustez e diversidade além de comparar a distância média do ótimo das soluções geradas e a média do tempo de convergência dos algoritmos. A partir da utilização destas métricas foi possível normalizar os dados obtidos nas simulações dos algoritmos e, assim, compará-los.

As métricas utilizadas para a definição do melhor conjunto de parâmetros específicos para os algoritmos na etapa 1 de análise foram a distância média do ótimo (μ) e a robustez (η). Na etapa 2, para que fosse possível realizar uma análise mais completa dos algoritmos e compará-los entre si, foram calculadas a distância média do ótimo (μ) e o tempo médio ($\overline{T}$) e a partir disso, calculadas a eficácia (θ), a eficiência (r), a robustez (η) e a diversidade (DP). As métricas foram baseadas na apostila da disciplina de otimização da UFRGS elaborada por SECCHI, A. R. (2001) e são descritas como segue:

- Distância média do ótimo (μ): A distância média do ótimo, dada pela Equação 3.15, traz o resultado médio gerado pelo algoritmo para cada

uma das funções de benchmark. Onde x_j é o resultado gerado em cada uma das simulações (j) e N é o número total de simulações realizadas. Logo para um algoritmo que teve seu desempenho testado por 5 funções, por exemplo, r será igual a 500, considerando que foram realizadas 100 simulações para cada função.

$$\mu = \frac{\sum_{j=1}^{N} x_j}{N} \tag{3.15}$$

- Tempo médio ($\overline{T}$): A média de tempo, dada pela Equação 3.16 é a medida que permite comparar as diferenças entre os tempos de convergência dos algoritmos. Para esta equação T_j é o tempo de convergência para cada uma das N simulações de um mesmo algoritmo e $\overline{T}$ é a média do tempo de convergência do algoritmo.

$$\overline{T} = \frac{\sum_{j=1}^{N} T_j}{N} \tag{3.16}$$

- Eficácia (θ): A eficácia, dada pela Equação 3.17, traduz o quanto um algoritmo é eficaz em relação ao tempo de convergência no que tange a proximidade dos tempos de convergência em relação ao melhor tempo entre as simulações. Nesta equação N o número de vezes que a simulação foi realizada, T^* o melhor tempo entre todas as simulações e T_j o tempo de cada simulação. Essa métrica foi aplicada para cada um dos algoritmos e comparada em relação à mesma função de benchmark.

$$\theta = \frac{100}{N} \sum_{j=1}^{N} \frac{T_j^*}{T_j} \tag{3.17}$$

- Eficiência (γ): A métrica dada pela Equação 3.18 traz uma avaliação em relação à proximidade das soluções geradas em relação à melhor solução gerada pelo algoritmo. Onde x^* é o resultado mais próximo do mínimo global dentre todas as simulações para um mesmo algoritmo e x_j o resultado de cada uma das N simulações realizadas com o algoritmo. Assumindo que x^* nunca será exatamente zero.

$$\gamma = \frac{100}{N} \sum_{j=1}^{N} \frac{x^*}{x_j} \tag{3.18}$$

- Robustez (η): A robustez é a medida da capacidade de otimização de problemas complexos do algoritmo. Ela é dada pela Equação 3.19, onde N^* é o número de problemas resolvidos pelo algoritmo e N o número total de simulações realizadas. O número de problemas

resolvidos (N^*) é dado pelo número de vezes que o algoritmo demonstrou resultados menores que 0,9 para a simulação, visto que todas as funções testadas neste trabalho possuem 0 como mínimo global.

$$\eta = 100\ \frac{N^*}{N} \tag{3.19}$$

- Diversidade (D): É a medida da diversidade das soluções alcançadas por um algoritmo. Ela é dada pela Equação 3.20, onde NP é o tamanho da população, x_j as soluções e $\bar{x}$ a média das soluções.

$$DP = \frac{1}{N}\sqrt{\sum_{j=1}^{N}(x_j - \bar{x})^2} \tag{3.20}$$

4 Resultados

Como a análise do desempenho dos algoritmos consistiu em duas etapas, este capítulo apresenta na Seção 4.1 os resultados da primeira etapa da análise, que tem por objetivo realizar o ajuste dos parâmetros específicos para cada algoritmo, definindo qual o melhor conjunto de parâmetros a ser utilizado. A segunda etapa da análise é apresentada na Seção 4.2, que visa comparar sistematicamente o desempenho dos algoritmos, levando em consideração os critérios de avaliação apresentados na Seção 3.4.

4.1 Ajuste de parâmetros específicos

Os algoritmos tiveram seu desempenho testado para os conjuntos de parâmetros específicos apresentados no Capítulo 3 em relação às funções de benchmark *Ackley, Rosenbrock e Schwefel,* ilustradas na Tabela 3.7. Foram realizadas simulações para cada um dos algoritmos mantendo-se o tamanho da população e número de gerações constantes em 30 e 10000, respectivamente. Os resultados para cada algoritmo, bem como a definição do conjunto de parâmetros são apresentados nas subseções seguintes.

4.1.1 Algoritmo Genético

A Tabela 4.1 apresenta os resultados dos critérios analisados nesta etapa para o algoritmo genético. Dessa forma, percebe-se que os conjuntos de parâmetros específicos 4 e 5 apresentaram uma menor robustez quando comparados com os conjuntos 1, 2 e 3. O conjunto 2 foi o que obteve a maior robustez dentre todos, isto significa que dentre 50 simulações realizadas para cada função, quando utilizado o conjunto 2, em média, foi possível encontrar o ótimo global 19 vezes. No entanto, a distância média do ótimo (μ) foi menor para todos os outros conjuntos de parâmetros, embora tenha apresentado pouca variação.

Como o principal critério de decisão para a escolha do conjunto de parâmetros específicos a ser utilizado nas análises da etapa 2 é a robustez, o conjunto 2 foi o escolhido.

Tabela 4.1: Desempenho médio para cada conjunto de parâmetros específicos do AG

Algoritmo	Parâmetros específicos	Distância média do ótimo (μ)	Robustez (η)
AG	Conjunto 1	185,42	36%
AG	Conjunto 2	189,32	38%
AG	Conjunto 3	174,83	36%
AG	Conjunto 4	178,56	17%
AG	Conjunto 5	186,84	17%

4.1.2 Algoritmo Evolução Diferencial

A partir dos resultados da análise dos critérios para o algoritmo Evolução Diferencial, apresentados na Tabela 4.2, é possível perceber que o conjunto de parâmetros específicos 1, 3 e 5 têm robustez (η) superior aos demais. Dessa forma o conjunto escolhido foi o 1, pois apresenta a maior robustez dentre todos os 5 conjuntos avaliados.

Tabela 4.2: Desempenho médio para cada conjunto de parâmetros específicos do ED

Algoritmo	Parâmetros específicos	Distância média do ótimo (μ)	Robustez (η)
ED	Conjunto 1	15,79	63%
ED	Conjunto 2	40,70	46%
ED	Conjunto 3	212,67	61%
ED	Conjunto 4	58,33	39%
ED	Conjunto 5	11,17	62%

4.1.3 Algoritmo Particle Swarm Optimization

O desempenho do algoritmo PSO em relação a cada um dos conjuntos de parâmetros específicos é apresentado na Tabela 4.3. O conjunto 1 de parâmetros é o que proporciona o melhor desempenho ao algoritmo, com robustez média um ponto percentual maior que os conjuntos 3, possuindo também a menor distância média do ótimo. Os demais apresentaram desempenho muito inferior aos conjuntos 1 e 3. Dessa forma, o conjunto de parâmetros escolhido para ser utilizado nas simulações da etapa 2 foi o conjunto 1.

Tabela 4.3: Desempenho médio para cada conjunto de parâmetros específicos do PSO

Algoritmo	Parâmetros específicos	Distância média do ótimo (μ)	Robustez (η)
PSO	Conjunto 1	39,48	35%
PSO	Conjunto 2	2728,96	15%
PSO	Conjunto 3	79,47	34%
PSO	Conjunto 4	7783,58	0,33%
PSO	Conjunto 5	99706	0

4.1.4 Algoritmo Bat Algorithm

Os resultados das avaliações dos conjuntos de parâmetros utilizados para o *Bat Algorithm* são apresentados na Tabela 4.4. O BA não demostrou um bom desempenho com nenhum dos conjuntos de parâmetros específicos testados para o tamanho da população e número de iterações estabelecidas. A escolha do conjunto de parâmetros para este algoritmo se deu então levando em consideração a menor média da distância do ótimo global ($\bar{\mu}$), logo o conjunto de parâmetros escolhido foi o 1.

Tabela 4.4: Desempenho médio para cada conjunto de parâmetros específicos do BA

Algoritmo	Parâmetros específicos	Distância média do ótimo (μ)	Robustez (η)
BA	Conjunto 1	678,18	0
BA	Conjunto 2	1250,14	0
BA	Conjunto 3	892,80	0
BA	Conjunto 4	680,81	0
BA	Conjunto 5	1319,70	0

4.1.5 Algoritmo Flower Pollination

Os resultados das avaliações dos conjuntos de parâmetros específicos do FPA são apresentados na Tabela 4.5. Para este algoritmo o conjunto de parâmetros escolhido foi o conjunto 1, devido à sua maior proximidade média do ótimo global, visto que a robustez fica empatada com o conjunto 2, onde para cada 50 simulações, o ótimo global é encontrado 16 vezes.

Tabela 4.5: Desempenho médio para cada conjunto de parâmetros específicos do FPA

Algoritmo	**Parâmetros específicos**	**Distância média do ótimo (μ)**	**Robustez (η)**
FPA	Conjunto 1	406,61	33%
FPA	Conjunto 2	561,11	33%
FPA	Conjunto 3	576,76	26%
FPA	Conjunto 4	554,09	15%
FPA	Conjunto 5	393,67	17%

4.2 Comparação sistemática de desempenho dos algoritmos

A segunda etapa das análises se deteve em realizar uma comparação sistemática do desempenho dos algoritmos. Para isso, esses foram testados utilizando as funções de *benchmark Ackley, Rosenbrock, Schwefel, Salomon* e *Alpine 1,* ilustradas na Tabela 3.7. Os testes foram realizados para 3 tamanhos de população, 10, 50 e 100, utilizando os parâmetros específicos definidos na Seção 284.1. Nesta seção são apresentados os resultados de desempenho dos algoritmos para cada um dos critérios de avaliação.

A robustez é o critério de avaliação que mede o quanto o algoritmo é capaz de resolver os problemas testados. A Figura 4.1 apresenta a robustez média para cada algoritmo em cada uma das populações testadas, ou seja, dentre todas as simulações realizadas para cada uma das funções testadas, em média para uma população de tamanho 10, o algoritmo ED foi o algoritmo com melhor desempenho, seguido pelo AG, FPA, PSO e BA. Para populações de tamanho 50, o ED também apresenta o melhor desempenho, seguido do PSO, AG, FPA e BA. Por fim, para uma população de tamanho 100, o AG sai na frente com desempenho superior, seguido pelo PSO, pelo ED, FPA e BA.

Dessa forma, é possível perceber que, em geral populações menores que 100 proporcionam uma maior robustez para os algoritmos. Em relação a este critério é possível perceber que, como esperado, os algoritmos mais antigos, o ED, PSO e AG, obtiveram melhor desempenho. O Algoritmo Genético dentre os três mais antigos, foi o algoritmo que obteve maior consistência em termos de robustez para os diferentes tamanhos de população, enquanto o ED e PSO apresentaram bons resultados para populações de tamanho 50, mas resultados abaixo do esperado para populações de tamanho 10 e 100.

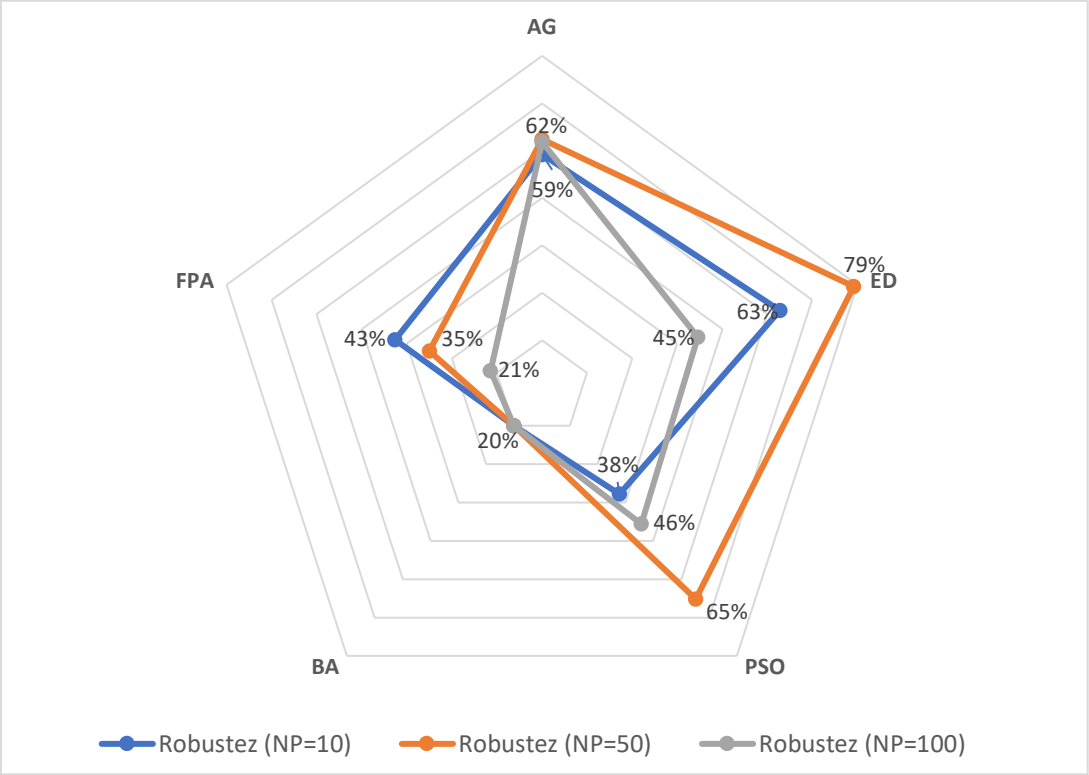


Figura 4.1: Robustez (η) dos cinco algoritmos para as três populações testadas.

O critério eficiência mede o quanto o algoritmo é consistente em encontrar o melhor resultado possível em todas as simulações comparando-se com o seu próprio melhor resultado gerado. Ou seja, conforme demonstrado na Figura 4.2, para uma população de tamanho 10, o BA foi o algoritmo com maior eficiência, apresentando o melhor resultado em 37% das simulações, seguido pelo FPA, AG, ED e PSO. Para populações de tamanho 50 o algoritmo que mais se destacou nesse critério foi o ED, que apresentou uma eficiência muito superior aos outros algoritmos, seguido pelo BA, PSO, AG e FPA. Por fim, para populações de tamanho 100, o ED continua sendo o algoritmo com maior eficiência seguido pelo BA, PSO, FPA e AG.

A partir das análises realizadas percebe-se que, para este critério, nenhum dos algoritmos obteve resultados satisfatórios, no entanto, exceto pelo algoritmo Evolução Diferencial que apresentou resultados superiores a todos os algoritmos, os algoritmos mais recentes, FPA e BA, tiveram desempenho similar aos clássicos AG e PSO. Isso denota que embora o FPA e BA tenham maior dificuldade para convergir, vide baixa robustez, estes conseguem manter a consistência dos seus bons resultados tanto quanto o AG e PSO.

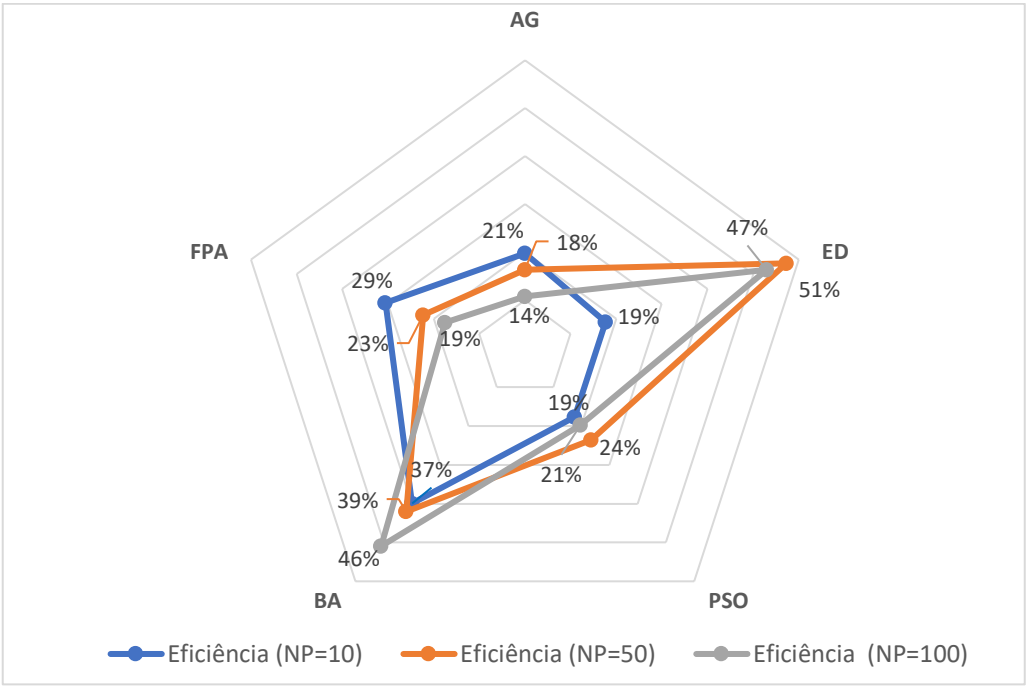


Figura 4.2: Eficiência (γ) dos cinco algoritmos para as três populações testadas.

A eficácia, com resultados apresentados na Figura 4.3, é o critério de avaliação que mede o quanto o algoritmo é consistente em seu tempo de convergência. Isto é, ela mede quantas vezes o algoritmo conseguiu obter o melhor tempo dentre todas as simulações realizadas quando comparado ao seu melhor tempo obtido. Dessa forma, embora nenhum algoritmo destaque-se neste critério, percebe-se que todos possuem em geral uma boa eficácia, independentemente do tamanho da população, embora denote-se que cada algoritmo apresenta melhor eficácia para um tamanho de população específico.

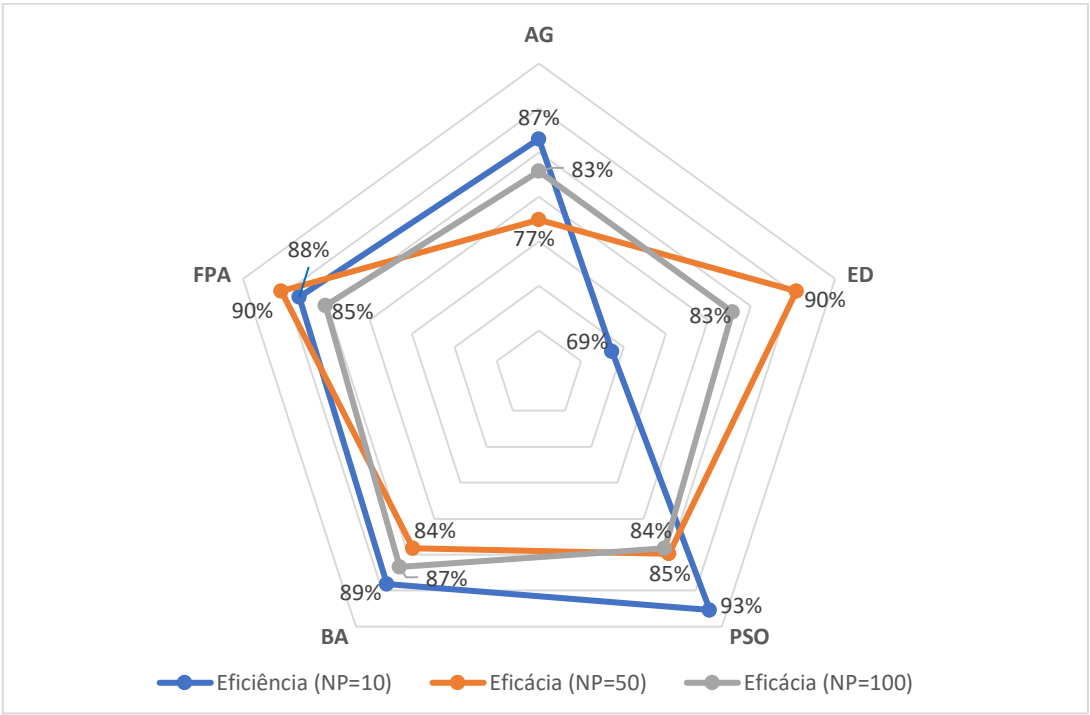


Figura 4.3: Eficácia média ($\bar{\theta}$) dos cinco algoritmos para as três populações testadas.

Os resultados apresentados nas Figuras 4.1, 4.2 e 4.3 e na Tabela 4.6 denotam que, exceto pelo algoritmo FPA, que apresenta melhor desempenho para populações de tamanho 10, os demais têm melhor desempenho para populações de tamanho 50. Além disso, percebe-se que populações maiores que 50 proporcionam, em geral, um pior desempenho.

Tabela 4.6: Médias da distância média do ótimo, diversidade e tempo médio de convergência dos cinco algoritmos para as três populações testadas.

Tamanho da população (NP)	**Algoritmo**	**Distância média do ótimo (μ)**	**Diversidade (DP)**	**Tempo médio (s) ($\overline{T}$)**
10	AG	121,39	1217,1	0,38
50	AG	110,33	4,34	0,73
100	AG	103,57	4,74	1,06
10	ED	18003,29	446887,2	0,33
50	ED	4,05	34,2	0,28
100	ED	86,59	311,9	0,32
10	PSO	3290,96	68181,1	0,55
50	PSO	160,29	618,9	0,61
100	PSO	765,53	31166,1	0,62
10	BA	452,66	5026,7	0,35
50	BA	388,86	1553,8	0,39
100	BA	400,34	2170,4	0,37
10	FPA	208,35	600,1	0,71
50	FPA	279,53	525,9	0,70
100	FPA	328,62	924,9	0,76

A Figura 4.4 traz o resultado da avaliação dos algoritmos em relação aos critérios robustez, eficiência e eficácia para a população que proporciona o melhor desempenho ao algoritmo. Dessa forma é possível compará-los a partir do seu melhor desempenho individual e assim estabelecer o melhor dentre os avaliados. Portanto, para as simulações realizadas neste trabalho, o algoritmo que apresenta o melhor desempenho, levando em consideração os critérios de robustez, eficiência e eficácia, em ordem de relevância, é o algoritmo Evolução Diferencial, seguido pelo *Particle Swarm Optimization* e pelo Algoritmo Genético. Isto significa que os algoritmos mais antigos tiveram um desempenho geral melhor do que o os mais novos, *Flower Pollination Algorithm* e *Bat Algorithm.*

Dentre os algoritmos mais recentes, o que apresentou melhor desempenho foi o *Flower Pollination Algorithm,* obtendo uma robustez 23 pontos percentuais maior do que o *Bat Algorithm* e eficácia 4 pontos percentuais maior. O único critério para o qual o FPA não foi superior ao BA foi o de eficiência, para este critério a diferença foi de 10 pontos percentuais a mais para o BA.

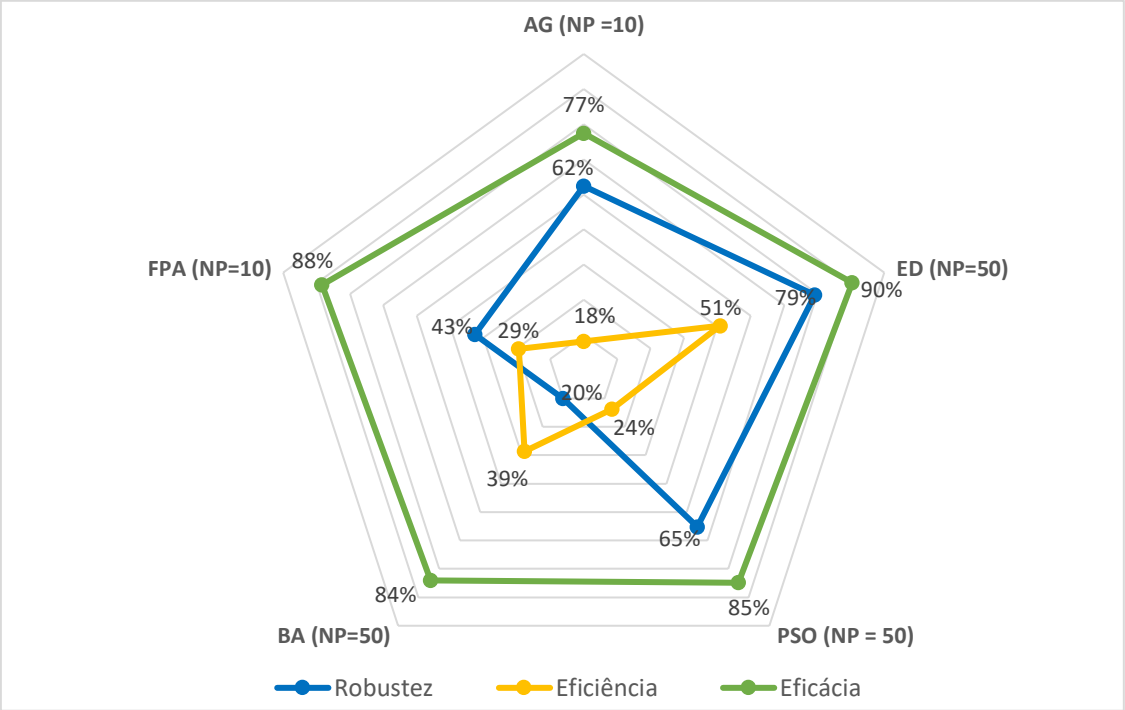


Figura 4.4: Comparação dos critérios robustez, eficiência e eficácia para cada algoritmo na sua população ideal.

Os resultados apresentados neste trabalho vão de encontro aos apresentados pelo criador dos algoritmos FPA e BA, YANG (2012), que expõe estes possuem desempenho superior aos clássicos AG e PSO, por exemplo. YANG (2012) comparou o FPA em relação aos algoritmos AG e PSO em termos de número de iterações, para funções com dimensão superior a 10 e YANG (2010) comparou o BA da mesma forma. Para o FPA, obteve-se 100% de convergência para todos os testes realizados, com um número de iterações menor que 10000. Já o BA obteve desempenho inferior ao FPA, assim como apresentado neste trabalho, mas ainda superior aos clássicos AG e PSO.

No entanto, outros trabalhos como WANG; ZHOU, (2014) e FISTER; FISTER; YANG, (2013) vão ao encontro com o que foi demonstrado no presente estudo. WANG; ZHOU (2014) apresenta resultados de desempenho para o FPA inferiores a todos os outros algoritmos clássicos, como o PSO e ED. FISTER; FISTER; YANG, (2013) apresenta convergência nula para o BA em todos os testes realizados.

Ademais a esse fato, encontra-se na literatura muitas pesquisas visando desenvolver adaptações aos algoritmos FPA e BA, como por exemplo YILMAZ; KÜÇÜKSILLE, (2015) que propõe uma nova estratégia para controlar o parâmetro de velocidade do BA como forma de melhorar seu desempenho, ou NABIL, (2016) que modificou o FPA, combinando-o com um segundo algoritmo também visando melhorar seu desempenho. A partir disso, fica claro que não há consenso na literatura quanto ao desempenho do FPA e BA e percebe-se que ainda são necessários mais estudos comprovando seu desempenho, bem como propondo melhorias que permitam que eles obtenham um desempenho satisfatório de forma consistente.

5 Conclusões e Trabalhos Futuros

Este trabalho teve como objetivo mensurar o desempenho de cinco diferentes algoritmos, o Algoritmo Genético (AG), Evolução Diferencial (ED), *Particle Swarm Optimization (PSO), Bat Algorithm (BA)* e *Particle Swarm Optimization,* por meio de uma comparação sistemática dos mesmos. Para isso os algoritmos foram testados em relação à sua capacidade de otimização para funções de *benchmark* selecionadas. As análises consistiram em duas etapas, a primeira se deteve em ajustar os parâmetros dos algoritmos e a segunda realizou a comparação final a partir dos critérios de avaliação definidos.

O estudo realizado concluiu que há um grande número de algoritmos estocásticos, disponíveis, dessa forma, quando se faz necessário realizar uma otimização, há uma ampla possibilidade de escolha. No entanto, os algoritmos não são caracterizados de modo que seja possível compará-los facilmente, isso demonstra a importância do método de comparação sistemática utilizado neste trabalho. O método permitiu que os algoritmos tivessem seu desempenho avaliado de maneira normalizada, ou seja, que o desempenho dos algoritmos fosse traduzido em critérios de avaliação que pudessem ser comparados sem a necessidade de avaliar resultados específicos, como valor do ótimo global e tempo de convergência, por exemplo, para definir qual possui melhor desempenho.

A comparação dos algoritmos, realizada a partir da aplicação dos critérios de avaliação apresentados, demonstrou que os algoritmos clássicos, como o AG, PSO e ED ainda se destacam quando comparados aos mais recentes FPA e BA. O algoritmo com o melhor desempenho dentre todos foi o Evolução Diferencial, seguido do *Particle Swarm Optimization* e Algoritmo Genético. É importante ressaltar que os resultados obtidos evidenciaram que o desempenho geral dos algoritmos é fortemente influenciado pela escolha dos seus parâmetros específicos, isso denota a importância de se estabelecer uma etapa para definir o melhor conjunto de parâmetros para cada um dos algoritmos.

Para os testes com os conjuntos de parâmetros específicos, buscou-se utilizar como ponto de partida valores propostos pela literatura, contudo, embora alguns estudos apresentem heurísticas para essa definição desses parâmetros, os dados encontrados nas pesquisas divergem e os melhores desempenhos se deram a partir dos valores estabelecidos pelos testes realizados neste trabalho, os quais foram de encontro às propostas apresentadas pela literatura. Assim, conclui-se que apesar de os parâmetros específicos dos algoritmos serem importantes fatores para potencializar o desempenho, não há estudos suficientes que proponham heurísticas satisfatórias para sua definição, por isso levanta-se a hipótese de os valores ideais variarem de acordo com o problema a ser resolvido e estrutura de código dos algoritmos.

Deste modo, como principais sugestões para trabalhos futuros destacam-se: a aplicação da comparação sistemática realizada neste trabalho em outros algoritmos estocásticos recentemente desenvolvidos; o desenvolvimento de um modelo que encontre o melhor conjunto de parâmetros para o algoritmo a ser utilizado, levando em consideração suas características específicas; e, além disso, sugere-se sejam realizados mais estudos no sentido de melhorar os algoritmos mais recentes, FPA e BA, para que estes atinjam o potencial esperado para a sua utilização e transcendam os algoritmos clássicos, permitindo assim a melhoria continua no tema otimização.

Referências

ABDELAZIZ, A. Y.; ALI, E. S.; ABD ELAZIM, S. M. Combined economic and emission dispatch solution using Flower Pollination Algorithm. International Journal of Electrical Power and Energy Systems, *[S. l.]*, v. 80, p. 264–274, 2016. Disponível em: https://doi.org/10.1016/j.ijepes.2015.11.093

ADARSH, B. R. *et al.* Economic dispatch using chaotic bat algorithm. Energy, *[S. l.]*, v. 96, p. 666–675, 2016. Disponível em: https://doi.org/10.1016/j.energy.2015.12.096

APARECIDO SALATTA, Vinícius; REGINA DOS SANTOS, Solange. Otimização Estocástica por meio de dois problemas de programação linear com coeficientes aleatórios. C.Q.D. – Revista Eletrônica Paulista de Matemática, *[S. l.]*, v. 10ic, p. 62–76, 2017. Disponível em: https://doi.org/10.21167/cqdvol10ermacic201723169664vassrs6276

BAHMANI-FIROUZI, Bahman; AZIZIPANAH-ABARGHOOEE, Rasoul. Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm. International Journal of Electrical Power and Energy Systems, *[S. l.]*, v. 56, p. 42–54, 2014. Disponível em: https://doi.org/10.1016/j.ijepes.2013.10.019

BEKDAŞ, Gebrail; NIGDELI, Sinan Melih; YANG, Xin She. Sizing optimization of truss structures using flower pollination algorithm. Applied Soft Computing Journal, *[S. l.]*, v. 37, p. 322–331, 2015. Disponível em: https://doi.org/10.1016/j.asoc.2015.08.037

BIRGE, John R.; LOUVEAUX, Franc. Introduction to Stochastic Programming Second Edition. *[S. l.: s. n.]*. *E-book*.

CÁRDENAS, Ramiro; RUIZ, Ramiro. Sobre Algoritmos de Otimização Estocásticos: Aplicações em Redesenho de Redes de Monitoramento e Mapeamento de QTL. *[S. l.]*, 2007.

CAVAZZUTI, M. Optimization Methods: From Theory to Design: Design of Experiments. *[S. l.: s. n.]*. *E-book*. Disponível em: https://doi.org/10.1007/978-3-642-31187-1

CUI, Zhihua *et al.* Hybrid many-objective particle swarm optimization algorithm for green coal production problem. *[S. l.: s. n.]* Disponível em: https://doi.org/10.1016/j.ins.2020.01.018

DE OLIVEIRA VASCONCELOS, Caio Francisco. Algoritmo para reconfiguração de sistemas de distribuição de energia elétrica baseado em evolução diferencial. 2016. *[s. l.]*, 2016.

DE SOUZA SERAPIÃO, Adriane Beatriz. FUNDAMENTOS DE OTIMIZAÇÃO POR INTELIGÊNCIA DE ENXAMES:UMA VISÃO GERAL. *[S. l.]*, 2009 a.

DE SOUZA SERAPIÃO, Adriane Beatriz. Fundamentos de Otimização por Inteligência de enxames: Uma Visão Geral. Controle y Automacao, *[S. l.]*, v. 20, n. 3, p. 271–304, 2009 b. Disponível em: https://doi.org/10.1590/s0103-17592009000300002

DURAND-LOSE, Jérôme; JONOSKA, Natasa. Unconventional Computation and Natural Computation. *[S. l.: s. n.]*. v. 3448*E-book*.

EBADIFARD, Fatemeh;; BABAMIR, Seyed. A PSO-based task scheduling algorithm improved

using a load-balancing technique for the cloud computing environment. *[S. l.]*, v. Volume 6, n. Number 3 / 2011, p. 167–171, 2017.

ESSIET, Ima O.; SUN, Yanxia; WANG, Zenghui. Optimized energy consumption model for smart home using improved differential evolution algorithm. Energy, *[S. l.]*, v. 172, p. 354–365, 2019. Disponível em: https://doi.org/10.1016/j.energy.2019.01.137

FALCONE, Marco Aurelio. Estudo Comparativo entre Algoritmos Genéticos e Evolução Diferencial para Otimização de um Modelo de Cadeia de Suprimento Simplificada. *[S. l.]*, 2004.

FISTER, Iztok; FISTER, Dusan; YANG, Xin She. A Hybrid Bat Algorithm. *[S. l.: s. n.]*.

GLAND, Eduardo D.; KLEIN, Michael T.; THOMAS, Edgar F. McGraw-Hill Chemical Engineering Series. *[S. l.: s. n.]*. *E-book*. Disponível em: https://gymkhana.iitb.ac.in/~scp/scp/ocw/chemical/cl 603 optimization/Optimization of Chemical Processes - Edgar Himmelblau and Lasdon 2nd ed.pdf

GONZÁLEZ, Juan R. *et al.* Nature Inspired Cooperative Strategies for Optimization. *In*: Studies in Computational Intelligence. *[S. l.: s. n.]*. v. 284p. 1689–1699. *E-book*. Disponível em: https://doi.org/10.1017/CBO9781107415324.004

HEGERTY, Brian; HUNG, Cc; KASPRAK, Kristen. A Comparative Study on Differential Evolution and Genetic Algorithms for Some Combinatorial Problems. Mexican International Conference on Artificial Intelligence, *[S. l.]*, 2009. Disponível em: http://www.micai.org/2009/proceedings/complementary/cd/ws-imso/88/paper88.micai09.pdf

LACERDA, Estéfane G. M. de;; CARVALHO, Andre Carlos P. L. F. de. Introdução aos Algoritmos Genéticos. *[S. l.: s. n.]*. *E-book*. Disponível em: http://www.leca.ufrn.br/~estefane/metaheuristicas/ag.pdf

LIBERTI, Leo; KUCHERENKO, Sergei. Comparison of deterministic and stochastic approaches to global optimization. International Transactions in Operational Research, *[S. l.]*, v. 12, n. 3, p. 263–285, 2005. Disponível em: https://doi.org/10.1111/j.1475-3995.2005.00503.x

MAGNANI, Bruna. Otimização do Scheduling de Nafta Petroquímica utilizando Algorítmos Genéticos. *[S. l.: s. n.]*.

MATTOS CAVALCANTE, César Lincoln. Aplicação de Algoritmos de Otimização Metaheurística para Gerência de Recursos de Rádio. *[S. l.]*, 2009.

MEDEIROS, José. Enxame de Partículas como Ferramenta de Otimização em Problemas Complexos de Engenharia Nuclear. *[S. l.]*, p. 102, 2005.

METAWA, Noura; HASSAN, M. Kabir; ELHOSENY, Mohamed. Genetic algorithm based model for optimizing bank lending decisions. Expert Systems with Applications, *[S. l.]*, v. 80, p. 75–82, 2017. Disponível em: https://doi.org/10.1016/j.eswa.2017.03.021

MONTAZ ALI, Mo; KHOMPATRAPORN, Charoenchai; ZABINSKY, Zelda B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization, *[S. l.]*, v. 31, n. 4, p. 635–672, 2005. Disponível

em: https://doi.org/10.1007/s10898-004-9972-2

NABIL, Emad. A Modified Flower Pollination Algorithm for Global Optimization. Expert Systems with Applications, *[S. l.]*, v. 57, p. 192–203, 2016. Disponível em: https://doi.org/10.1016/j.eswa.2016.03.047

NOCEDAL, Jorge; WRIGHT, Stetphen J. Numerical Optimization. *[S. l.: s. n.]*. v. 83*E-book*. Disponível em: https://doi.org/10.1103/PhysRev.83.134

PESSIN, Gustavo; OSÓRIO, Fernando. Otimização por Enxame de Partículas aplicado à formação e atuação de grupos robóticos. Scientia, *[S. l.]*, v. 20, n. 2, p. 94–106, 2009. Disponível em: https://doi.org/10.4013/sct.2009.20.2.03

REY NARIÑO, Giovanny Alfredo. Otimização de Risers em Catenária com Amortecedores Hidrodinâmicos. *[S. l.]*, 2014.

RIBEIRO RAMALHO, Reiga. Reconstrução de imagens de tomografia por impedância elétrica usando evolução diferencial. *[S. l.]*, 2016.

SOUZA, Guilherme V *et al.* Planejamento De Redes Wlan Utilizando Algoritmo Evolução Diferencial. Xlviii Sbpo, *[S. l.]*, p. 3244–3255, 2016.

STORN, Rainer;; PRICE, Kenneth. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, *[S. l.]*, p. 341–359, 1997. Disponível em: https://doi.org/10.1071/AP09004

WANG, Rui; ZHOU, Yongquan. Flower pollination algorithm with dimension by dimension improvement. Mathematical Problems in Engineering, *[S. l.]*, v. 2014, 2014. Disponível em: https://doi.org/10.1155/2014/481791

YANG, Xin-She;; GANDOMI, Amir H. Chaotic Bat Algorithm. Journal of Computational Science, *[S. l.]*, 2013. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1877750313001099

YANG, Xin She. Flower pollination algorithm for global optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *[S. l.]*, v. 7445 LNCS, p. 240–249, 2012. Disponível em: https://doi.org/10.1007/978-3-642-32894-7_27

YILMAZ, Selim; KÜÇÜKSILLE, Ecir U. A new modification approach on bat algorithm for solving optimization problems. Applied Soft Computing Journal, *[S. l.]*, v. 28, p. 259–275, 2015. Disponível em: https://doi.org/10.1016/j.asoc.2014.11.029