Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Civil

Elaboração de um Algoritmo de Otimização Aplicado à Engenharia Estrutural: *Circle-Inspired Optimization Algorithm* (CIOA)

Otávio Augusto Peter de Souza

Porto Alegre
2021

OTÁVIO AUGUSTO PETER DE SOUZA

ELABORAÇÃO DE UM ALGORITMO DE OTIMIZAÇÃO APLICADO À ENGENHARIA ESTRUTURAL: *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM* (CIOA)

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Civil da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Mestre em Engenharia.

Porto Alegre

2021

Souza, Otávio Augusto Peter de
Elaboração de um Algoritmo de Otimização Aplicado à Engenharia Estrutural: Circle-Inspired Optimization Algorithm (CIOA) / Otávio Augusto Peter de Souza. -- 2021.
152 f.
Orientadora: Letícia Fleck Fadel Miguel.

Dissertação (Mestrado) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Programa de Pós-Graduação em Engenharia Civil, Porto Alegre, BR-RS, 2021.

1. Otimização estrutural. 2. Algoritmos meta-heurísticos. 3. Treliças. 4. Circle-Inspired Optimization Algorithm. I. Miguel, Letícia Fleck Fadel, orient. II. Título.

OTÁVIO AUGUSTO PETER DE SOUZA

ELABORAÇÃO DE UM ALGORITMO DE OTIMIZAÇÃO APLICADO À ENGENHARIA ESTRUTURAL: *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM* (CIOA)

Esta dissertação de mestrado foi julgada adequada para a obtenção do título de MESTRE EM ENGENHARIA, Área de Concentração Estruturas, e aprovada em sua forma final pelo professor orientador e pelo Programa de Pós-Graduação em Engenharia Civil da Universidade Federal do Rio Grande do Sul.

Porto Alegre, 09 de março de 2021

Profª. Letícia Fleck Fadel Miguel
Drª. pela Universidade Federal do Rio Grande do Sul
Orientadora

Prof. Ph.D. Nilo Cesar Consoli
Coordenador do PPGEC/UFRGS

BANCA EXAMINADORA

Prof. Jesús Daniel Villalba Morales (PUJ/Bogotá)
Dr. pela Universidade de São Paulo (USP)

Prof. João Ricardo Masuero (PPGCI/UFRGS)
Dr. pelo PPGEC da Universidade Federal do Rio Grande do Sul (UFRGS)

Prof. Inácio Benvegnu Morsch (PPGEC/UFRGS)
Dr. pelo PPGEC da Universidade Federal do Rio Grande do Sul (UFRGS)

Dedico este trabalho à memória de minha avó, Nilda Nachtigall Peter, por ter me proporcionado as melhores lembranças e os maiores ensinamentos: sobre carinho, bondade e persistência.

AGRADECIMENTOS

Toda caminhada se torna mais leve, agradável e bonita quando temos boas companhias durante o percurso. Por isso, agradeço profundamente a todos que, de alguma forma, me acompanharam durante esta trajetória do mestrado.

Agradeço a meus pais, Romildo e Marlene, por todos os esforços para que eu tivesse acesso à educação de qualidade. A vocês devo o que sou e tudo aquilo que já alcancei nesta vida. Obrigado por lutarem, junto comigo, pela realização dos meus sonhos.

Agradeço à minha irmã Mônica e ao meu cunhado Márcio por toda ajuda e confiança que sempre depositaram em mim. O apoio de vocês e o fato de acreditarem no meu potencial, estiveram entre os maiores incentivos para que eu continuasse em frente, mesmo nas situações mais difíceis.

Agradeço ao meu avô Arno Peter, pelo carinho e alegria contagiantes que sempre demonstrou a mim e a toda nossa família através de palavras e atitudes. Você é e sempre será um grande exemplo de ser-humano para mim.

Agradeço a meus tios, Roberto e Renata, por, desde os tempos da graduação, sempre terem me incentivado e me ajudado para tornar possível minha jornada nos estudos.

Agradeço aos meus amigos Lamartine Júnior, Leonardo Souza, Lucia Souza, Mateus Ramos e Patrick Dudeck, por estarem (virtualmente) presentes em todos os bons e maus momentos que enfrentei. A distância de milhares de quilômetros entre mim e cada um de vocês torna-se pequena diante da grandeza de nossa amizade.

Agradeço aos colegas José Lucas Borges e Lucas Corona, pela parceria e companhia durante os estudos na biblioteca e a realização dos trabalhos em grupo. Nossa amizade e convivência foram uma oportunidade de aprendizado e uma forma de diminuir as dificuldades da vida acadêmica.

Agradeço à amiga de longa data e colega do PPGEC, Barbara Chagas, pelo convívio, apoio e momentos compartilhados desde a época da graduação. Nossa amizade foi um dos grandes presentes que a Engenharia me ofereceu.

Agradeço aos professores do PPGEC/UFRGS, pelo conhecimento transmitido e contribuição com a minha formação.

Agradeço à Professora Drª. Letícia Miguel, pela excelente orientação ao longo da realização desse trabalho. A atenção, o comprometimento e a dedicação da Profª. Letícia para com seus alunos e orientandos, fazem dela um exemplo notável de profissional e de pessoa.

Agradeço ao PPGEC e à UFRGS, pela oportunidade de cursar um programa de pós-graduação de excelência e pela infraestrutura oferecida.

Agradeço à CAPES, pela bolsa de estudos, que foi fundamental para a minha dedicação exclusiva às atividades acadêmicas.

A educação é o único caminho para emancipar o homem.

Leonel Brizola

RESUMO

SOUZA, O. A. P. **Elaboração de um algoritmo de otimização aplicado à engenharia estrutural: *Circle-Inspired Optimization Algorithm* (CIOA).** 2021. Dissertação (Mestrado em Engenharia Civil) – Programa de Pós-Graduação em Engenharia Civil, Universidade Federal do Rio grande do Sul, Porto Alegre.

A otimização estrutural é atualmente uma importante área de estudo para a engenharia, visto que existe uma demanda crescente de que os projetos estruturais desenvolvidos sejam capazes de minimizar o uso de recursos e o custo da obra e, ao mesmo tempo, maximizar os parâmetros relacionados à qualidade e à resistência da estrutura. Com o avanço da tecnologia computacional, muitos algoritmos meta-heurísticos foram desenvolvidos para resolver problemas de otimização com rapidez e precisão. Todavia, a randomização presente nos algoritmos meta-heurísticos, aliada à impossibilidade de garantir que a melhor solução seja sempre alcançada, exigem um estudo constante baseado no desenvolvimento de novos algoritmos mais eficientes e no aprimoramento dos algoritmos já formulados. No presente trabalho, um novo algoritmo meta-heurístico de otimização foi proposto: O *Circle-Inspired Optimization Algorithm* – (CIOA), inspirado em formulações matemáticas simples relacionadas à circunferência trigonométrica e incorporando alguns aspectos de outros algoritmos meta-heurísticos. O CIOA foi validado através da otimização global de 15 funções matemáticas e, em seguida, foi aplicado em problemas clássicos de otimização paramétrica e de forma de treliças. Análises estatísticas demonstraram a eficácia do CIOA, comparando o seu desempenho com o de outros algoritmos meta-heurísticos já consagrados e amplamente utilizados em problemas de otimização. Os resultados apontaram ainda que as maiores vantagens do *Circle-Inspired Optimization Algorithm* são o número reduzido de parâmetros a serem definidos pelo usuário e o baixo tempo computacional de operação.

Palavras-chave: *Otimização estrutural; Algoritmos meta-heurísticos; Treliças; Circle-Inspired Optimization Algorithm.*

ABSTRACT

SOUZA, O. A. P. **Elaboração de um algoritmo de otimização aplicado à engenharia estrutural: *Circle-Inspired Optimization Algorithm* (CIOA).** 2021. Dissertação (Mestrado em Engenharia Civil) – Programa de Pós-Graduação em Engenharia Civil, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Structural optimization is currently an important area of study for engineering, as there is a growing demand that the structural projects developed can minimize the use of resources and the cost of the work and, at the same time, maximize the parameters related to the quality and strength of the structure. With the advancement of computational technology, many metaheuristic algorithms were developed to solve optimization problems quickly and accurately. However, the randomization present in the metaheuristic algorithms, combined with the impossibility of guaranteeing that the best solution is always achieved, requires a constant study based on the development of new and more efficient algorithms and on the improvement of the algorithms already formulated. In this work, a new metaheuristic optimization algorithm was proposed: The Circle-Inspired Optimization Algorithm – (CIOA), inspired by simple mathematical formulations related to trigonometric circumference and incorporating some aspects of other metaheuristic algorithms. The CIOA was validated through the global optimization of 15 mathematical functions and then, it was applied in classical problems of shape and size optimization of trusses. Statistical analyzes have demonstrated the effectiveness of the CIOA, comparing its performance with that of famous and widely used metaheuristic optimization algorithms. The results showed that the biggest advantages of the Circle-Inspired Optimization Algorithm are the reduced number of parameters to be defined by the user and the fast computational time of operation.

Key-words: *Structural Optimization; Metaheuristic Algorithms; Trusses; Circle-Inspired Optimization Algorithm.*

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ABC	*Artificial Bee Colony*
BA	*Bat Algorithm*
BOA	*Butterfly Optimization Algorithm*
CIOA	*Circle-Inspired Optimization Algorithm*
CS	*Cuckoo Search*
DE	*Differential Evolution*
FA	*Firefly Algorithm*
FPA	*Flower Pollination Algorithm*
GSA	*Gravitational Search Algorithm*
GWO	*Grey Wolf Optimizer*
HCMR	*Harmony Search Considering Rate*
HM	*Harmony Memory*
HS	*Harmony Search*
KH	*Krill Herd*
PAR	*Pitch Adjusting Rate*
PSO	*Particle Swarm Optimization*
SGA	*Search Group Algorithm*
WOA	*Whale Optimization Algorithm*

LISTA DE SÍMBOLOS

I – LETRAS ROMANAS MAIÚSCULAS

$\boldsymbol{A}$	Vetor de coeficientes do *Whale Optimization Algorithm*
C	Ponto qualquer sobre uma circunferência
$\boldsymbol{C_C}$	Vetor de coeficientes do *Whale Optimization Algorithm*
C_{X1}	Coordenada do ponto C no eixo X_1
C_{X2}	Coordenada do ponto C no eixo X_2
$\boldsymbol{D}$	Distância entre um agente específico e o melhor agente de busca no *Whale Optimization Algorithm*
$\boldsymbol{D'}$	Distância entre uma baleia e a presa no *Whale Optimization Algorithm*
E	Operador de valor médio do *Search Group Algorithm*
J	Número de restrições de igualdade
K	Número de restrições de desigualdade
L_b	Limite inferior de uma variável
L_{b1}	Novo limite inferior de uma variável no *Circle-Inspired Optimization Algorithm*
M	Número de funções objetivo
N_{ag}	Número de agentes de pesquisa
O	Centro de uma circunferência
$O_{1,2}$	Centro do círculo durante o movimento de um agente do ponto 1 ao ponto 2 no *Circle-Inspired Optimization Algorithm*
$O_{2,3}$	Centro do círculo durante o movimento de um agente do ponto 2 ao ponto 3 no *Circle-Inspired Optimization Algorithm*
O_{X1}	Coordenada do ponto O no eixo X_1
O_{X2}	Coordenada do ponto O no eixo X_2
$\boldsymbol{P}$	População inicial do *Search Group Algorithm*
$\boldsymbol{R}$	Grupo de pesquisa do *Search Group Algorithm*

R_1	Melhor *design* do grupo de pesquisa no *Search Group Algorithm*
R_{ng}	Pior *design* do grupo de pesquisa no *Search Group Algorithm*
T^+	Soma das classificações onde um primeiro algoritmo tem desempenho melhor do que um segundo algoritmo no Teste dos Postos Sinalizados de Wilcoxon
T^-	Soma das classificações onde um primeiro algoritmo tem desempenho pior do que um segundo algoritmo no Teste do Postos Sinalizados de Wilcoxon
U	Variável uniforme do *Search Group Algorithm*
U_b	Limite superior de uma variável
U_{b1}	Novo limite superior de uma variável no *Circle-Inspired Optimization Algorithm*
$\boldsymbol{X}^*$	Vetor de posição da melhor solução obtida no *Whale Optimization Algorithm*
$\boldsymbol{X_{rand}}$	Vetor de posição aleatória no *Whale Optimization Algorithm*

II – LETRAS ROAMANAS MINÚSCULAS

$\boldsymbol{a}$	Vetor de coeficientes do *Whale Optimization Algorithm*
b	Constante própria do *Whale Optimization Algorithm*
b_s	Parâmetro próprio do *Search Group Algorithm*
c_r	Constante do *Circle-Inspired Optimization Algorithm*
d	Dimensão de movimento do *Firefly Algorithm*
d_i	Diferença entre resultados de dois algoritmos no Teste dos Postos Sinalizados de Wilcoxon
f_1	Primeira frequência natural de uma estrutura
f_2	Segunda frequência natural de uma estrutura
f_3	Terceira frequência natural de uma estrutura
$f_i(\boldsymbol{x})$	Funções objetivo
g_B	Melhor desempenho global do sistema no *Particle Swarm Optimization*
$g_k(\boldsymbol{x})$	Restrições de desigualdade

$glob_{It}$	Parâmetro que define a proporção de iterações com pesquisa global no *Circle-Inspired Optimization Algorithm*
$h_j(\boldsymbol{x})$	Restrições de igualdade
it_{global}^{max}	Número máximo de iterações para a fase global no *Search Group Algorithm*
k	Número da iteração presente
l	Valor aleatório entre -1 e 1 no *Whale Optimization Algorithm*
n	Número de variáveis de projeto
n_g	Número de membros do grupo de pesquisa no *Search Group Algorithm*
n_{pop}	Número de indivíduos da população no *Search Group Algorithm*
p_B	Posição da partícula que levou ao seu melhor desempenho no *Particle Swarm Optimization*
r	Raio de uma circunferência
$\boldsymbol{r}$	Vetor de raios do *Circle-Inspired Optimization Algorithm*
r_1	Primeiro elemento do vetor de raios do *Circle-Inspired Optimization Algorithm*
$r_{1,2}$	Raio que rege o movimento de um agente do ponto 1 ao ponto 2 no *Circle-Inspired Optimization Algorithm*
$r_{2,3}$	Raio que rege o movimento de um agente do ponto 2 ao ponto 3 no *Circle-Inspired Optimization Algorithm*
$r_{N_{ag}}$	Último elemento do vetor de raios do *Circle-Inspired Optimization Algorithm*
$\boldsymbol{r_a}$	Vetor de valores aleatórios entre 0 e 1 do *Whale Optimization Algorithm*
r_f	Distância entre um vaga-lume e uma fonte de luz no *Firefly Algorithm*
r_j	$j^{ésimo}$ elemento do vetor de raios do *Circle-Inspired Optimization Algorithm*
$\boldsymbol{r_{new}}$	Novo vetor de raios após redução do limite de variáveis no *Circle-Inspired Optimization Algorithm*
$rand_1$	Variável aleatória entre 0 e 1 do *Circle-Inspired Optimization Algorithm*
$rand_2$	Variável aleatória entre 0 e 1 do *Circle-Inspired Optimization Algorithm*
$rand_3$	Variável aleatória entre 0 e 1 do *Circle-Inspired Optimization Algorithm*

$rand_4$	Variável aleatória entre 0 e 1 do *Circle-Inspired Optimization Algorithm*
t	Número de iterações
t_d	Variável que controla a distância onde um novo indivíduo é gerado no *Search Group Algorithm*
v_i	Velocidade de uma partícula no *Particle Swarm Optimization*
$\boldsymbol{x}$	Vetor de variáveis de projeto
x_{2i}	Variável de uma dimensão par do *Circle-Inspired Optimization Algorithm*
x_{2i-1}	Variável de uma dimensão ímpar do *Circle-Inspired Optimization Algorithm*
x_i	Componentes do vetor de variáveis de projeto
$x_{i_{best}}$	Variável na dimensão i da melhor solução produzida no *Circle-Inspired Optimization Algorithm*
$x_{i,k}$	$k^{ésimo}$ componente de uma variável do $i^{ésimo}$ agente de pesquisa
x_j^{mut}	j-ésima variável de um indivíduo mutado no *Search Group Algorithm*
x_j^{new}	Família gerada por um membro do grupo de pesquisa do *Search Group Algorithm*
x_{max_i}	Limite superior da i-ésima variável de projeto
x_{min_i}	Limite inferior da i-ésima variável de projeto

III – LETRAS GREGAS

α	Variável que controla o tamanho da perturbação no *Search Group Algorithm*
α_r	Parâmetro de randomização do *Search Group Algorithm*
α_{min}	Valor mínimo admissível para a variável α no *Search Group Algorithm*
β	Atratividade de um vaga-lume do *Firefly Algorithm*
β_0	Atratividade de um vaga-lume do *Firefly Algorithm* quando $r = 0$
γ	Coeficiente de absorção de luz do *Firefly Algorithm*
ε	Vaiável aleatória do *Search Group Algorithm*
ϵ_i	Vetor de números aleatórios extraído de uma distribuição gaussiana ou uniforme no *Firefly Algorithm*

θ	Ângulo que rege os agentes do *Circle-Inspired Optimization Algorithm*
ρ	Massa específica de uma estrutura
σ	Operador de desvio padrão do *Search Group Algorithm*
σ_f	Tensão de flambagem
φ_1	Componente cognitivo do *Particle Swarm Optimization*
φ_2	Componente social do *Particle Swarm Optimization*

SUMÁRIO

1 INTRODUÇÃO

O processo de otimização está presente no cotidiano da humanidade. Empresários que buscam maximizar os lucros de suas empresas, industriais que procuram desenvolver em suas fábricas produtos com a maior qualidade possível, até mesmo pessoas que tentam aproveitar o tempo livre de forma mais produtiva, podem ser citados como exemplos de otimização. De acordo com Yang (2010c), em todas estas atividades, tenta-se atingir um objetivo específico, ou promover a otimização de algo como lucro, qualidade e tempo. Desta forma, em termos gerais, a otimização pode ser definida como o ato de obter o melhor resultado possível mediante circunstâncias específicas.

No ramo da Engenharia, o processo de otimização assume grande relevância toda vez em que há a necessidade de o engenheiro tomar decisões técnicas que o levem a um objetivo final. Rao (2009) aponta que, geralmente, o objetivo é minimizar custos e/ou maximizar benefícios. Como estes fatores podem ser expressos em função de variáveis de decisão, o processo de otimização pode ser formalizado como o ato de encontrar as variáveis – ou condições – que fornecem o valor mínimo ou máximo de uma determinada função ou conjunto de funções.

Contudo, de acordo com Borges (2013), grande parte dos problemas de otimização na engenharia estrutural são não-lineares e possuem inúmeras restrições que precisam ser atendidas simultaneamente. Yang (2010c) salienta que a maioria dos algoritmos clássicos/determinísticos de otimização são baseados em gradiente, ou seja, utilizam os valores da função e suas derivadas. Este processo é eficiente em problemas unimodais suaves, mas não funciona tão bem quando existem descontinuidades na função objetivo. Neste último caso, é preferível a utilização de um algoritmo que não seja baseado em gradiente. Diante disso, surge a necessidade da implementação e utilização de algoritmos meta-heurísticos de otimização que sejam capazes de produzir, de forma eficiente, resultados confiáveis e precisos para a solução de problemas complexos.

1.1 JUSTIFICATIVA

A crescente competitividade existente no mercado da construção civil tem exigido cada vez mais que o processo de otimização seja levado em consideração em todas as etapas: desde o

projeto à finalização da obra, promovendo assim uma redução da utilização de recursos/materiais, a minimização de gastos e a diminuição do tempo total da obra. Entretanto, estas exigências devem ser atendidas de forma que não interfiram negativamente na qualidade e na segurança.

Para ajudar a atender essas demandas, inúmeros algoritmos meta-heurísticos de otimização foram desenvolvidos nas últimas décadas e sua aplicação em problemas de otimização estrutural é amplamente utilizada no meio acadêmico e/ou profissional. De acordo com Yang (2010c), os algoritmos meta-heurísticos são algoritmos estocásticos que utilizam uma certa troca entre a randomização e a pesquisa local. Em geral, esses algoritmos trabalham com um processo de tentativa e erro, o que não garante que a melhor solução seja sempre obtida, mas sim uma aproximação dessa solução em que a precisão pode depender, por exemplo, da complexidade do problema.

Uma vez exposta essa limitação dos algoritmos meta-heurísticos na capacidade de busca de um ótimo global com total precisão, o estudo e aprimoramento de algoritmos já existentes bem como o desenvolvimento de novos algoritmos torna-se altamente relevante no campo da engenharia estrutural, como tentativa de se produzir ferramentas cada vez mais eficientes e capazes de resolverem problemas complexos de otimização com boa precisão e em um tempo computacional de execução aceitável.

1.2 OBJETIVOS

O objetivo principal do presente trabalho consiste na elaboração de um novo algoritmo meta-heurístico de otimização, em linguagem MATLAB, cuja aplicação seja eficiente na solução de problemas de otimização de engenharia estrutural, especialmente em casos de otimização paramétrica e/ou de forma de treliças 2D e 3D submetidas a diferentes tipos de restrições, de natureza estática e dinâmica.

Para cumprir o objetivo principal, alguns objetivos específicos, listados a seguir, precisam ser atingidos:

a) Implementar, em linguagem MATLAB, códigos de otimização de funções matemáticas padrão (*benchmark*);

b) Validar o novo algoritmo proposto, através da solução de problemas de otimização de funções *benchmark;*

c) Promover uma análise de sensibilidade do algoritmo, avaliando a influência dos parâmetros próprios no desempenho do mesmo;

d) Implementar e validar códigos de otimização paramétrica e de forma de treliças planas e espaciais sujeitas a diferentes tipos de restrições, de natureza estática e dinâmica;

e) Avaliar a capacidade do algoritmo criado, quando aplicado a problemas de engenharia, a saber: otimização paramétrica e de forma de treliças planas e espaciais sujeitas a diferentes tipos de restrições;

f) Comparar o desempenho do novo algoritmo proposto com o desempenho de algoritmos consagrados pela literatura, através de diferentes tipos de análises.

1.3 ORGANIZAÇÃO DO TRABALHO

Em relação à organização do trabalho, foi adotada a seguinte divisão de capítulos:

No **primeiro capítulo** tem-se a introdução do tema da pesquisa, apresentando a justificativa e os objetivos do presente trabalho.

O **segundo capítulo** dedica-se à revisão bibliográfica geral, contendo os principais trabalhos realizados na linha de pesquisa de otimização meta-heurística.

Em seguida, o **terceiro capítulo** contém a fundamentação teórica, no qual são apresentadas informações básicas do processo de otimização estrutural; formulações de cinco dos principais algoritmos meta-heurísticos desenvolvidos nas últimas décadas, utilizados na validação do novo algoritmo desenvolvido; formulação do Teste dos Postos Sinalizados do Wilcoxon, utilizado para uma análise estatística avançada de algoritmos meta-heurísticos.

O **quarto capítulo** consiste na apresentação e detalhamento da formulação do *Circle-Inspired Optimization Algorithm* (CIOA): O novo algoritmo meta-heurístico de otimização desenvolvido neste trabalho.

No **quinto capítulo** tem-se a validação e uma análise de sensibilidade do CIOA através da otimização de funções *benchmark* e demais funções matemáticas.

No **sexto capítulo**, o novo algoritmo proposto, CIOA, é aplicado em problemas clássicos de otimização paramétrica e de forma de treliças sujeitas a diversos tipos de restrições, sendo seu desempenho comparado a demais algoritmos meta-heurísticos de otimização.

No **sétimo capítulo**, o algoritmo desenvolvido é aplicado em um estudo de caso envolvendo uma estrutura realista, uma torre de linha de transmissão de energia, sujeita a diversas restrições.

No **oitavo capítulo**, é aplicado o Teste dos Postos Sinalizados de Wilcoxon para a comparação do CIOA com demais algoritmos através de uma análise estatística avançada.

No **nono capítulo** são apresentadas as conclusões da presente pesquisa e apontadas sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

De acordo com Mirjalili e Lewis (2016), os algoritmos meta-heurísticos de otimização estão se tornando cada vez mais comuns em aplicações de engenharia devido a, basicamente, quatro fatores principais:

a) Formulação simples e fácil implementação;

b) Ausência da necessidade de informações do gradiente da função;

c) Capacidade de fuga de ótimos locais;

d) Possibilidade do uso em um amplo leque de problemas.

A seguir, é apresentada uma breve revisão de alguns dos principais algoritmos meta-heurísticos de otimização desenvolvidos nos últimos anos, além de trabalhos que se utilizaram desses algoritmos para a solução de problemas de otimização estrutural. Para os algoritmos efetivamente utilizados neste trabalho – *Particle Swarm Optimization* (PSO), *Harmony Search* (HS), *Firefly Algorithm* (FA), *Search Group Algorithm* (SGA) e *Whale Optimization Alorithm* (WOA) – uma apresentação detalhada com a formulação matemática é realizada na seção 3.3.

Kennedy e Eberhart (1995) desenvolveram o *Particle Swarm Optimization* (PSO), onde os agentes de pesquisa são considerados partículas que se movem por um espaço de busca. O comportamento desse conjunto de partículas pode ser comparado, por exemplo, ao comportamento social de um enxame de pássaros em busca de alimento. Durante o processo de otimização, numa determinada iteração cada agente de pesquisa "lembra" em qual posição do espaço de busca obteve o melhor resultado até então e compara esse valor com o valor atual. Além disso, cada agente conhece a melhor posição geral que um membro do grupo encontrou, bem como o valor correspondente a essa posição. O algoritmo foi validado através da sua aplicação, com bons resultados, em problemas relacionados à rede neural e à otimização da função f_6 de Schaffer, que possui diversos mínimos locais.

Gomes (2011) utilizou o algoritmo *Particle Swarm Optimization* para a otimização paramétrica e de forma de treliças submetidas a restrições de frequência e ressaltou as dificuldades de convergência que problemas dinâmicos de otimização podem apresentar, uma vez que ao minimizar a massa da estrutura, os modos de vibração podem sofrer alterações que causam mudanças substanciais na frequência da estrutura. Quatro modelos estruturais clássicos da

literatura foram otimizados e os resultados demonstraram que o PSO possui um desempenho competitivo em relação a outros métodos de otimização.

Storn e Price (1997) propuseram o algoritmo *Differential Evolution* (DE), algoritmo heurístico onde a ideia principal é um esquema capaz de gerar vetores. Basicamente, cada novo indivíduo (ou vetor) representa uma possível solução e é gerado através um vetor de diferença ponderada entre dois indivíduos da população que é adicionado a um terceiro indivíduo. Ao longo das iterações, os indivíduos podem sofrer mutações ou cruzamentos. O algoritmo foi avaliado em nove problemas de otimização de funções a apresentou um bom desempenho.

Geem et al. (2001) criaram o algoritmo *Harmony Search* (HS), inspirado na atuação de um grupo musical que busca a harmonia perfeita produzida por diversos instrumentos simultaneamente, de forma que os valores de cada variável de projeto que influenciam na avaliação da função objetivo são comparados aos sons de cada um dos instrumentos que compõem a harmonia musical. O algoritmo foi validado por meio da sua utilização em três problemas clássicos de otimização: O problema do caixeiro-viajante, que consiste em encontrar o menor caminho que um vendedor deve tomar para visitar cada uma, entre várias cidades, precisamente uma vez; um problema de otimização não linear com duas restrições de desigualdade e, por fim, um projeto de uma rede de tubulações para abastecimento de água numa cidade Vietnamita. Os resultados obtidos mostraram que o algoritmo tem um alto potencial na resolução de problemas de variáveis contínuas e problemas combinatórios.

Lee e Geem (2004) propuseram um novo método de otimização estrutural baseado no algoritmo *Harmony Search* (HS) e o aplicaram na otimização de estruturas de treliças, comparando-o com métodos matemáticos convencionais ou abordagens baseadas em algoritmos genéticos. Os resultados do estudo revelaram que a estratégia de busca baseada no algoritmo HS foi capaz de resolver problemas de otimização paramétrica com variáveis contínuas, contudo, sem se limitar apenas a essa modalidade de problemas, uma vez que o algoritmo também pode ser facilmente utilizado em problemas com variáveis discretas e com variáveis relacionadas à configuração da estrutura. Posteriormente, Lee e Geem (2005) ampliaram o estudo do algoritmo HS, analisando o desempenho dele na minimização de funções com e sem restrições e em problemas de otimização estrutural, tais quais projetos de vasos de pressão e de viga soldada, otimização paramétrica e de forma de treliças e calibração dos parâmetros de um modelo hidrológico. Os resultados denotaram a eficácia e robustez do algoritmo como ferramenta de pesquisa global.

Karaboga (2005) desenvolveu o *Artificial Bee Colony* (ABC), algoritmo meta-heurístico inspirado no comportamento social das abelhas. Cada ciclo do algoritmo consiste nas seguintes etapas: mover as abelhas empregadas e observadoras até as fontes de alimento e calcular a suas quantidades de néctar; determinar as abelhas exploradoras e direcioná-las para possíveis fontes de alimentos. A posição de uma fonte de alimento representa uma possível solução para o problema a ser otimizado. A quantidade de néctar de uma fonte alimentar corresponde à qualidade da solução representada por aquela fonte alimentar. Posteriormente, Karaboga e Barsturk (2007) implementaram uma versão modificada do algoritmo ABC para ser utilizada também em problemas de otimização com restrições. O desempenho da nova versão do algoritmo é comparado com outros tradicionais algoritmos disponíveis na literatura e os resultados mostram que o ABC possui uma boa eficiência na solução de problemas de otimização restrita.

Yang (2009a) propôs o *Firefly Algorithm* (FA), onde é feita uma analogia entre o processo de otimização e a forma como os vagalumes usam suas características luminescentes para atrair parceiros e possíveis presas. No FA, os agentes de pesquisa possuem uma atratividade relativa, de acordo com a distância entre eles e proporcional ao seu brilho, sendo este em função da distância do agente em relação ao ótimo global da função objetivo. O desempenho do algoritmo foi avaliado diante da otimização de funções multimodal, os resultados mostraram que o FA tem um potencial de eficiência melhor que outros algoritmos, tais como, por exemplo, o PSO.

Borges (2013) aplicou os algoritmos *Harmony Search* (HS) e *Firefly Algorithm* (FA) na otimização de exemplos clássicos de treliças em 2D e 3D submetidas a diferentes tipos de restrições, efetivando a otimização paramétrica e de forma. No mesmo trabalho, os algoritmos também foram aplicados na otimização da estrutura de um projeto de engenharia adaptado. Os resultados comprovaram a eficiência de ambos os algoritmos, mas apontaram que o FA apresentou resultados um pouco melhores que o HS. Todavia, para um mesmo número de avaliações da função objetivo, o HS apresentou menor tempo computacional de execução que o FA.

Miguel e Fadel Miguel (2013) promoveram uma comparação entre os algoritmos *Harmony Search* (HS), *Artificial Bee Colony* (ABC) e *Firefly Algorithm* (FA). Os três algoritmos foram utilizados na otimização paramétrica e de forma de treliças planas e espaciais com diferentes tipos de restrições, bem como na otimização de um modelo estrutural realista, de uma torre de transmissão de 82 metros de altura que foi danificada durante um tufão no Japão. Os resultados

do trabalho apontaram a eficácia dos três algoritmos, com leve vantagem em favor do FA, que obteve resultados levemente melhores que os demais algoritmos. Todavia, em relação ao custo computacional, o algoritmo HS foi capaz de resolver os problemas em um tempo um pouco menor que os algoritmos ABC e FA.

Rashedi et al. (2009) desenvolveram o *Gravitational Search Algorithm* (GSA), onde o processo de otimização é baseado nas leis da gravidade e interações de massas. Os agentes de pesquisa do GSA são uma coleção de massas que interagem entre si de acordo com as leis da gravidade Newtoniana. Desta forma, a atração gravitacional promove um movimento global de todos os agentes em direção aos agentes com massas mais pesadas sendo que estas, representam as boas soluções ou soluções próximas à solução ótima. Ao final, espera-se que todas as massas sejam atraídas pela massa mais pesada, que representará a solução ótima do problema. O GSA foi validado através da solução de dezenas de funções *benchmark* e seus resultados se mostraram bons quando comparados com aqueles obtidos através de outros algoritmos.

Yang e Deb (2009) criaram o algoritmo *Cuckoo Search* (CS), inspirado no comportamento parasitário que algumas espécies de cuco usam durante a sua reprodução combinado com o comportamento de voos de Lévy, característico de algumas espécies de pássaros. O CS pode ser resumido através de três regras básicas: Cada cuco põe um ovo por vez e escolhe um ninho hospedeiro aleatório para depositá-lo; os melhores ninhos com ovos de alta qualidade (correspondentes às melhores soluções) serão transportados para novas gerações; existe uma certa probabilidade de a ave hospedeira descobrir o ovo depositado em seu ninho e jogá-lo fora, neste caso, o ninho será substituído por um novo ninho (situação correspondente à busca aleatória de novas soluções). O algoritmo foi testado mediante a otimização de 11 funções *benchmark* e os resultados obtidos foram comparados com o PSO e com algoritmos genéticos, demonstrando a eficiência do CS.

Yang (2010a) propôs o *Bat Algorithm* (BA), baseado no comportamento dos morcegos. O algoritmo possui notáveis semelhanças com o PSO e o FA, entretanto, no BA os agentes de pesquisa atualizam suas posições e velocidades através de frequências emitidas por pulsos sonoros, tal qual fazem os morcegos durante o ataque a suas presas. O algoritmo foi validado através da otimização de funções *benchmark* e os resultados mostraram que o BA é tão ou até melhor eficiente do que outros algoritmos de otimização.

Yang (2012) propôs o *Flower Pollination Algorithm* (FPA), inspirado no processo de polinização de flores, que é subdividida entre polinização cruzada e autopolinização. A polinização cruzada é considerada um processo de polinização global e necessita de polinizadores portadores de pólen que realizarão voos aleatórios enquanto a autopolinização é considerada uma polinização local. A escala do processo de polinização/otimização irá variar entre local e global através de uma probabilidade de troca. Além disso, o algoritmo também leva em consideração a constância floral, parâmetro que é representado como a probabilidade de reprodução e é proporcional à semelhança das duas flores envolvidas no processo de polinização cruzada. O algoritmo foi utilizado para a solução de problemas de otimização envolvendo dez funções de teste, além de um problema de projeto não linear com restrições. Os resultados mostraram que o FPA tem um desempenho compatível ou até mesmo superior a algoritmos famosos, como, por exemplo, o PSO.

Bekdas et al. (2015) aplicaram o *Flower Pollination Algorithm* (FPA) em problemas de otimização de treliças planas e espaciais. Para tal, é incluída no mecanismo de busca do algoritmo uma estratégia adaptativa para o tratamento de restrições. A análise de três problemas clássicos de otimização de treliça demonstrou que o FPA apresenta resultados competitivos quando comparado a outros algoritmos e alcança esses resultados com um número de análises estruturais relativamente menor. Isso ocorre porque na busca global os melhores conjuntos de variáveis são usados na geração de novas variáveis. Desta forma, a convergência do FPA para os valores ótimos é eficaz em comparação com os algoritmos que usam um processo de randomização simples para otimização global.

Gandomi e Alavi (2012) criaram o algoritmo *Krill Herd* (KH), cuja inspiração é o comportamento coletivo de Krills, um conjunto de espécies de invertebrados similares ao camarão. O algoritmo opera basicamente com três ações: o movimento induzido pela presença de outros indivíduos, a atividade de busca de comida e a difusão aleatória. Vinte funções *benchmark* foram otimizadas com o *Krill Herd* e os resultados obtidos foram comparados com os produzidos por outros algoritmos, denotando boa eficiência para o KH.

Mirjalili et al. (2014) desenvolveram o *Grey Wolf Optimizer* (GWO), algoritmo inspirado na hierarquia de liderança e no mecanismo de caça dos lobos cinzentos. O algoritmo, que opera em três etapas, foi testado na otimização de 29 funções *benchmark*, três problemas clássicos de engenharia e um estudo de caso real. Os resultados obtidos, especialmente nos problemas aplicados à engenharia, demonstraram que o GWO apresenta boa eficiência.

Gonçalves et al. (2015) propuseram o *Search Group Algorithm* (SGA), que é definido basicamente em cinco etapas: Geração da população inicial, seleção do grupo de pesquisa inicial, mutação do grupo de pesquisa, geração das famílias de cada membro do grupo de pesquisa e seleção do novo grupo de pesquisa. O principal diferencial do SGA em relação a outros algoritmos é que ele utiliza a estratégia de que quando melhor é o integrante do grupo de pesquisa, mais indivíduos ele gera em uma determinada iteração. Para a validação do algoritmo, seis problemas de otimização paramétrica, de forma e topológica de treliças foram implementados e solucionados por meio do SGA, que obteve resultados tão bons ou melhores do que aqueles obtidos por outros algoritmos disponíveis na literatura.

Mirjalili e Lewis (2016) desenvolveram o *Whale Optimization Algorithm* (WOA), onde o processo de otimização foi inspirado na estratégia de caça da baleia jubarte. O algoritmo é, portanto, regido por equações que descrevem os movimentos de ataque da baleia jubarte e a interação entre elas durante a caça de presas. O algoritmo foi validado através da otimização de 29 funções consideradas caso-padrão pela literatura, entre elas, funções unimodais, multimodais e funções com e sem um número fixo de dimensões. O algoritmo WOA também foi utilizado na solução de seis problemas de otimização aplicados à engenharia, sendo três deles referentes à otimização de treliças. Os resultados obtidos demonstraram que o algoritmo produziu resultados competitivos em relação a outros algoritmos consagrados pela literatura.

Kaveh e Zolghadr (2018) revisaram diferentes técnicas de otimização meta-heurística aplicadas em problemas de otimização de treliças sujeitas a restrições de frequência natural. Em seu trabalho, são comparados os resultados obtidos por diferentes autores utilizando os mais diversos algoritmos meta-heurísticos em seis problemas de otimização de treliças envolvendo ações dinâmicas. Os resultados obtidos mostram o quanto a solução do problema pode variar a depender do algoritmo utilizado.

Arora e Singh (2019) propuseram o *Butterfly Optimization Algorithm* (BOA), baseado no comportamento das borboletas, que utilizam do olfato para determinar a direção de uma fonte de alimento ou de um parceiro para acasalamento. A formulação do algoritmo considera que: Todas as borboletas emitem alguma fragrância que atrai as demais borboletas; cada borboleta se moverá aleatoriamente ou em direção à melhor borboleta, sendo essa, a que possui maior intensidade de fragrância; A intensidade do estímulo de uma borboleta é determinada levando-se em conta a função objetivo. O algoritmo foi executado para solucionar 30 problemas de otimização de funções *benchmark* e três problemas clássicos de engenharia, a saber: projeto de

mola, projeto de viga soldada e projeto de trem de engrenagens. Os resultados indicaram que o BOA tem um desempenho superior a grande parte dos algoritmos com o qual foi comparado.

No Quadro 2.1, apresenta-se um breve resumo dos algoritmos tratados neste capítulo, onde são apontadas as vantagens e características mais relevantes de cada um destes algoritmos.

Quadro 2.1 – Vantagens de cada algoritmo meta-heurístico

Algoritmo	Principais vantagens
PSO	Fácil implementação, número reduzido de parâmetros a serem definidos pelo usuário.
DE	Alta eficiência em problemas de otimização estrutural
HS	Aplicabilidade em diversos tipos de problemas de otimização.
ABC	Formulação simples, aplicabilidade em diversos tipos de problemas de otimização
FA	Aplicabilidade em diversos tipos de problemas de otimização
GSA	Bom desempenho na otimização de funções lineares e não-lineares
CS	Alta eficiência em problemas de otimização multimodal, número reduzido de parâmetros a serem definidos pelo usuário
BA	Aplicabilidade em diversos tipos de problemas de otimização
FPA	Versatilidade para a aplicação em diversos tipos de problemas de otimização
KH	Número reduzido de parâmetros a serem definidos pelo usuário.
GWO	Elevada eficiência em diversos tipos de problemas de otimização, incluindo problemas multimodais e com restrições.
SGA	Alta precisão em problemas de otimização estrutural, baixo tempo computacional de operação.
WOA	Alta precisão em diversos tipos de problemas de otimização, número reduzido de parâmetros a serem definidos pelo usuário, baixo tempo computacional de operação.
BOA	Formulação simples, bom desempenho em diferentes tipos de problemas de otimização

3 FUNDAMENTAÇÃO TEÓRICA

3.1 CONCEITOS GERAIS DE OTIMIZAÇÃO

Um problema genérico de otimização pode ser matematicamente formulado conforme apresentado na Equação 3.1:

$$\begin{gathered}\text{Minimize } f_i(\boldsymbol{x}), \quad (i = 1,2,\dots,M), \\ \text{Sujeito às restrições:} \\ h_j(\boldsymbol{x}) = 0, \quad (j = 1,2,\dots,J) \\ g_k(\boldsymbol{x}) \leq 0, \quad (k = 1,2,\dots,K) \\ \text{Sendo: } \boldsymbol{x} = (x_1, x_2, \dots, x_n)\end{gathered} \tag{3.1}$$

na qual os componentes x_i de $\boldsymbol{x}$ são as **variáveis de projeto**; as funções $f_i(\boldsymbol{x})$ são as **funções objetivo** a serem otimizadas; $h_j(\boldsymbol{x})$ e $g_k(\boldsymbol{x})$ são, respectivamente, as **restrições** de igualdade e desigualdade; J, K e M são, respectivamente, o número de restrições de igualdade, o número de restrições de desigualdade e o número de funções objetivo. É importante ressaltar que as restrições de desigualdade também podem ser formuladas da forma $g_k(x) \geq 0$ e que os problemas de otimização também podem ser formulados com o objetivo de maximizar uma função ou conjunto de funções (YANG, 2010c).

De acordo com Borges (2013), as **variáveis de projeto** são os parâmetros do problema que podem ser alterados para a otimização do sistema. Estas variáveis são classificadas como contínuas quando podem assumir qualquer valor real ou como discretas quando são limitadas a assumirem apenas valores pré-estabelecidos. A **função objetivo** é responsável por quantificar aquilo que será otimizado e é expressa em função das variáveis de projeto. O problema de otimização consiste em encontrar o mínimo ou o máximo para a função objetivo.

As **restrições** surgem sempre que as variáveis de projeto não podem ser escolhidas arbitrariamente, ou seja, elas devem satisfazer certos requisitos funcionais especificados no problema. De acordo com Rao (2009), as restrições que representam limitações apenas nas variáveis de projeto são denominadas restrições laterais, e são comumente apresentadas como na Equação 3.2

$$x_{min_i} \leq x_i \leq x_{max_i} \tag{3.2}$$

Já as restrições que representam limitações no comportamento ou desempenho do sistema como um todo, são chamadas de restrições de comportamento e podem ser classificadas em restrições de igualdade e restrições de desigualdade. Uma restrição de igualdade pode ser expressa de acordo com a Equação 3.3

$$h_j(\boldsymbol{x}) = 0, \quad (j = 1,2, \dots, J) \tag{3.3}$$

Enquanto uma restrição de desigualdade pode ser expressa conforme a Equação 3.4

$$g_k(\boldsymbol{x}) \leq 0, \quad (j = 1,2, \dots, K) \tag{3.4}$$

Em relação ao seu estado, a restrição pode ser classificada como ativa ou inativa. Borges (2013) aponta que uma restrição está ativa sempre que:

$$g_k(\boldsymbol{x}) = 0 \tag{3.5}$$

E uma restrição está inativa quando:

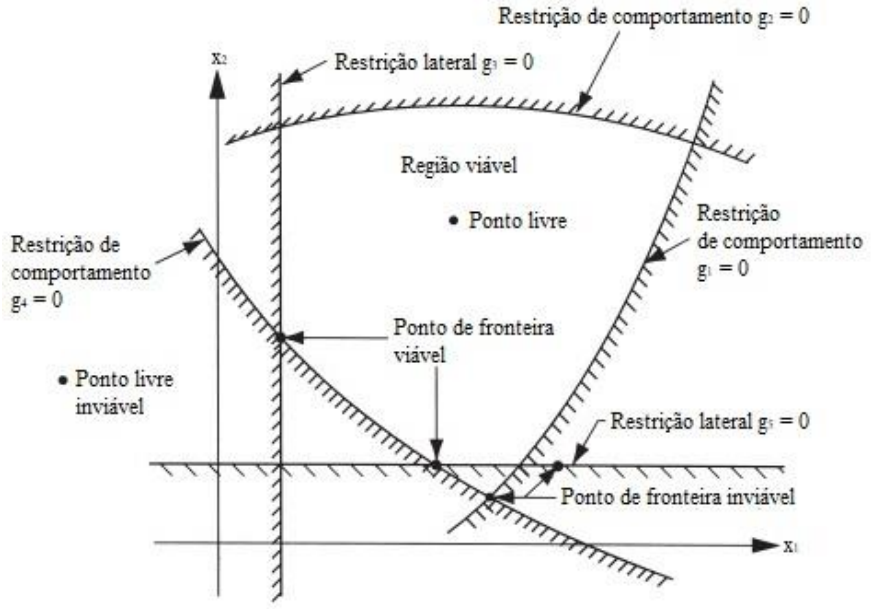
$$g_k(\boldsymbol{x}) > 0 \text{ ou } g_k(\boldsymbol{x}) < 0 \tag{3.6}$$

É importante salientar que restrições inativas não influenciam diretamente no resultado do problema, tornando-se então desnecessárias. Desta forma, desprezar restrições inativas poderia ser algo positivo, pois geraria um ganho no tempo computacional sem prejuízos à solução do problema. Contudo, conhecer de antemão quais restrições influenciam ou não o resultado da otimização é uma tarefa difícil e, dessa forma, todas as restrições são consideradas na solução de um problema. (BORGES, 2013).

Rao (2009) ilustra a otimização de uma função com restrições considerando um problema de otimização composto apenas por restrições de desigualdade do tipo $g_k(\boldsymbol{x}) \leq 0$. O conjunto dos valores de x que irá satisfazer $g_k(\boldsymbol{x}) = 0$ forma uma hipersuperfície denominada superfície de restrição que divide o espaço de projeto em duas regiões: Uma onde $g_k(\boldsymbol{x}) < 0$ e outra na qual $g_k(\boldsymbol{x}) > 0$. Desta forma os pontos localizados na região em que $g_k(\boldsymbol{x}) > 0$, que violam a restrição, são considerados inviáveis, enquanto os pontos situados onde $g_k(\boldsymbol{x}) < 0$ são viáveis. Os pontos situados exatamente na superfície são viáveis e tornarão a restrição $g_k(\boldsymbol{x})$ ativa. Finalmente, a coleção de todas as superfícies de restrições $g_k(\boldsymbol{x}) = 0$ com $k = 1,2, \dots, K$ é denominada como superfície de restrição composta. Na Figura 3.1, é apresentado um espaço de

projeto bidimensional hipotético, delimitado por cinco restrições, onde a região inviável é indicada pelo lado hachurado.

Figura 3.1 – Superfícies de restrição em um espaço de projeto bidimensional hipotético.



Fonte: Adaptada de Rao, 2009

Os pontos de projeto que não estão em nenhuma superfície de restrição, isto é, não tornam nenhuma restrição ativa, são denominados pontos livres.

3.2 CLASSIFICAÇÕES DE PROBLEMAS DE OTIMIZAÇÃO

Os problemas de otimização podem ser classificados de diferentes maneiras, as mais relevantes para a realização desse trabalho são apresentadas a seguir.

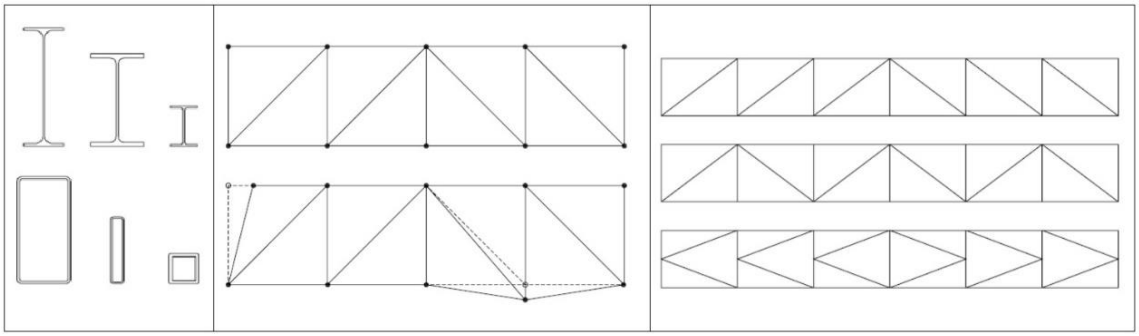
Em relação à **existência de restrições**, conforme já abordado na seção anterior, um problema de otimização pode ser classificado como restrito ou irrestrito, dependendo da existência ou não-existência de restrições.

Quanto à **linearidade**, a função objetivo pode ser linear ou não-linear. No caso de otimização com restrições, se todas as restrições são lineares o problema é classificado como linearmente restrito. Se além das restrições, a função objetivo também for linear, a otimização torna-se um problema de programação linear. Mas, se todas as restrições e a função objetivo são não-

lineares, o problema é classificado como de otimização não-linear, o que ocorre na maior parte dos problemas de engenharia. (YANG, 2010c).

Quanto ao **tipo de otimização estrutural**, três são as classificações: otimização paramétrica, otimização de forma e otimização topológica. Estas classificações são apresentadas na Figura 3.2, ilustradas em problemas de otimização de treliças.

Figura 3.2 – Problemas básicos de otimização de treliças: (a) Otimização paramétrica; (b) Otimização de forma; (c) Otimização topológica.



(a) (b) (c)

Fonte: Souza, 2009

De acordo com Borges (2013), na otimização paramétrica, são otimizados parâmetros referentes aos elementos estruturais que compõem a estrutura, como por exemplo, a área da seção transversal, mas a forma pré-definida da estrutura é totalmente mantida. Na otimização de forma, são modificadas as coordenadas dos nós da estrutura, alterando a forma final dela. Na otimização topológica, o material é distribuído dentro de um determinado domínio de projeto, podendo formar uma nova topologia.

3.3 FORMULAÇÃO DOS PRINCIPAIS ALGORITMOS META-HEURÍSTICOS DE OTIMIZAÇÃO

Nesta seção são apresentadas as formulações de cinco dos principais algoritmos meta-heurísticos de otimização desenvolvidos nas últimas décadas, a saber: *Particle Swarm Optimization* (PSO), *Harmony Search* (HS), *Firefly Algorithm* (FA), *Search Group Algorithm* (SGA) e *Whale Optimization Algorithm* (WOA). Esses algoritmos foram estudados de forma bastante aprofundada ao longo da realização do presente trabalho e suas formulações, em

grande parte, serviram de inspiração para a elaboração do novo algoritmo de otimização proposto nesta dissertação.

3.3.1 *Particle Swarm Optimization* (PSO)

O *Particle Swarm Optimization* (PSO) é um algoritmo meta-heurístico de otimização desenvolvido por Kennedy e Eberhart (1995), e surgiu de experiências que buscaram modelar, computacionalmente, o comportamento social de algumas espécies de animais, como por exemplo, pássaros e peixes. Em uma das primeiras experiências, os agentes eram considerados pássaros a prova de colisões e a intenção primordial do algoritmo era apenas simular a coreografia imprevisível de um bando de pássaros.

Posteriormente, variações dessa experiência foram implementadas, fazendo com que os agentes de pesquisa voassem em torno de um ponto específico do espaço de busca. Cada agente foi programado para avaliar sua posição atual em relação ao ponto que deve ser atingido e para ser capaz de "lembrar" o melhor valor atingido até então e a posição que resultou nesse valor. Além do mais, cada agente "conhecia" a melhor posição global que o grupo havia encontrado e o valor correspondente a essa posição. (KENNEDY E EBERHART, 1995).

De acordo com Serapião (2009), a experiência própria de cada agente de pesquisa do PSO é correspondente à aprendizagem individual (cognitiva) enquanto o conhecimento do comportamento dos seus vizinhos é comparado à transmissão cultural (social). Desta forma, pode-se dizer que a tomada de decisão de qualquer agente de pesquisa se dá em função do seu próprio desempenho no passado e do desempenho de alguns de seus agentes vizinhos.

Visando a implementação do algoritmo PSO, Serapião (2009) aponta que a posição x_i da partícula ou agente de pesquisa p_i na próxima iteração é estabelecida como uma influência aditiva da posição antiga e de uma velocidade calculada v_i. Esse processo é apresentado na Equação 3.7

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.7}$$

A velocidade calculada pode ser obtida através da Equação 3.8

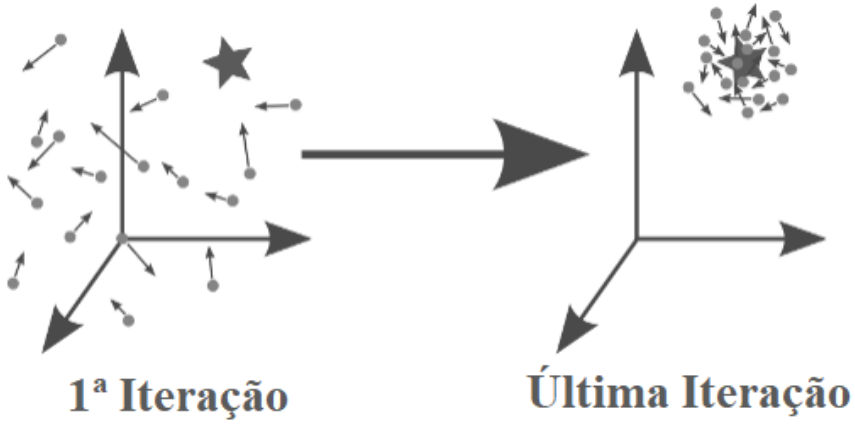
$$v_i(t+1) = v_i(t) + \varphi_1 \times \left(p_B - x_i(t)\right) + \varphi_2 \times \left(g_B - x_i(t)\right) \tag{3.8}$$

Na qual p_B refere-se à posição da partícula que levou ao seu melhor desempenho até o momento, g_B é o melhor desempenho global do sistema até o momento, t representa a iteração

corrente e φ_1 e φ_2 são constantes que representam, respectivamente, os componentes cognitivo e social do algoritmo, cujos valores devem ser definidos pelo operador e influenciam na velocidade de convergência do algoritmo. No presente trabalho, adotou-se $\varphi_1 = \varphi_2 = 2$

Após o transcorrerem-se o número máximo t de iterações, todos os agentes de pesquisa tendem a se concentrar próximos ao ponto de ótimo global da função otimizada, conforme a Figura 3.3

Figura 3.3 – Convergência do *Particle Swarm Optimization* ao longo das iterações



Fonte: Baseado em Marçon, 2018

O pseudocódigo do algoritmo PSO é apresentado na Figura 3.4

Figura 3.4 – Pseudocódigo do *Particle Swarm Optimization* (PSO)

1. Determine o número de partículas P da população
2. Inicialize aleatoriamente a posição inicial x de cada partícula p de P
3. Atribua uma velocidade inicial v igual para todas as partículas
4. Para cada partícula p em P faça:
   (a) Avalie a função objetivo
   (b) Calcule a melhor posição p_B da partícula p até o momento
5. Descubra a partícula com a melhor avaliação de toda a população g_B
6. Para cada partícula p em P faça:
   (a) Atualize a velocidade da partícula de acordo com a equação 3.8
   (b) Atualize a posição da partícula conforme a equação 3.7
7. Se a condição de término não for alcançada, retorne ao passo 4

Fonte: Baseado em Serapião, 2009

3.3.2 *Harmony Search* (HS)

O *Harmony Search* é um algoritmo meta-heurístico de otimização proposto por Geem et al. (2001), cuja inspiração encontra-se na performance de um grupo de músicos que buscam a harmonia perfeita produzida por diferentes instrumentos tocados simultaneamente.

A harmonia musical, definida teoricamente como uma relação especial entre várias ondas sonoras com diferentes frequências naturais, pode ser considerada como uma combinação esteticamente agradável de sons distintos. As performances de grupos musicais procuram obter sempre a melhor combinação estética possível de sons, de forma análoga, os algoritmos de otimização realizam um processo de busca pelo melhor resultado determinado por uma função objetivo. Ressalta-se ainda que, os valores de cada uma das variáveis de projeto que influenciam na avaliação da função objetivo têm papel semelhante aos dos sons de cada um dos instrumentos que compõem a harmonia musical. (GEEM et al., 2001).

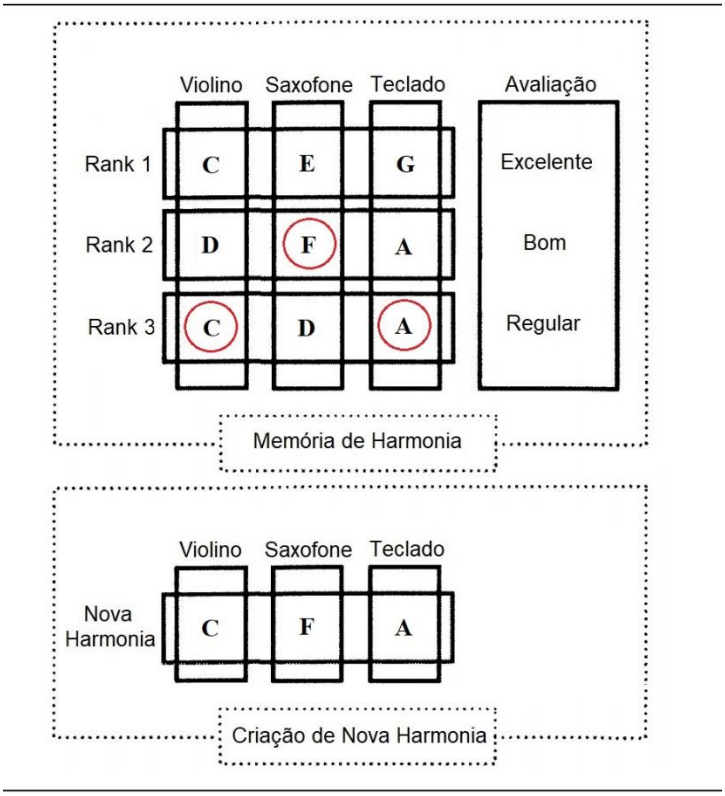
O algoritmo *Harmony Search* (HS) pode ser descrito, conforme apontaram Geem et al. (2001), através de um procedimento de quatro etapas que é apresentado a seguir.

a) Inicialize uma Memória de Harmonia HM (*Harmony Memory*);
b) Improvise uma nova harmonia na HM;
c) Se a nova Harmonia for menor que a harmonia mínima existente na HM, inclua a nova harmonia na HM e exclua a harmonia mínima da HM;
d) Retorne à etapa 2 enquanto os critérios de parada não forem totalmente atendidos.

Para exemplificar a estrutura, o funcionamento e a atualização da Memória de Harmonia ao longo da execução do algoritmo, Geem et al. (2001) propõem pensar no problema a ser resolvido como um trio de músicos composto por violino, saxofone e teclado, que buscam a melhor harmonia musical. Desta forma, inicialmente a Memória de Harmonia é preenchida por harmonias aleatórias, tais como: (C, E, G), (D, F, A) e (C, D, A) que são classificadas por estimativa estética, nas quais C representa a nota Dó, E representa a nota Mi, G representa a nota Sol, D representa a nota Ré, F representa a nota Fá e A representa a nota Lá. Na sequência, ocorre o processo de improvisação, onde os três instrumentos produzem uma nova harmonia a partir das notas já existentes na HM. No exemplo considerado, o violino toca a nota {C} entre {C, D, C}; o saxofone toca {F}, entre {E, F, D} e o teclado toca {A}, entre o conjunto de notas

{G, A, A}. Desta forma, a nova harmonia produzida pelos três instrumentos é (C, F, A). Este processo pode ser visualizado na Figura 3.5. Se a harmonia recém-criada for melhor que qualquer uma já existente na HM, a nova harmonia será incluída na HM e a pior harmonia será excluída.

Figura 3.5 – Exemplo de criação de uma nova harmonia a partir da HM



Fonte: Adaptado de Geem et al. (2001)

Até então, pressupõe-se que todas as partes da solução global do problema já estejam inseridas inicialmente na HM. Como isso não acontece na maior parte dos casos, o algoritmo se utiliza de um parâmetro denominado HCMR (*Harmony Memory Considering Rate*) que é fixado num valor entre 0 e 1. Em seguida, é gerada uma variável aleatória, também entre 0 e 1 que será comparada com o valor do HCMR. Se o valor da variável aleatória for maior que o valor do HCMR, o algoritmo procura novas notas de forma aleatória, dentro de um intervalo possível, mas sem considerar a HM. (GEEM et al., 2001)

Se o parâmetro HCMR possuir um valor extremamente baixo (próximo de 0), poucas harmonias próximas da harmonia ótima serão selecionadas, fazendo com que a convergência do algoritmo seja extremamente lenta. Entretanto, se este parâmetro for alto (próximo de 1), quase todas

novas harmonias serão criadas a partir de harmonias pré-existentes na HM deixando com que outras harmonias não sejam bem exploradas, levando a soluções equivocadas. (YANG, 2009b).

Visando a melhoria de desempenho do HS, Geem et al. (2001) introduziram um parâmetro PAR (*Pitch Adjusting Rate*) – uma taxa que varia de 0 a 1 – responsável por imitar o ajuste de afinação de cada instrumento do conjunto. Este parâmetro é inserido no algoritmo para mudar ligeiramente a solução para valores vizinhos, dentro de uma faixa de valores possíveis. Yang (2009b) aponta que uma taxa PAR muito baixa pode retardar a convergência do algoritmo, uma vez que há um limite de exploração a um subespaço muito pequeno do espaço total de pesquisa. Em contrapartida, taxas PAR muito elevadas podem ocasionar o espalhamento da solução em torno de vários ótimos possíveis, como ocorre em uma pesquisa totalmente aleatória.

Na Figura 3.6, é ilustrado o pseudocódigo do Algoritmo *Harmony Search*.

Figura 3.6 – Pseudocódigo do *Harmony Search* (HS)

Início
  Defina a função objetivo $f(\boldsymbol{x}), \boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$
  Defina o parâmetro HCMR
  Defina o parâmetro PAR
  Gere a Memória de Harmonia com harmonias aleatórias
  Enquanto ($t <$ número máximo de iterações)
    Enquanto ($i \leq$ número de variáveis)
      Se (rand < HCMR), *Escolha um valor da HM para a variável* i
        Se (rand < PAR), *ajuste o valor para um valor próximo*
        Fim se
      Senão *escolha um valor aleatório*
      Fim se
    Fim enquanto
    aceite a nova harmonia (solução) se ela for melhor
  Fim enquanto
  Encontre a melhor solução atual
Fim

Fonte: baseado em Yang, 2009b

3.3.3 *Firefly Algorithm* (FA)

O *Firefly Algorithm* (FA) é um algoritmo meta-heurístico de otimização proposto em 2009 por Yang e teve como inspiração as características luminescentes dos vaga-lumes. O algoritmo, portanto, aponta que a luz intermitente produzida pelos vagalumes como forma de atrair

parceiros de acasalamento e/ou potenciais presas esteja associada à função objetivo que será otimizada.

Yang (2009a) aponta que, para a implementação do algoritmo FA, devem ser seguidas basicamente as três regras apresentadas a seguir:

a) Um vaga-lume pode ser atraído por quaisquer outros vaga-lumes;

b) A atratividade é proporcional ao brilho. Desta forma, considerando-se dois vaga-lumes, o menos brilhante sempre se moverá em direção ao vaga-lume mais brilhante. Tanto a atratividade quanto o brilho diminuem à medida que a distância aumenta. Se não houver nenhum vaga-lume mais brilhante que um vaga-lume específico, este se moverá de forma aleatória;

c) O brilho ou a intensidade da luz de um vagalume é afetado ou determinado pela função objetivo a ser otimizada.

Para a implementação do FA, duas questões são consideradas de suma importância: A variação da intensidade da luz e a formulação da atratividade. Por conveniência, assume-se que a atratividade de um vaga-lume é determinada pela intensidade da sua luz e esta, por sua vez, está relacionada com a função objetivo. Entretanto, a atratividade será relativa, uma vez que será julgada pelos outros vaga-lumes. Desta forma, a atratividade irá variar de acordo com a distância entre os vaga-lumes. (YANG, 2010b).

Yang (2010b) aponta que, a atratividade β de um vaga-lume específico, julgada por outro vaga-lume a uma distância r_f, pode ser definida pela Equação 3.9

$$\beta = \beta_0 e^{-\gamma . {r_f}^2} \tag{3.9}$$

sendo que β_0 é a atratividade em $r_f = 0$ e γ é um coeficiente de absorção de luz.

De acordo com Yang (2010c), a distância entre quaisquer dois vagalumes i e j localizados nas posições x_i e x_j é a distância cartesiana apresentada na Equação 3.10

$$r_{f_{ij}} = \left|\left|x_i - x_j\right|\right| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{3.10}$$

sendo que $x_{i,k}$ é o k-ésimo componente da coordenada espacial x_i do i-ésimo vaga-lume.

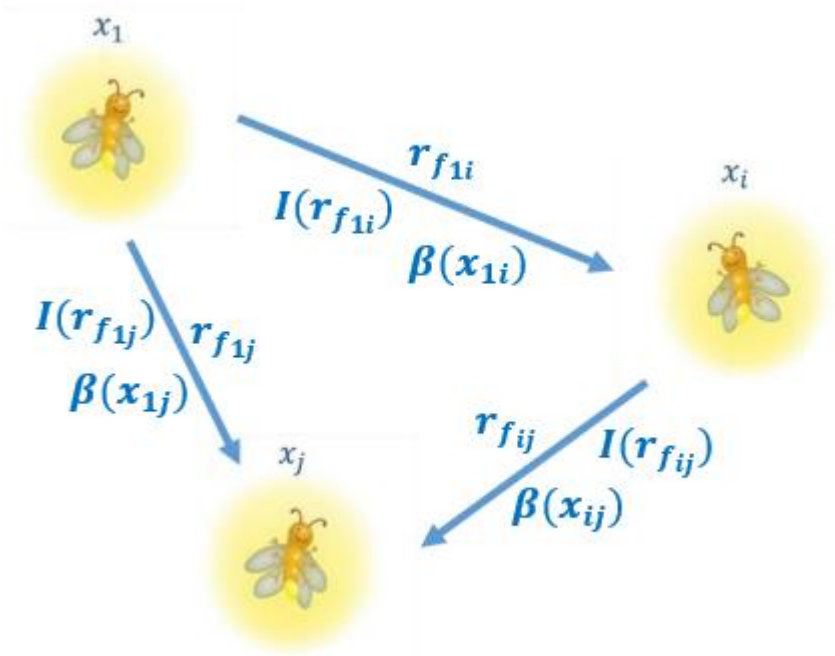
Finamente, Yang (2010b, 2010c) aponta que o movimento de um vaga-lume i, atraído por outro vaga-lume mais brilhante j, é regido pela Equação 3.11

$$x_{i+1} = x_i + \beta_0 e^{-\gamma r_{fij}{}^2}\left(x_j - x_i\right) + \alpha_r \epsilon_i \tag{3.11}$$

onde o 2º termo da direita se dá devido à atração, e o terceiro termo é a randomização, sendo α_r um parâmetro de randomização e ϵ_i um vetor de números aleatórios extraídos de uma distribuição gaussiana ou uniforme. Geralmente, para a implementação do algoritmo, utiliza-se $\beta_0 = 1;\ \alpha_r \in [0,1]$ e $\epsilon_i = rand - 1/2$, onde $rand$ é um gerador de número aleatórios distribuído uniformemente entre 0 e 1 (YANG, 2010b, 2010c).

Na Figura 3.7 ilustra-se o comportamento dos agentes de pesquisa no *Firefly Algorithm*

Figura 3.7 – Relações do *Firefly Algorithm* incluindo localizações x, distância r_f, brilho $I(r_f)$ e atratividade $\beta(x)$



Fonte: Baseado em Louzazni et al., 2018

O parâmetro γ, que representa a variação da atratividade, é crucial para determinar a velocidade de convergência do algoritmo. Teoricamente, γ pode assumir qualquer valor entre 0 e ∞, todavia, ao se definir $\gamma \to 0$, o comportamento do *Firefly Algorithm* corresponderá a uma versão acelerada do algoritmo PSO. No extremo oposto, ao utilizar $\gamma \to \infty$, o desempenho do FA será reduzido a uma simples pesquisa aleatória. Desta forma, a possibilidade de ajustar o

parâmetro γ entre os dois extremos, torna o algoritmo capaz de superar tanto o PSO como as pesquisas aleatórias. (YANG, 2010b, 2010c).

Na Figura 3.8 é apresentado o pseudocódigo do *Firefly Algorithm*.

Figura 3.8 – Pseudocódigo do *Firefly Algorithm* (FA)

Início
Defina a função objetivo $f(\boldsymbol{x}), \boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$
Gere a população inicial de vaga-lumes $\boldsymbol{x_i}$ $(i = 1,2, \ldots n)$
A intensidade da luz $\boldsymbol{I_i}$ *em* $\boldsymbol{x_i}$ *é determinada por* $f(\boldsymbol{x_i})$
Defina o coeficiente de absorção de luz γ
Enquanto ($t <$ número máximo de iterações)
Para $i = 1 : n$ *para todos* n *vaga-lumes*
Para $j = 1 : i$ *para* todos n vaga-lumes (loop interno)
Se $(I_i < I_j)$, *mova o vaga-lume* i *em direção a* j *na dimensão* d; **Fim se**
Varie a atratividade com a distância r_f *via* $exp[-\gamma r_f]$
Avalie novas soluções e atualize a intensidade da luz
Fim para j
Fim para i
Classifique os vaga-lumes e encontre o melhor global atual
Fim Enquanto
Resultados e visualização pós-processo
Fim

Fonte: Baseado em Yang, 2009a

3.3.4 *Search Group Algorithm* (SGA)

O *Search Group Algorithm* (SGA) é um algoritmo meta-heurístico de otimização desenvolvido por Gonçalves et al. (2015), que visa ter um bom equilíbrio entre a pesquisa local e global dentro do domínio de projeto.

O algoritmo é composto por cinco etapas que são apresentadas detalhadamente a seguir, de acordo com a formulação de Gonçalves et al. (2015).

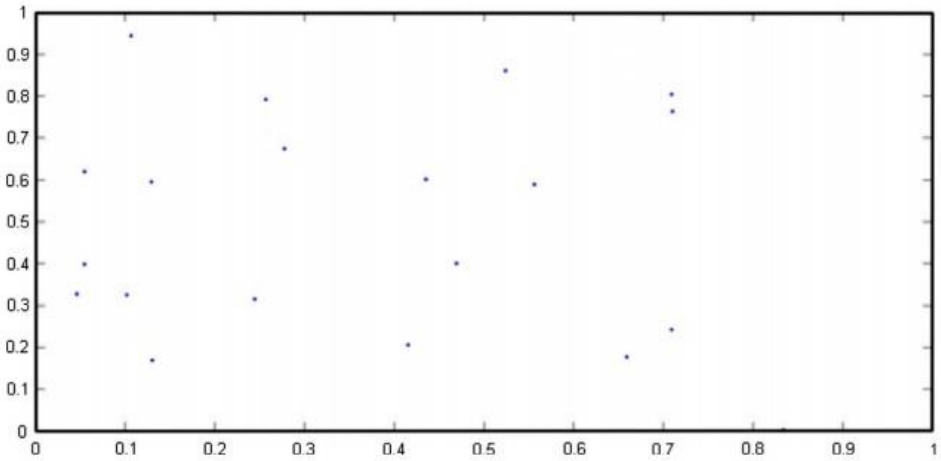
Na primeira etapa, a população inicial $\boldsymbol{P}$ é gerada aleatoriamente no domínio de pesquisa de acordo com a Equação 3.12

$$P_{ij} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) U[0,1] \tag{3.12}$$

onde i varia de 1 até o número de indivíduos n_{Pop} da população; j varia de 1 até o número n de variáveis de projeto; P_{ij} é a j-ésima variável de projeto do i-ésimo indivíduo da população; $U[0,1]$ é uma variável uniforme e aleatória que varia entre 0 e 1; x_j^{min} e x_j^{max} são,

respectivamente, os limites inferior e superior da j-ésima variável de projeto. Na Figura 3.9 é apresentada uma população gerada aleatoriamente num domínio bidimensional, onde cada ponto representa um indivíduo da população.

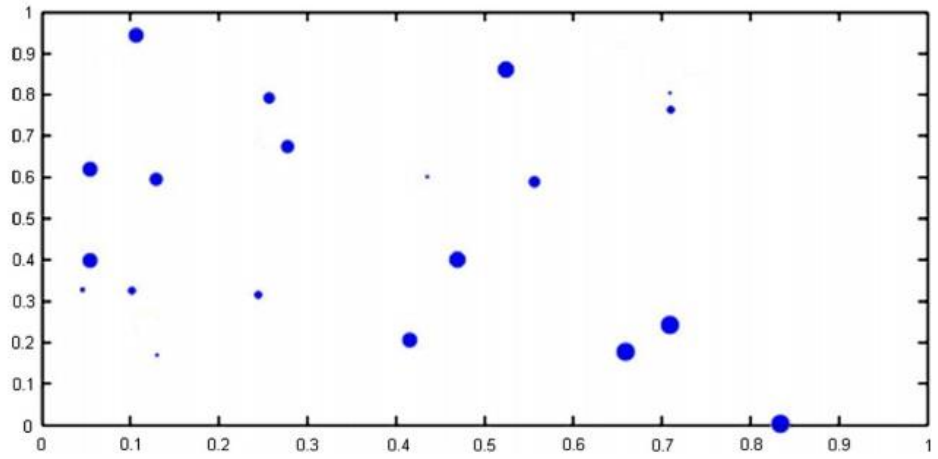
Figura 3.9 – População inicial gerada aleatoriamente em um domínio bidimensional



Fonte: Gonçalves et al., 2015

Na segunda etapa, ocorre a avaliação da função objetivo de cada indivíduo. Os indivíduos são dimensionados de forma que um maior tamanho será atribuído àqueles que geraram os melhores valores da função objetivo, conforme ilustrado na Figura 3.10.

Figura 3.10 – Avaliação da função objetivo de cada indivíduo

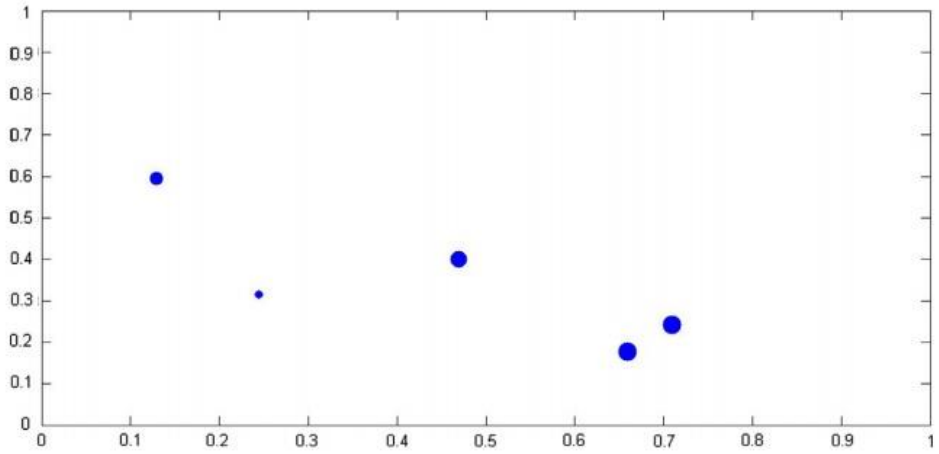


Fonte: Gonçalves et al., 2015

Após a avaliação das funções objetivo, o grupo de pesquisa $\boldsymbol{R}$ é criado a partir da seleção de torneio-padrão de n_g indivíduos de $\boldsymbol{P}$. Na Figura 3.11 mostra-se um exemplo dessa seleção.

Em cada iteração, os membros R_i do grupo de pesquisa são classificados de modo que R_1 representa o melhor *design* e R_{ng} o pior *design* entre todos os membros do grupo.

Figura 3.11 – Grupo de pesquisa inicial composto por cinco indivíduos da população inicial



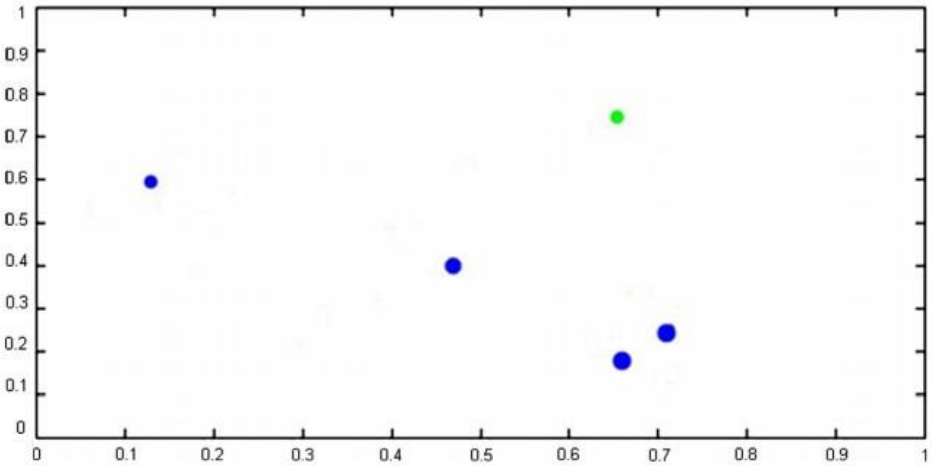
Fonte: Gonçalves et al, 2015

Na terceira etapa, uma estratégia de mutação é realizada a cada iteração do algoritmo, visando aumentar a capacidade de busca global do algoritmo. Essa estratégia consiste em substituir indivíduos n_{mut} de R por novos indivíduos gerados com base nas estatísticas do atual grupo de pesquisa. Um novo indivíduo é, portanto, gerado de acordo com a Equação 3.13

$$x_j^{mut} = E\left[\boldsymbol{R}_{:j}\right] + t_d \varepsilon \sigma[\boldsymbol{R}_{:j}] \tag{3.13}$$

onde x_j^{mut} é a j-ésima variável de um determinado indivíduo mutado; E e σ são os operadores de valor médio e desvio padrão; ε é uma variável aleatória conveniente; t_d é uma variável que controla a distância na qual o novo indivíduo é gerado e $R_{:j}$ é a j-ésima coluna da matriz do grupo de pesquisa. É importante ressaltar que a probabilidade de um indivíduo ser substituído depende de sua classificação no grupo de pesquisa atual, isto é, quanto pior o *design* do indivíduo, maior probabilidade ele tem de ser substituído. Para isso, uma seleção de torneio-inverso é utilizada. Na Figura 3.12 é apresentado o mesmo grupo de pesquisa da Figura 3.11 após a mutação, onde, o indivíduo mutado (círculo verde) substituiu um membro do grupo de pesquisa.

Figura 3.12 – Mutação do grupo de pesquisa



Fonte: Gonçalves et al., 2015

Na quarta etapa, ocorre a Geração das Famílias de cada Membro do Grupo de Pesquisa. Uma família F_i é o conjunto composto pelo membro do grupo de pesquisa e os indivíduos que ele gerou. Desta forma, cada membro do grupo de pesquisa gera uma família através da perturbação apresentada na Equação 3.14

$$x_j^{new} = R_{ij} + \alpha\varepsilon \tag{3.14}$$

onde α controla o tamanho da perturbação.

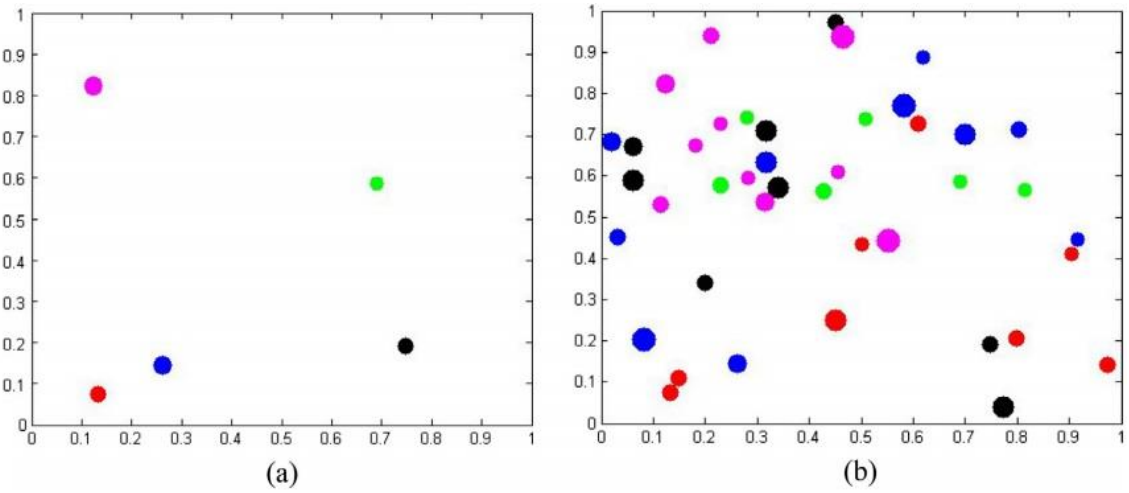
A cada iteração k, o parâmetro α é reduzido de acordo com a Equação 3.15

$$\alpha^{k+1} = b_s \alpha^k \tag{3.15}$$

onde b_s é um parâmetro próprio do algoritmo.

Assim, nas primeiras iterações do SGA, o valor de α^k é definido de forma que um indivíduo gerado por um membro do grupo de pesquisa surja em qualquer ponto do domínio de projeto, não estando necessariamente no entorno da vizinhança do membro que o gerou. Este processo é ilustrado na Figura 3.13, onde (a) mostra cinco membros do grupo de pesquisa e (b) ilustra os indivíduos que cada membro gerou espalhados pelo domínio de projeto.

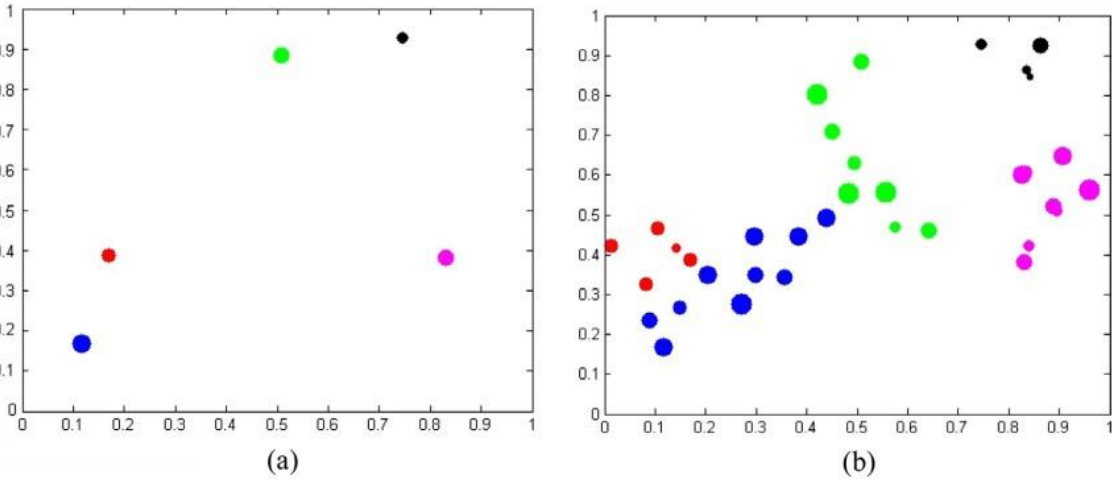
Figura 3.13 – Geração das famílias dos membros do grupo de pesquisa nas primeiras iterações do SGA



Fonte: Gonçalves et al., 2015

À medida que as iterações avançam, o tamanho da perturbação α^k diminui, conforme mostrado na Equação 3.15. Esse efeito faz com que os novos indivíduos surjam na vizinhança do membro que os gerou, conforme é apresentado na Figura 3.14, onde (a) mostra cinco membros do grupo de pesquisa e (b) ilustra as famílias que cada um desses membros gerou. É importante ressaltar que é usual atribuir um valor mínimo a α^k, de forma que se $\alpha^k < \alpha_{mín}$, então utiliza-se $\alpha^k = \alpha_{mín}$.

Figura 3.14 – Geração das famílias dos membros do grupo de pesquisa nas últimas iterações do SGA



Fonte: Gonçalves et al., 2015

Um dos recursos mais relevantes do SGA é que, quanto melhor um membro do grupo de pesquisa é classificado, mais indivíduos ele gera, isto é, o número de indivíduos gerados por um determinado membro depende da qualidade da sua função objetivo.

Por fim, a última etapa consiste na seleção de um novo grupo de pesquisa. Na fase global do SGA, que ocorre nas primeiras it_{global}^{max} iterações, o objetivo do algoritmo é explorar a maior parte do espaço de projeto, logo um novo grupo de pesquisa é formado pelo melhor membro de cada família. Quando o número da iteração atual ultrapassa o valor it_{global}^{max}, o processo de seleção é modificado de modo a um novo grupo de pesquisa ser formado pelos melhores n_g indivíduos entre todas as famílias.

Para melhor exemplificar o funcionamento do SGA, na Figura 3.15 é apresentado o pseudocódigo do algoritmo.

Figura 3.15 – Pseudocódigo do *Search Group Optimization* (SGA)

1. Inicialize os parâmetros do algoritmo
2. Gere a população inicial $\boldsymbol{P}$
3. Crie o grupo de pesquisa inicial $\boldsymbol{R}^k$ selecionando n_g indivíduos da população inicial usando uma seleção de torneio
4. Substitua n_{mut} indivíduos por novos membros criados como descrito na Equação 3.13
5. Gere as famílias $\boldsymbol{F}_i$ usando a Equação 3.14
6. Selecione o novo grupo de pesquisa de acordo com as regras:
   (a) Se $k < it_{global}^{max}$: O grupo de pesquisa $\boldsymbol{R}^{k+1}$ é formado pelo melhor membro de cada família
   (b) Se $k \geq it_{global}^{max}$: O grupo de pesquisa $\boldsymbol{R}^{k+1}$ é formado pelos melhores n_g indivíduos da população
7. Atualize α^{k+1} de acordo com a Equação 3.15
8. Faça $k = k + 1$, se $k < it^{max}$ volte ao passo 4
9. Solução obtida: $x^* = \boldsymbol{R}_{1,:}$

Fonte: Baseado em Gonçalves et al., 2015

3.3.5 *Whale Optimization Algorithm* (WOA)

O *Whale Optimization Algorithm* (WOA) é um algoritmo meta-heurístico de otimização desenvolvido por Mirjalili e Lewis (2016), inspirado no processo de caça das baleias jubarte e basicamente dividido em três fases distintas, cuja ordem ocorrência é definida aleatoriamente.

Na fase de cercar presas o algoritmo assume que a presa é, ou está próxima, da melhor solução atual, e o agente que obteve essa melhor solução será definido como melhor agente de pesquisa. Os demais agentes atualizarão, cada um, a sua posição em relação ao melhor agente de pesquisa, num comportamento representado pelas Equações 3.16 e 3.17

$$\boldsymbol{D} = |\boldsymbol{C_C}\, \boldsymbol{X}^*(t) - \boldsymbol{X}(t)| \tag{3.16}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}^*(t) - \boldsymbol{A}\boldsymbol{D} \tag{3.17}$$

onde t representa a iteração atual, $\boldsymbol{X}$ é o vetor de posição do agente considerado, $\boldsymbol{X}^*$ é o vetor de posição da melhor solução obtida até então e os vetores $\boldsymbol{A}$ e $\boldsymbol{C_C}$ são vetores de coeficientes definidos através das Equações 3.18 e 3.19

$$\boldsymbol{A} = 2\boldsymbol{a}\boldsymbol{r_a} - \boldsymbol{a} \tag{3.18}$$

$$\boldsymbol{C_C} = 2\boldsymbol{r_a} \tag{3.19}$$

sendo que $\boldsymbol{a}$ é um vetor de valores que será reduzido linearmente de 2 para zero ao longo das iterações e $\boldsymbol{r_a}$ é um vetor de valores aleatórios entre zero e 1.

Na busca local, denominada como método de ataque com rede de bolhas, o WOA pode assumir dois comportamentos. O primeiro, chamado de Mecanismo de Encolhimento Reduzido, diminui o valor de $\boldsymbol{a}$ na Equação 3.18, de forma que, ao longo desse processo, $\boldsymbol{A}$ se portará como um vetor de valores aleatórios no intervalo $[-\boldsymbol{a}, \boldsymbol{a}]$. Já o segundo comportamento que pode ser assumido, imita o movimento em espiral das baleias em torno da presa, através da Equação 3.20.

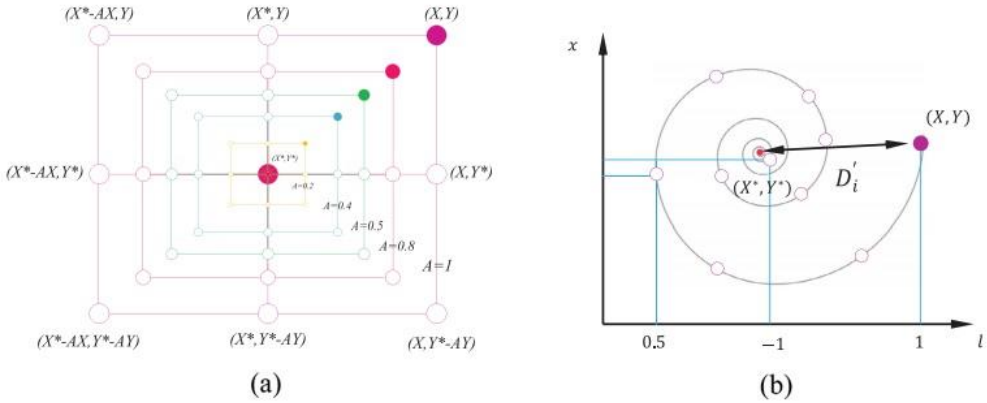
$$\boldsymbol{X}(t+1) = \boldsymbol{D'} e^{bl} \cos(2\pi l) + \boldsymbol{X}^*(t) \tag{3.20}$$

onde b é uma constante que garante o formato de espiral logarítmica e l é um valor gerado aleatoriamente no intervalo $[-1, 1]$. O termo $\boldsymbol{D'}$ é o vetor que indica a distância entre a i-ésima baleia e a presa, e seu valor é dado pela Equação 3.21

$$\boldsymbol{D'} = |\boldsymbol{X}^*(t) - \boldsymbol{X}(t)| \tag{3.21}$$

Na Figura 3.16 são apresentados ambos os comportamentos que podem ser assumidos na fase de busca através do método de ataque com rede de bolhas, sendo (a) o mecanismo de encolhimento reduzido e (b) a atualização de posição em movimento espiral.

Figura 3.16 – Mecanismo de pesquisa em rede de bolhas do WOA



Fonte: Mirjalili e Lewis, 2016

Por fim, na fase de busca global, deve ser adotada uma estratégia que irá forçar o agente de pesquisa a se afastar de uma baleia de referência. Portanto, deve-se utilizar o vetor **A** com valores aleatórios, geralmente maiores que 1 ou menores que -1, para atualizar a posição de um agente de busca de acordo com um agente escolhido aleatoriamente, em vez de escolher o melhor agente encontrado até então. Este procedimento é descrido pelas Equações 3.22 e 3.23

$$\boldsymbol{D} = |\boldsymbol{C_C X_{rand}} - \boldsymbol{X}| \tag{3.22}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X_{rand}} - \boldsymbol{AD} \tag{3.23}$$

onde $\boldsymbol{X_{rand}}$ é um vetor de posição aleatória (uma baleia aleatória) escolhido da população atual.

Na Figura 3.17 é apresentado o pseudocódigo do WOA.

Figura 3.17 – Pseudocódigo do *Whale Optimization Algorithm* (WOA)

Início
  Inicialize a população de baleias $X_i (i = 1,2,\dots,n)$
  Calcule a posição de cada agente de pesquisa
  $X^* =$ *Melhor agente de pesquisa*
  Enquanto ($t <$ número máximo de iterações)
    Para *cada agente de pesquisa*
    Atualize a, A, C, I e p
      Se 1 ($p < 0{,}5$)
        Se 2 ($|A| < 1$)
          Atualize a posição do agente de pesquisa atual pela Eq. 3.16
        Ou Se 2 ($|A| \geq 1$)
          Selecione um agente de pesquisa aleatório
          Atualize a posição do agente de pesquisa atual pela 3.23
        Fim Se 2
      Ou Se 1 ($p \geq 0{,}5$)
          Atualize a posição do agente de pesquisa atual pela 3.20
      Fim Se 1
    Fim Para
    Verifique se algum agente ultrapassa o espaço de pesquisa e altere-o
    Calcule a posição de cada agente de pesquisa
    Atualize X^* *se houver uma solução melhor*
    $t = t + 1$
  Fim Enquanto
  retorne X^*
Fim

Fonte: Baseado em Mirjalili e Lewis, 2016

3.4 FORMULAÇÃO DO TESTE DOS POSTOS SINALIZADOS DE WILCOXON

Grande parte dos algoritmos meta-heurísticos de otimização têm o seu desempenho avaliado através de análises estatísticas simples, comparando o seu desempenho ao desempenho de outros algoritmos meta-heurísticos. Estas análises, geralmente levam em consideração apenas os valores de média e desvio padrão entre os resultados obtidos após sucessivas execuções de um algoritmo para um mesmo problema.

Todavia, análises estatísticas avançadas podem revelar com maior precisão a eficiência de um algoritmo levando-se em consideração o resultado individual de cada simulação. De acordo com Derrac et al. (2011), o Teste dos Postos Sinalizados de Wilcoxon (*Wilkoxon Signed-Rank Test*), é um procedimento não-paramétrico que pode ser empregado em situação de testes do comportamento de duas amostras de resultados. Desta forma, o Teste de Wilcoxon, tem grande relevância na comparação-por-pares entre a eficiência de dois algoritmos meta-heurísticos.

A seguir é apresentada a formulação do Teste dos Postos Sinalizados de Wilcoxon, para a aplicação na comparação de algoritmos, conforme apontado por Derrac et al. (2011).

Durante n execuções de dois algoritmos para a solução de um mesmo problema de otimização, considera-se d_i a diferença numérica entre os resultados obtidos a cada execução. De modo que i varia de 1 até n. As diferenças d_i são classificadas de acordo com os seus valores absolutos, de modo que a menor diferença recebe a classificação 1, ou seja, $rank(d_i) = 1$, enquanto a maior diferença recebe a classificação n, isto é, $rank(d_i) = n$. Em caso de empate entre duas diferenças, ambas receberão o valor médio entre as duas classificações em questão (por exemplo, se duas diferenças estão empatadas nas classificações 2 e 3, ambas receberão classificação 2,5).

Em seguida, as classificações serão agrupadas em dois somatórios: T^+, que corresponde à soma das classificações onde o resultado do primeiro algoritmo supera o resultado do segundo algoritmo e T^-, que representa a soma das classificações nas quais o resultado obtido pelo segundo algoritmo supera o resultado do primeiro. Para o caso de problemas de minimização, onde terá melhor desempenho o algoritmo que apresenta o **menor** resultado, este processo é exposto nas Equações 3.24 e 3.25

$$T^+ = \sum_{d_i<0} rank(d_i) \tag{3.24}$$

$$T^- = \sum_{d_i>0} rank(d_i) \tag{3.25}$$

Em problemas de maximização, inverte-se o sinal dos índices dos somatórios, neste caso, o processo pode ser representado pelas Equações 3.26 e 3.27

$$T^+ = \sum_{d_i>0} rank(d_i) \tag{3.26}$$

$$T^- = \sum_{d_i>0} rank(d_i) \tag{3.27}$$

Sempre que, em uma determinada simulação, a diferença entre os resultados obtidos por cada algoritmo em análise for nula, ou seja, ambos algoritmos geraram o mesmo resultado, atribui-se classificação zero, ou seja, sempre que $d_i = 0$ tem-se que $rank(d_i) = 0$.

Uma vez calculados os valores de T^+ e T^-, poderá se dizer que, entre os dois algoritmos em análise, o primeiro algoritmo apresentará melhor desempenho sobre o segundo algoritmo, sempre que $T^+ > T^-$.

Para o caso específico onde as diferenças de todas as simulações sejam nulas, não será possível determinar qual dos dois algoritmos analisados é mais eficiente, tendo-se, neste caso, $T^+ = T^- = 0$.

Neste trabalho, o Teste dos Postos Sinalizados de Wilcoxon foi utilizado para avaliar o desempenho do novo algoritmo meta-heurístico formulado. Os resultados da avaliação completa, utilizando o Teste de Wilcoxon, podem ser observados no Capítulo 8.

4 *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM* (CIOA) – UM NOVO ALGORITMO META-HEURÍSTICO DE OTIMIZAÇÃO

Neste capítulo apresenta-se a formulação detalhada do ***Circle-Inspired Optimization Algorithm* (CIOA)**, ou seja, o novo algoritmo de otimização desenvolvido no presente trabalho.

Conforme pode ser observado nos capítulos anteriores, a maior parte dos algoritmos meta-heurísticos já desenvolvidos, apresenta inspiração em fenômenos da natureza como, por exemplo, o comportamento de certas espécies de animais. Todavia, para incorporar estes comportamentos ao algoritmo, são necessários: O conhecimento prévio do fenômeno natural que inspirou o algoritmo, a possibilidade deste fenômeno ser descrito através de equações matemáticas adequadas para a implementação computacional.

A maior parte dos algoritmos apresentados no Capítulo 3, possuem agentes de pesquisa que descrevem diferentes tipos de movimentos, muitas vezes com formulação complexa. Durante o desenvolvimento do CIOA, buscou-se garantir que os agentes de pesquisa fossem regidos por movimentos oriundos de equações matemáticas amplamente conhecidas, tornando a implementação do algoritmo de fácil entendimento, ou seja, sem a necessidade de conhecimentos prévios de outras áreas de estudo, como por exemplo, a biologia. Em outras palavras, na elaboração do CIOA procurou-se inspiração na própria matemática, em vez de utilizá-la como uma ferramenta para descrever fenômenos específicos da natureza ou de outras áreas de estudo. Deste modo, surge a ideia de utilizar, como inspiração para a formulação do CIOA, a equação de um círculo, por ser uma das formas geométricas mais conhecidas. Assim, o algoritmo proposto trata-se de uma ferramenta de otimização aplicável à engenharia estrutural (conforme será observado nos próximos capítulos) cuja formulação exige conhecimento prévio apenas em áreas totalmente relacionadas à engenharia, tais quais, matemática, trigonometria e computação.

4.1 FORMULAÇÕES DA CIRCUNFERÊNCIA TRIGONOMÉTRICA

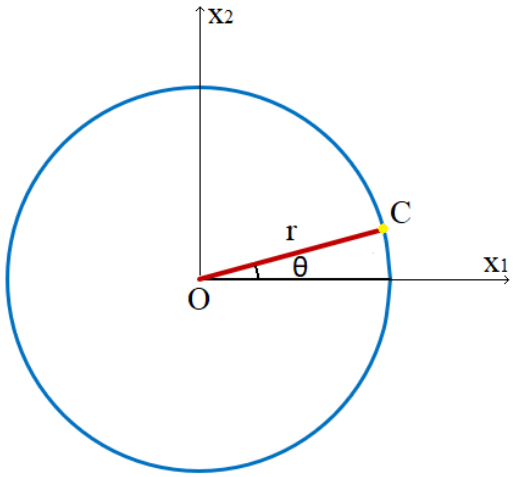
Dada uma circunferência de centro O e raio r, contida num plano cartesiano X_1X_2, conforme a Figura 4.1, sabe-se que qualquer ponto C, sobre esta circunferência, pode ter suas coordenadas C_{X1} e C_{X2} calculadas através da Equações 4.1 e 4.2:

$$C_{X1} = O_{X1} + r\cos\theta \tag{4.1}$$

$$C_{X2} = O_{X2} + r\,\mathrm{sen}\,\theta \tag{4.2}$$

sendo O_{X1} e O_{X2} as coordenadas do centro da circunferência e θ é o ângulo formado entre o eixo horizontal X_1 e o raio que liga O até C.

Figura 4.1 – Circunferência trigonométrica



Da mesma forma, o caminho reverso pode ser realizado, calculando-se as coordenadas do centro da circunferência a partir de qualquer ponto C localizado sobre o círculo. Para isso, uma pequena manipulação nas Equações 4.1 e 4.2 se faz necessária, de modo a obter-se as equações 4.3 e 4.4:

$$O_{X1} = C_{X1} - r\cos\theta \tag{4.3}$$

$$O_{X2} = C_{X2} - r\,\mathrm{sen}\,\theta \tag{4.4}$$

Utilizando-se basicamente apenas estas quatro equações, aliadas a conhecimentos da área de otimização meta-heurística, como, por exemplo, a necessidade de equilibrar intensificação (pesquisa local) e diversificação (pesquisa global) através de randomização, pode-se formular um novo algoritmo meta-heurístico de otimização onde os agentes de pesquisa descrevem trajetórias similares a arcos de circunferência, regidas por raios r que podem variar ao longo das iterações, separados entre si por um ângulo θ constante.

Por ter a circunferência como fonte principal de inspiração, o algoritmo criado neste trabalho foi denominado ***Circle-Inspired Optimization Algorithm*** **(CIOA)**, ou, em tradução literal para o português: Algoritmo de Otimização Inspirado em Círculo.

4.2 INICIALIZAÇÃO DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM*

No *Circle-Inspired Optimization Algorithm*, cada agente de pesquisa se move ao longo de arcos regidos por dois parâmetros principais: Um ângulo θ, definido pelo próprio usuário e um raio r, calculado pelo algoritmo e cujo valor depende da avaliação da função objetivo: quanto melhor for a avaliação da função objetivo feita por um determinado agente de pesquisa, menor será o valor de r para este agente na iteração seguinte.

Sejam N_{ag} o número de agentes de pesquisa utilizados, L_b e U_b os limites inferior e superior, respectivamente, de cada variável de projeto e n o número total de variáveis de projeto, o algoritmo é inicializado gerando um vetor $\boldsymbol{r}$, composto de N_{ag} elementos onde o valor r_j de cada elemento é dado pela Equação 4.5

$$r_j = \frac{c_r . j^2}{N_{ag}} , \qquad 1 \leq j \leq N_{ag} \tag{4.5}$$

sendo c_r uma constante determinada através da Equação 4.6.

$$c_r = \frac{\sqrt{U_b - L_b}}{N_{ag}} \tag{4.6}$$

Deste modo, verifica-se na Equação 4.5 que os elementos do vetor de raios estão ordenados de forma crescente, sendo que o primeiro elemento, r_1, assumirá o valor $r_1 = c_r / N_{ag}$ enquanto o último elemento, $r_{N_{ag}}$, assumirá o valor $r_{N_{ag}} = c_r \, . N_{ag}$.

É importante ressaltar que, como em qualquer outro algoritmo meta-heurístico, o número de agentes de pesquisa do CIOA deve ser definido pelo operador. Na seção 5.2 são apresentados alguns testes de sensibilidade que revelam como o número de agentes influencia no comportamento do algoritmo.

Uma vez realizada a inicialização, a primeira solução gerada pelo *Circle-Inspired Optimization Algorithm* é realizada de forma aleatória. Nesta solução, uma variável x_i pode assumir qualquer valor possível entre L_b e U_b, conforme a equação 4.7.

$$x_i = L_b + (U_b - L_b) . rand \tag{4.7}$$

Na qual $rand$ é um número aleatório, extraído de uma distribuição uniforme entre zero e um.

Após cada agente de pesquisa atribuir valores aleatórios para as variáveis de projeto na primeira solução, uma avaliação da função objetivo é realizada, e então cada agente de pesquisa é classificado em um ranking de acordo com a qualidade da solução que obteve. É importante ressaltar que o ponto de partida do CIOA, ou seja, a primeira solução gerada, não interfere de forma significativa no comportamento do algoritmo durante as iterações seguintes.

4.3 LOOP PRINCIPAL DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM*

Ao longo das iterações do CIOA, cada agente de pesquisa terá suas coordenadas na próxima iteração definidas pela classificação realizada na iteração anterior, de modo que os agentes mais bem classificados, ou seja, aqueles que obtiveram as melhores soluções, façam movimentos mais curtos (utilizando raios menores do vetor $\boldsymbol{r}$) enquanto os agentes que ocupam as piores classificações façam longos movimentos.

Um agente de pesquisa, classificado com a $j^{ésima}$ melhor solução em uma determinada iteração k, terá suas novas coordenadas, na iteração $k+1$ calculadas através das equações 4.8 e 4.9.

$$x_{2i}\,(k+1) = x_{2i}\,(k) - rand_1\,.\,r_j\,.\,sen(k\,.\,\theta) + rand_2\,.\,r_j\,.\,sen\big((k+1)\,.\,\theta\big) \qquad (4.8)$$

$$x_{2i-1}\,(k+1) = x_{2i-1}\,(k) - rand_3\,.\,r_j\,.\,cos(k\,.\,\theta) + rand_4\,.\,r_j\,.\,cos\big((k+1)\,.\,\theta\big) \qquad (4.9)$$

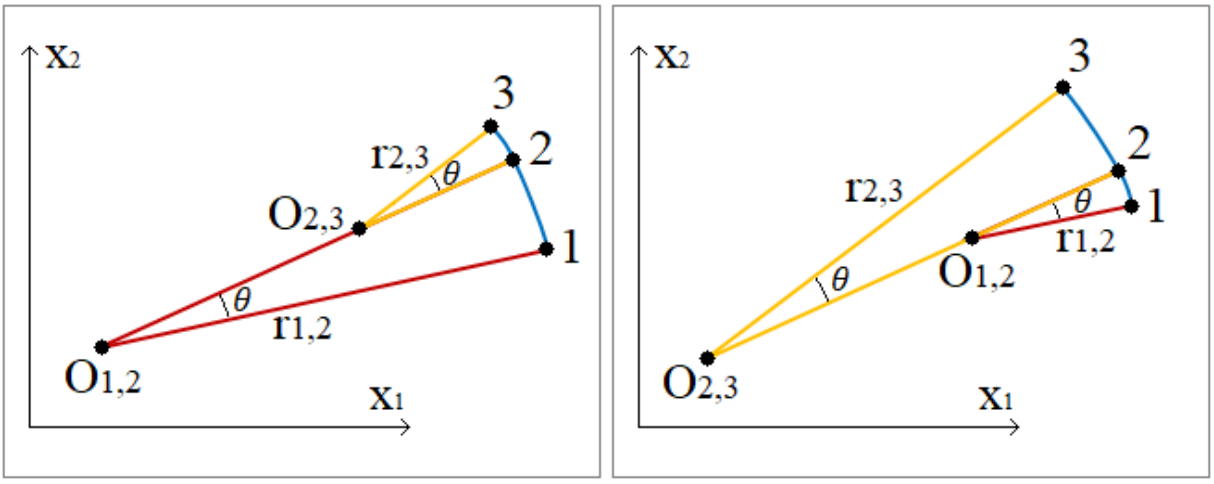
Nas quais os índices $2i$ e $2i-1$, das coordenadas, referem-se respectivamente a números pares e ímpares. Logo a Equação 4.8 atualiza, a cada iteração, as coordenadas (ou variáveis de projeto) $\boldsymbol{x} = [x_2;\ x_4;\ x_6;\ ...]$ enquanto a Equação 4.9 é responsável por atualizar as coordenadas (ou variáveis de projeto) $\boldsymbol{x} = [x_1;\ x_3;\ x_5;\ ...]$; as variáveis $rand$ referem-se a números aleatórios extraídos de distribuição uniforme entre zero e um; o ângulo θ é um dos parâmetros fornecidos pelo usuário do algoritmo; a variável r_j corresponde ao $j^{ésimo}$ elemento do vetor $\boldsymbol{r}$, ou seja, o $j^{ésimo}$ menor raio. A utilização de duas equações ocorre por dois motivos principais: o primeiro deles é para manter a premissa da inspiração no círculo, onde cada par de variáveis gera um plano trigonométrico; o segundo motivo é para garantir maior diversidade ao algoritmo.

Desta forma, um agente que obtém a melhor solução para a função objetivo em uma iteração k, terá suas coordenadas atualizadas utilizando-se o menor raio do vetor $\boldsymbol{r}$, ou seja, r_1. No outro extremo, o agente que obtém a pior solução em uma iteração k, terá suas coordenadas atualizadas através do maior raio de $\boldsymbol{r}$, isto é, $r_{N_{ag}}$.

As Equações 4.8 e 4.9 podem ser consideradas como equações de três termos à direita do sinal da igualdade. O primeiro termo refere-se à coordenada da iteração corrente; o segundo termo, inspirado nas Equações 4.3 e 4.4, é responsável pela atualização do centro do círculo enquanto o terceiro termo promove a busca pela nova coordenada da iteração seguinte. Pode-se dizer, portanto que, apenas com as Equações 4.8 e 7.9, o algoritmo é capaz de realizar a busca de novas coordenadas para cada agente de pesquisa levando em consideração: O novo raio de circunferência, para cada agente de pesquisa, definido através da classificação das soluções obtidas por cada agente na última iteração; a atualização do centro da circunferência através da redução ou do aumento do raio obtido em relação à iteração anterior.

O processo de alteração de raios e da atualização do centro do círculo é esquematizado a seguir: Supõe-se que, numa determinada iteração, um agente de pesquisa parte do ponto 1 para o ponto 2, num movimento regido pelo ângulo θ e pelo raio $r_{1,2}$ com centro em $O_{1,2}$. Na iteração seguinte, o agente deixa o ponto 2 e segue para o ponto 3, desta vez, mantido o ângulo θ, o movimento é regido por um raio $r_{2,3}$ e possui centro em $O_{2,3}$. Esquemas ilustram esse processo na Figura 4.2 em dois casos distintos: O primeiro, em (a), no qual o agente melhora a sua classificação, reduzindo o tamanho do raio, ou seja, $r_{2,3} < r_{1,2}$; o segundo, ilustrado em (b), onde o agente de pesquisa piora a sua classificação, aumentando-se o tamanho do raio, isto é, $r_{2,3} > r_{1,2}$. É importante ressaltar que, para facilitar o entendimento do princípio do CIOA, a ação das variáveis aleatórias não foi considerada na Figura 4.2

Figura 4.2 – Alteração de raio e atualização do centro do círculo



(a) (b)

Ao longo de sucessivas iterações, é altamente provável que o comportamento de cada agente de pesquisa não siga nenhum padrão específico. Em outras palavras, pode-se se dizer que cada agente terá um comportamento que ora apresente melhorias de classificação no ranking das melhores soluções, ora apresente quedas de classificação. Entretanto, chama-se a atenção para três casos hipotéticos de comportamentos extremos onde:

a) **Caso 1:** Um agente de pesquisa sempre piora sua classificação no ranking, aumentando o raio que rege seu movimento ao longo das iterações, até obter a pior solução do grupo de agentes;

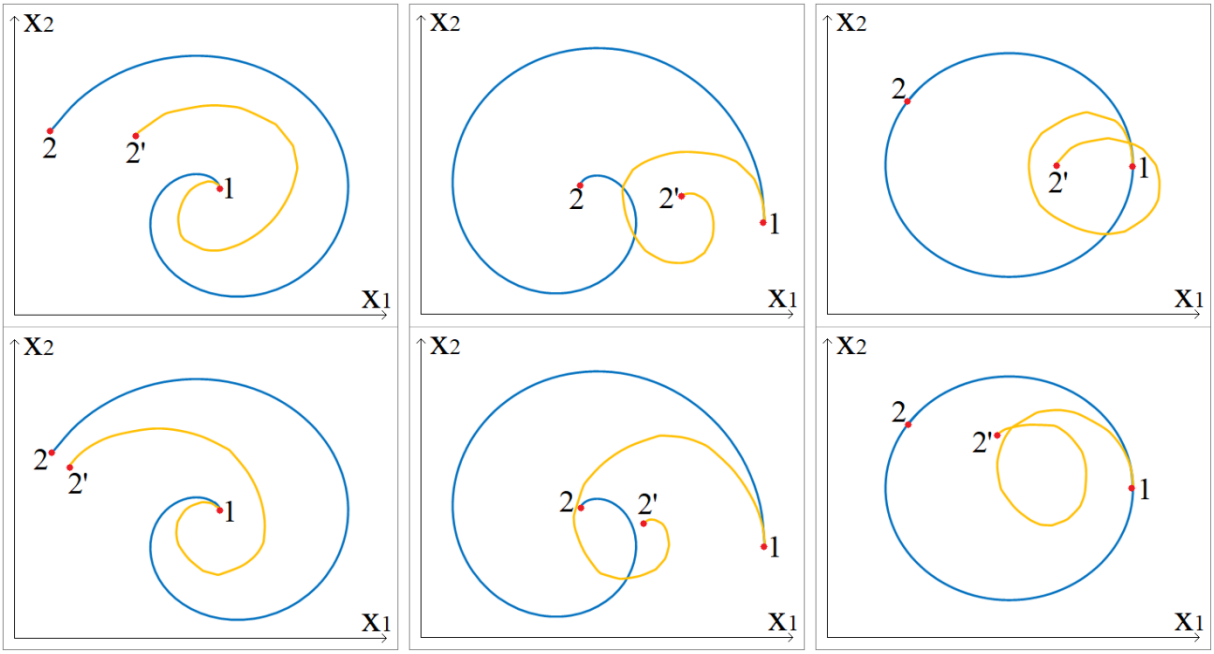
b) **Caso 2:** Um agente sempre melhora sua classificação no ranking, diminuindo o raio que indica seu movimento, até obter a melhor solução do grupo de agentes;

c) **Caso 3:** Um agente sempre apresenta a mesma classificação entre o grupo de agentes, mantendo constante o raio que rege o movimento.

Na Figura 4.3 é ilustrado o comportamento de um agente de pesquisa em cada uma dessas situações extremas hipotéticas aplicadas a um problema de duas dimensões: em (a) é descrito o comportamento apresentado no Caso 1, em (b) o comportamento exposto no Caso 2 e em (c) o comportamento descrito no Caso 3. Em cada quadro da Figura, o agente inicia o seu percurso na posição $\mathbf{1}$. A linha azul representa a trajetória do agente sem a inclusão da randomização (variáveis $rand$ nas Equações 4.7 e 4.8) enquanto a linha amarela representa uma das possíveis trajetórias quando a randomização é considerada. Desta forma, o agente termina seu percurso na posição $\mathbf{2}$ quando a randomização não é considerada e na posição $\mathbf{2'}$ quando se considera as variáveis randômicas. São apresentados dois exemplos para cada caso extremo, onde, em cada caso, observa-se que sem a randomização, o comportamento de um determinado agente é igual nos dois exemplos, ao passo que ao considerar as variáveis aleatórias/randômicas, o comportamento tende a mudar a cada execução.

De acordo com Yang (2010c), a inclusão da randomização torna-se importante pois aumenta a diversidade do algoritmo, estimulando-o a fugir de ótimos locais. No caso do *Circle-Inspired Optimization Algorithm*, a situação mais relevante é no Caso 3, conforme Figura 4.3 (c), situação em que o agente mantém a classificação constante ao longo de diversas iterações e, portanto, sua trajetória é regida por um raio constante, formando uma circunferência completa. Na eventual hipótese de os movimentos do agente superarem $360°$ com um raio constante, o agente descreveria um movimento de círculo perfeito, passando sempre pelos mesmos pontos do espaço de pesquisa, o que não é desejável.

Figura 4.3 – Comportamento de um agente de pesquisa em situações extremas



(a) (b) (c)

Desta forma, pode-se dizer que, na eventualidade de um agente de pesquisa, ao longo de sucessivas iterações, piorar a classificação de sua solução, a trajetória deste agente será de uma espiral que parte do centro para a extremidade (a). Na eventual situação oposta, a espiral descrita pelo agente parte do lado externo para um ponto central (b). No último caso, onde a classificação do agente no ranking é constante, a trajetória descrita lembrará um círculo, ainda que imperfeito quando são considerados os efeitos da randomização (c).

Durante as iterações de um algoritmo meta-heurístico, sempre que uma variável de projeto ultrapasse os limites estabelecidos pelo problema, o algoritmo deve ser capaz de substituir esta variável, de modo que ela retorne à faixa de valores permitida pelos limites (MIRJALILI E LEWIS, 2016). No algoritmo CIOA, se em determinada iteração k, o valor de uma variável qualquer x_i for maior que o limite superior U_b ou menor que o limite inferior L_b, esta variável receberá o valor correspondente à variável x_i do agente de pesquisa que obteve a melhor solução na presente iteração k. Caso esta ultrapassagem de limites ocorra no agente de pesquisa que fornece a melhor solução na iteração corrente, o valor da variável será substituído por U_b ou L_b, caso o limite ultrapassado seja, respectivamente, o superior ou o inferior.

Até então, o raio que cada agente de pesquisa utilizará para se mover é selecionado do vetor $\boldsymbol{r}$, que se mantém constante ao longo das iterações. Em outras palavras, o agente altera o elemento

r_j, dentro do vetor $\boldsymbol{r}$, a cada iteração, dependendo da classificação de sua avaliação da função objetivo, dentro de um conjunto que se mantém constante. Visando acelerar a convergência do algoritmo, é proposto que, de tempos em tempos, ocorra uma redução de todos os elementos que integram o vetor $\boldsymbol{r}$. Processos semelhantes, de redução do movimento dos agentes de pesquisa podem ser observados, por exemplo, em versões modernas do algoritmo PSO (SERAPIÃO, 2009), no algoritmo SGA (GONÇALVES ET AL., 2015) e no algoritmo WOA (MIRJALILI E LEWIS, 2016).

Desta forma, no algoritmo CIOA, toda vez que após k iterações os agentes completarem uma volta inteira, ou seja, $k\,.\,\theta$ ultrapassar um múltiplo de $360°$, ocorre a redução do vetor $\boldsymbol{r}$ conforme a Equação 4.10

$$\boldsymbol{r_{new}} = \boldsymbol{r}\,.\,0{,}99 \tag{4.10}$$

na qual $\boldsymbol{r_{new}}$ é o novo vetor de raios após a redução. O coeficiente de redução dos raios é fixado em $0{,}99$ devido ao fato deste valor dar maior estabilidade ao algoritmo em diferentes tipos de problemas de otimização. A análise completa da influência do coeficiente de redução no desempenho do algoritmo pode ser observada ao longo da seção 5.2 deste trabalho.

4.4 REFINAMENTO DA SOLUÇÃO ATRAVÉS DE PESQUISA LOCAL

Pode-se dizer que o laço principal do *Circle-Inspired Optimization Algorithm*, conforme formulado na seção 4.3, já promove pesquisas globais e locais simultaneamente. Isto porque os agentes com melhores soluções descrevem movimentos pequenos, correspondentes à pesquisa local enquanto os agentes com as piores soluções descrevem grandes movimentos, correspondentes à pesquisa global.

Todavia, torna-se importante que o algoritmo dedique algumas das iterações para uma pesquisa exclusivamente local, onde todos os agentes de pesquisa ficam restritos às áreas mais promissoras do espaço de busca. O processo de separação de iterações para a pesquisa exclusivamente local pode ser observado em algoritmos estudados neste trabalho, como, por exemplo, o SGA, desenvolvido por Gonçalves et al. (2015).

Para inserir pesquisa exclusivamente local ao CIOA, foi estabelecido um parâmetro $Glob_{It}$, que pode assumir valores no intervalo $(0{,}1]$. Este parâmetro irá definir a proporção de iterações utilizadas antes da pesquisa exclusivamente local (refinamento da solução) e seu valor deve ser

definido pelo usuário. Testes realizados neste trabalho, apontam que o CIOA apresenta maior estabilidade em um amplo número de problemas de otimização utilizando-se a faixa de valores correspondente a $0{,}75 \leq Glob_{It} \leq 0{,}95$. Estes testes, mostram ainda que, de modo geral, valores de $Glob_{It} \leq 0{,}75$ tendem a tornar os resultados ligeiramente mais precisos, porém interferem negativamente no tempo computacional. Em contrapartida, valores elevados, tais quais $Glob_{It} \geq 0{,}95$, reduzem o tempo computacional de operação, mas promovem a perda de precisão nos resultados. Os resultados de todos estes testes podem ser vistos, de forma gráfica, na seção 5.2 deste trabalho.

A pesquisa exclusivamente local iniciará na iteração k quando a razão entre k e o número total de iterações for maior que o parâmetro $Glob_{It}$. Neste momento, todos os agentes de pesquisa são reiniciados assumindo as variáveis/coordenadas que produziram a melhor solução até o momento, considerando-se todas as iterações transcorridas. No momento desta atualização, todos agentes terão seu primeiro movimento regido pelo raio mínimo, ou seja, o raio r_1 do vetor $\boldsymbol{r}$. Além disso, é realizado uma alteração nos limites superior U_b e inferior L_b, de modo que os novos limites para cada variável x_i, denotados U_{b1_i} e L_{b1_i} são dados em função da variável $x_{i_{best}}$, considerada a variável na dimensão i que produziu a melhor solução até o momento, conforme as equações 4.11 e 4.12

$$U_{b1_i} = x_{i_{best}} + \frac{U_b - L_b}{10000} \tag{4.11}$$

$$L_{b1_i} = x_{i_{best}} - \frac{U_b - L_b}{10000} \tag{4.12}$$

O denominador das equações 4.11 e 4.12, responsável pela redução dos limites de variáveis tem seu valor fixado em 10000. Testes de sensibilidade foram realizados na seção 5.2, onde revelam que o melhor desempenho do CIOA ocorre justamente quando o denominador de redução dos limites de variáveis apresenta valores desta ordem de grandeza.

Para os casos específicos em que $L_{b1_i} < L_b$ ou $U_{b1_i} > U_b$, sempre que o valor de uma variável de projeto estiver nos intervalos $L_{b1_i} < x_i < L_b$ ou $U_{b1_i} > x_i > U_b$, o valor desta variável será automaticamente atualizado para, respectivamente, $x_i = L_b$ ou $x_i = U_b$

Uma vez inicializada, esta etapa de refinamento das soluções é regida pelas mesmas equações e formulações do Loop principal, descrito na Seção 4.3, substituindo-se apenas U_b por U_{b1} e L_b por L_{b1}. Deste modo, restringe-se o espaço de busca dos agentes, fazendo com que os possíveis

valores para uma determinada variável estejam muito próximos do melhor valor obtido nas iterações do *Loop* principal do algoritmo.

4.5 INFORMAÇÕES RELEVANTES AO USUÁRIO DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM*

Nas seções anteriores apresentou-se a formulação do *Circle-Inspired Optimization Algorithm* (CIOA): o novo algoritmo meta-heurístico de otimização desenvolvido neste trabalho. Nesta seção, são adicionadas informações importantes que devem ser consideradas pelos usuários do CIOA, de modo a tirar o melhor proveito do algoritmo proposto.

a) Definir $0{,}75 \leq Glob_{It} \leq 0{,}95$. Recomenda-se o valor de $Glob_{It} = 0{,}85$, que promove um bom equilíbrio entre precisão e tempo computacional. Valores maiores de $Glob_{it}$ podem reduzir o tempo computacional de operação, enquanto valores menores tendem a aumentar a precisão do algoritmo;

b) Teoricamente, o ângulo θ pode ser definido com qualquer valor entre 1 e 360 graus. Entretanto, testes realizados neste trabalho indicam que o algoritmo tem um desempenho superior para valores de θ relativamente pequenos e que, preferencialmente, não sejam divisores de 360. Valores sugeridos com bom desempenho são de $\theta = 13°$, $\theta = 17°$ ou $\theta = 19°$.

c) O algoritmo tem um melhor desempenho com um número de agentes de pesquisa relativamente alto (acima de 100) e com uma relação aproximada de, entre 1 para 3 e 1 para 5 entre agentes de pesquisa e número de iterações. Portanto, num exemplo hipotético de um problema onde desejam ser utilizadas 40000 avaliações, é preferível que estas avaliações sejam construídas por 100 agentes e 400 iterações do que por 40 agentes e 1000 iterações.

O conteúdo destas sugestões é corroborado com a análise de sensibilidade do algoritmo, apresentada na Seção 5.2 do presente trabalho.

Na Figura 4.4 é ilustrado o pseudocódigo do *Circle-Inspired Optimization Algorithm* (CIOA).

Figura 4.4 – Pseudocódigo do *Circle-Inspired Optimization Algorithm* (CIOA)

Início
Defina θ *e* $Glob_{It}$
Inicialize um vetor de raios $\boldsymbol{r}$ *de acordo com a Eq. 4.5*
Atribua valores aleatórios às variáveis de projeto conforme Eq. 4.7
Avalie a função objetivo para cada agente de pesquisa
Enquanto 1 *(*$k \leq Glob_{It}$ x *Número máximo de iterações)*
Classifique os agentes de pesquisa quanto à qualidade da solução obtida
Atualize a posição dos agentes através das Eq. 4.8 e 4.9
Verifique se nenhuma variável de projeto excede os limites impostos
Se k *for um múltiplo do valor arredondado para baixo de* $360/\theta$
Atualize $\boldsymbol{r}$ *de acordo com a Eq. 4.10*
Fim Se
Fim Enquanto 1
Atribua a todos os agentes a posição que gerou a melhor solução até o momento
Atualize os limites de variáveis conforme as Eq. 4.11 e 4.12
Enquanto 2 *(*$k \leq$ *Número máximo de iterações)*
Repita os procedimentos de **Enquanto 1**, *usando os novos limites de variáveis*
Fim Enquanto 2
Visualização dos resultados
Fim

De modo geral, o grande diferencial do *Circle-Inspired Optimization Algorithm* frente a outros algoritmos, se dá na sua fácil implementação e no número reduzido de parâmetros definidos pelo usuário. Nos próximos capítulos, análises de desempenho apresentadas indicam que o CIOA é extremamente competitivo e, muitas vezes, melhor do que outros algoritmos meta-heurísticos de otimização, especialmente quando aplicado em problemas clássicos de engenharia estrutural.

5 VALIDAÇÃO DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM* E AVALIAÇÃO DOS PARÂMETROS

Neste capítulo valida-se o *Circle-Inspired Optimization Algorithm*, ou seja, o algoritmo desenvolvido no presente trabalho. Na seção 5.1 deste capítulo, é realizada a avaliação do CIOA mediante a otimização de 15 funções *benchmark*, também conhecidas como funções de teste ou de referência. Na seção 5.2 do presente capítulo, realiza-se uma análise de sensibilidade do algoritmo, avaliando o seu comportamento mediante a alteração dos principais parâmetros indicados na formulação.

5.1 VALIDAÇÃO ATRAVÉS DA OTIMIZAÇÃO DE FUNÇÕES *BENCHMARK*

Um algoritmo meta-heurístico pode ser validado através de uma análise estatística simples dos resultados obtidos após inúmeras simulações em problemas de otimização global. Para isto, comumente são empregadas funções *benchmark* – ou funções de teste. De acordo com Arora e Singh (2019), as funções *benchmark* podem avaliar a capacidade do algoritmo em convergir com boa velocidade, em fugir de ótimos locais e em evitar a convergência prematura.

Desta forma, o *Circle-Inspired Optimization Algorithm* foi validado através da comparação entre o seu desempenho e o desempenho de cinco algoritmos meta-heurísticos consagrados pela literatura e amplamente utilizados em problemas de otimização. A escolha desses cinco algoritmos utilizados na comparação com o CIOA se baseou nos seguintes critérios:

a) Selecionar algoritmos elaborados por diferentes autores, uma vez que algoritmos criados por um mesmo autor podem apresentar semelhanças em suas formulações;

b) Buscar por algoritmos criados em épocas distintas, uma vez que existem algoritmos antigos extremamente competitivos ainda na atualidade e, ao mesmo tempo, algoritmos relativamente modernos que já alcançaram excelente reputação graças ao seu bom desempenho;

c) Utilizar, ao menos, um algoritmo desenvolvido por autores brasileiros, a saber: *Search Group Algorithm* (SGA).

Deste modo, na Tabela 5.1, é apresentado um resumo das informações de cada um dos algoritmos utilizados para a validação do CIOA.

Tabela 5.1 – Informações dos algoritmos avaliados neste trabalho

Algoritmo	Autoria	Ano
Particle Swarm Optimization (PSO)	Kennedy e Russell	1995
Harmony Search (HS)	Geem et al.	2001
Firefly Algorithm (FA)	Yang	2009
Search Group Algorithm (SGA)	Gonçalves et al.	2015
Whale Optimization Algorithm (WOA)	Mirjalili e Lewis	2016

5.1.1 Procedimento para a validação

Para promover a validação do *Circle-Inspired Optimization Algorithm*, quinze problemas de otimização de funções de teste foram solucionados através do CIOA e dos algoritmos apresentados na Tabela 5.1.

Os problemas utilizados para a validação do algoritmo consistem na minimização de 11 funções *benchmark*, apresentadas na Tabela 5.2; na minimização de duas funções com restrições, apresentadas na Tabela 5.3 e na maximização de duas funções *benchmark*, apresentadas na Tabela 5.4. Estas funções, amplamente utilizadas na validação de algoritmos meta-heurísticos, podem ser encontradas na bibliografia, como, por exemplo, nos trabalhos de Mirjalili e Lewis (2016) e de Arora e Singh (2019).

No presente trabalho, dez variáveis de projeto foram consideradas nas funções sem dimensões fixas, ou seja, nas funções de f_1 a f_8 da Tabela 5.2 e na função f_{14} da tabela 5.4. Nas funções restritas, correspondentes à f_{12} e f_{13} da Tabela 5.3, métodos de penalidade foram utilizados para lidar com as restrições.

Tabela 5.2 – Minimização de funções *benchmark*

Função	Intervalo das Variáveis
$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]
$f_2(\boldsymbol{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30, 30]
$f_3(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i + 0{,}5)^2$	[-100, 100]
$f_4(\boldsymbol{x}) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0,1)$	[-1,28, 1,28]
$f_5(\boldsymbol{x}) = \sum_{i=1}^{n} -x_i \,\text{sen}\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]
$f_6(\boldsymbol{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5,12, 5,12]
$f_7(\boldsymbol{x}) = -20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-32, 32]
$f_8(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]
$f_9(\boldsymbol{x}) = \left(x_2 - \frac{5{,}1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	[-5, 5]
$f_{10}(\boldsymbol{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2]$ $\times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	[-2, 2]
$f_{11}(\boldsymbol{x}) = x_1^2 + x_2^2 + 25(\text{sen}^2 x_1 + \text{sen}^2 x_2)$	[-2π, 2π]

Tabela 5.3 – Minimização de funções com restrições

Função	Intervalo das Variáveis
$f_{12}(\boldsymbol{x}) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$ Sujeito a: $g_1(x) = -2x_1 - x_2 + 4 \leq 0$ e $g_2(x) = -x_1 - 2x_2 + 4 \leq 0$	[-50, 50]
$f_{13}(\boldsymbol{x}) = (x_1 - 1)^2 + (x_2 - 5)^2$ Sujeito a: $g_1(x) = -x_1^2 + x_2 \leq 4$ e $g_2(x) = -(x_1 - 2)^2 + x_2 \leq 3$	[-50, 50]

Tabela 5.4 – Maximização de funções *benchmark*

Função	Intervalo das Variáveis
$f_{14}(\boldsymbol{x}) = -\sum_{i=1}^{n} x_i^2$	[-100, 100]
$f_{15}(\boldsymbol{x}) = -(4x_1^2 - 2{,}1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4)$	[-5, 5]

Cada problema de otimização de funções *benchmark* foi solucionado 50 vezes por cada um dos algoritmos considerados. Denomina-se **execução** ou **simulação** cada uma das 50 vezes em que o problema é solucionado por um determinado algoritmo. O número elevado de simulações é extremamente importante uma vez que, devido às caraterísticas randômicas de cada algoritmo meta-heurístico, os resultados de cada simulação podem ser distintos e, não raramente, o grau desta distinção pode ser alto. Autores como Mirjalili e Lewis (2016) consideram que o número adequado de execuções para a otimização de funções *benchmark* seja igual ou superior a 30.

Em cada execução realizada por um algoritmo meta-heurístico, o operador deve definir o número de **avaliações** da função objetivo. O número de avaliações definido terá grande influência na qualidade dos resultados produzidos pelo algoritmo, de modo que um maior número de avaliações tende a produzir resultados melhores, mas, em contrapartida, aumenta-se o tempo computacional de operação. Para realizar uma comparação justa de desempenho, é importante que todos os algoritmos de otimização considerados trabalhem com o mesmo número de **avaliações** da função objetivo em cada problema. Para o caso da otimização de funções *benchmark*, diferentes autores utilizam diferentes números de avaliações. Por exemplo, Mirjalili e Lewis (2016) utilizaram 15000 avaliações para a validação do WOA, enquanto Civicioglu (2013) utilizou em seu trabalho 2000000 avaliações. No presente trabalho, todas as funções *benchmark* apresentadas neste capítulo foram otimizadas através de 200000 avaliações.

O número total de avaliações utilizado por cada algoritmo, pode ser obtido pela multiplicação entre o número de **agentes de pesquisa** e o número de **iterações**. Entre os algoritmos abordados no presente trabalho, o *Harmony Search* (HS) é o único que possui avaliação única em cada iteração, assim, o número de iterações no HS é igual ao número de avaliações. Portanto, as 200000 avaliações consideradas no presente trabalho são construídas da seguinte forma:

- 800 iterações e 250 agentes de pesquisa para os algoritmos CIOA, WOA, SGA e PSO;
- 5000 iterações e 40 agentes de pesquisa para o algoritmo FA;
- 200000 iterações para o algoritmo HS.

Os valores obtidos são justificados com base no funcionamento de cada algoritmo. Os algoritmos CIOA, WOA, SGA e PSO apresentam um ótimo desempenho quando o número de agentes de pesquisa é elevado. Em contrapartida, o algoritmo FA torna-se mais eficiente, em termos de tempo computacional, com um número relativamente baixo de agentes de pesquisa,

aspecto geralmente levado em consideração nos trabalhos que utilizam este algoritmo, como por exemplo, Miguel e Fadel Miguel (2013) e Borges (2013). Em relação aos parâmetros do CIOA, todos os problemas foram executados utilizando-se $Glob_{It} = 0{,}85$ e $\theta = 17°$. Novamente ressalta-se que na seção 5.2 é apresentada uma análise de sensibilidade, que demonstra como o comportamento do algoritmo pode variar de acordo com os valores atribuídos a cada parâmetro. Este estudo em questão, corrobora os valores atribuídos para os parâmetros $Glob_{It}$ e θ na otimização das funções *benchmark*.

5.1.2 Resultados estatísticos da validação

Os resultados apresentados nesta seção foram obtidos após 50 execuções de cada algoritmo e consistem na média, no desvio padrão e no coeficiente de variação do valor ótimo obtido para cada uma das 15 funções após todas as simulações.

Na Tabela 5.5, são apresentados os valores médios para o ótimo global das funções analisadas. Os resultados obtidos em cada algoritmo foram comparados com o valor ótimo exato de cada função, conhecido da literatura. Observa-se que o desempenho de cada algoritmo varia de acordo com as características da função otimizada, isto é, algoritmos com uma boa performance numa determinada função podem apresentar uma performante inferior em outras funções.

Tabela 5.5 – Valor médio do ótimo global para as funções analisadas

f	Exato	PSO	HS	FA	SGA	WOA	CIOA
f_1	0	**0**	2,26E-04	0,0831	1,39E-03	**0**	1,77E-04
f_2	0	2,8613	5,9068	8,5028	3,1304	4,6460	**2,8346**
f_3	0	**0**	7,94E-09	0,0639	2,90E-05	1,23E-11	1,57E-07
f_4	0	4,23E-04	4,33E-04	0,0263	0,4839	**3,21E-04**	1,01E-03
f_5	-4189,83	-2510,28	**-4189,83**	-3160,32	-3393,05	-3567,34	-3481,39
f_6	0	10,5665	**1,42E-06**	9,0029	1,6914	6,5292	0,2331
f_7	0	5,08E-15	1,08E-04	0,1831	2,16E-03	**0**	1,97E-04
f_8	0	0,0889	0,0752	0,3582	1,59E-03	0,2001	**5,39E-07**
f_9	0,3979	**0,3979**	**0,3979**	**0,3979**	**0,3979**	**0,3979**	**0,3979**
f_{10}	3	**3**	**3**	**3**	**3**	**3**	**3**
f_{11}	0	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	0,2222	**0,2222**	**0,2222**	0,2244	0,2223	0,2224	**0,2222**
f_{13}	0,2539	**0,2539**	**0,2539**	0,2658	0,2541	0,2543	**0,2539**
f_{14}	0	**0**	-6,79E-09	-0,0667	-3,13E-05	**0**	-1,73E-07
f_{15}	1,0316	**1,0316**	**1,0316**	**1,0316**	**1,0316**	**1,0316**	**1,0316**

De acordo com os resultados apresentados na Tabela 5.5, o *Circle-Inspired Optimization Algorithm* (CIOA), proposto neste trabalho, apresentou resultados compatíveis com os demais algoritmos e obteve o melhor desempenho, de forma isolada, nas funções f_2 e f_8, obteve ainda o melhor desempenho, juntamente com o PSO e o HS nas funções com restrições f_{12} e f_{13}. Nas funções de f_9 a f_{11}, bem como na função f_{15} o algoritmo CIOA juntamente como todos os demais algoritmos alcançaram o resultado exato. De modo geral, pode-se dizer que o CIOA apresentou o melhor desempenho, de forma isolada ou juntamente com outros algoritmos, em 8 das 15 funções analisadas.

No Quadro 5.1 é ilustrado um ranking onde os algoritmos são classificados de acordo com os resultados ilustrados na Tabela 5.5. Deste modo, nos problemas de minimização, a classificação ocorre de acordo com a ordem crescente dos resultados enquanto nos problemas de maximização, o ranking é feito mediante a ordem decrescente dos resultados obtidos apresentados na Tabela 5.5.

Quadro 5.1 – Ranking do valor médio do ótimo global

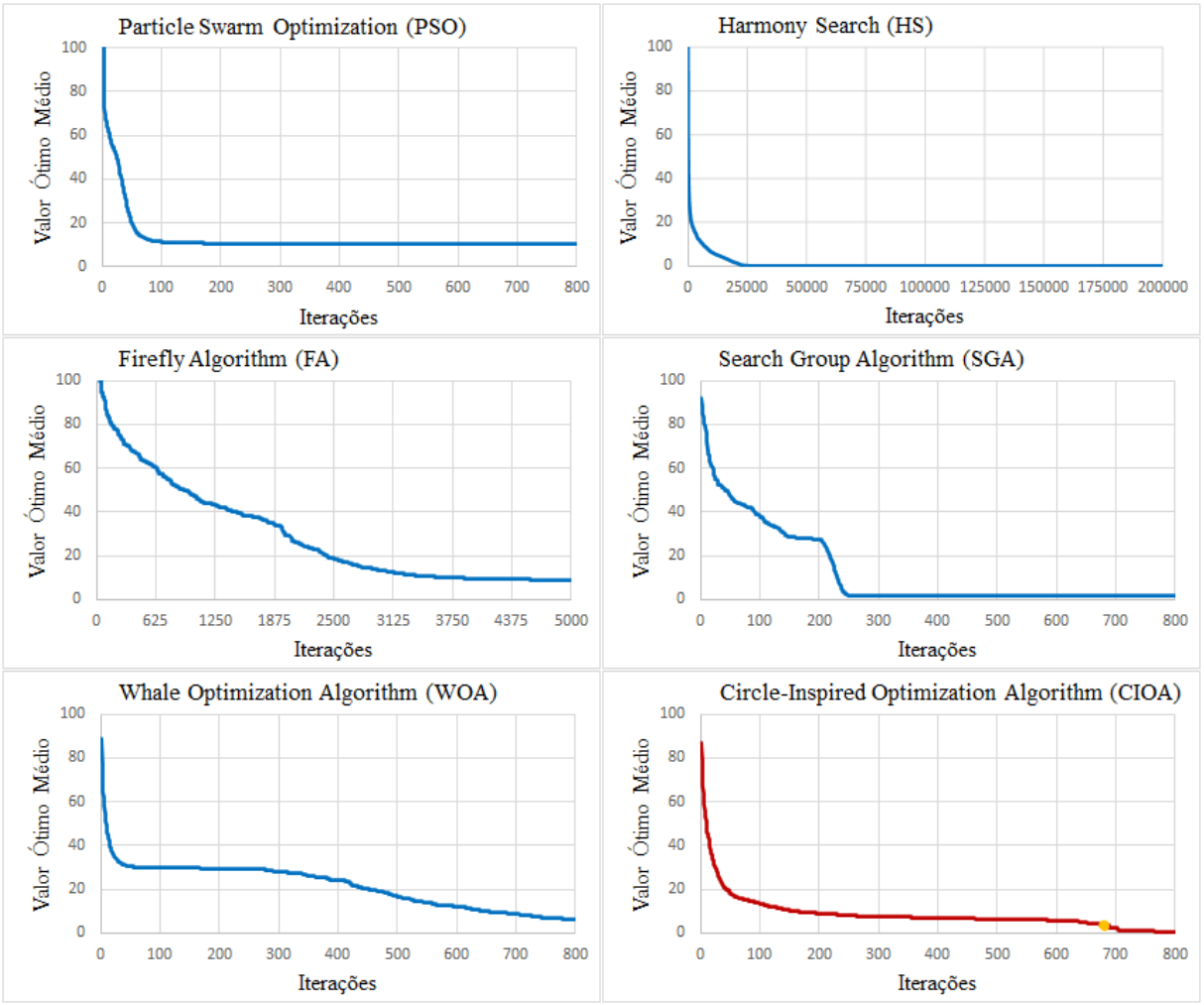
Tipo de Problema	f	Ranking
Minimização	f_1	[WOA = PSO] < [**CIOA**] < [HS] < [SGA] < [FA]
	f_2	[**CIOA**] < [PSO] < [SGA] < [WOA] < [HS] < [FA]
	f_3	[PSO] < [WOA] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_4	[WOA] < [PSO] < [HS] < [**CIOA**] < [FA] < [SGA]
	f_5	[HS] < [WOA] < [**CIOA**] < [SGA] < [FA] < [PSO]
	f_6	[HS] < [**CIOA**] < [SGA] < [WOA] < [FA] < [PSO]
	f_7	[WOA] < [PSO] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_8	[**CIOA**] < [SGA] < [HS] < [PSO] < [WOA] < [FA]
	f_9	[**CIOA** = WOA = SGA = FA = HS = PSO]
	f_{10}	[**CIOA** = WOA = SGA = FA = HS = PSO]
	f_{11}	[**CIOA** = WOA = SGA = FA = HS = PSO]
Minimização com restrições	f_{12}	[**CIOA** = HS = PSO] < [SGA] < [WOA] < [FA]
	f_{13}	[**CIOA** = HS = PSO] < [SGA] < **[**WOA**]** < [FA]
Maximização	f_{14}	[WOA = PSO] > [HS] > [**CIOA**] > [SGA] > [FA]
	f_{15}	[**CIOA** = WOA = SGA = FA = HS = PSO]

Com base no Quadro 5.1, nota-se que o CIOA apresentou uma regularidade de desempenho ao longo de todas as funções otimizadas, estando na grande maioria dos casos classificado entre as melhores posições ou ainda em posições intermediárias, quando comparado com os demais

algoritmos. Salienta-se que o CIOA foi o único, dos seis algoritmos em análise, que não ocupou a última ou a penúltima posição, correspondente aos piores desempenhos, em nenhuma das quinze funções analisadas.

Apresenta-se na Figura 5.1, a nível de exemplo, ilustrações das curvas de convergência geradas por cada algoritmo na otimização da função f_6. Esta função foi a escolhida para a análise de convergência devido ao fato de ser uma função multimodal, apresentando inúmeros pontos de ótimos locais, o que aumenta a complexidade de sua otimização e, por consequência, a dificuldade da convergência dos algoritmos no ótimo Global.

Figura 5.1 – Curvas de Convergência para a Função f_6



Observa-se que o *Circle-Inspired Optimization Algorithm* apresenta uma taxa de convergência relativamente rápida nas primeiras iterações, onde a busca global e a busca local ocorrem simultaneamente, chegando a resultados interessantes dentro de áreas promissoras ainda nas primeiras 300 iterações. Na etapa de pesquisa exclusivamente local, iniciada a partir da iteração

680 (ponto amarelo do gráfico no CIOA), a busca é restrita à área mais promissora do espaço de pesquisa, promovendo uma melhoria significativa do resultado. O eventual aumento de iterações dedicadas exclusivamente à pesquisa local poderia aumentar ainda mais a precisão do algoritmo, todavia, testes realizados sugeriram que o algoritmo CIOA consome um maior tempo computacional quando se aumenta a proporção de iterações dedicadas à busca local. Os resultados destes testes são apresentados na seção 5.2.3.

A Tabela 5.6 aponta os resultados em termos de desvio padrão, após 50 execuções de cada algoritmo na otimização das funções analisadas. Nota-se que um mesmo algoritmo produziu diferentes taxas de desvio padrão a depender da complexidade da função analisada. Em funções simples como, por exemplo, de f_9 a f_{11} o desvio padrão foi nulo para todos os algoritmos, isto é, a solução **exata** de ótimo local foi alcançada em todas as simulações. Já em funções mais complexas, como a f_5, o desvio padrão entre os resultados de cada execução foi bastante elevado em vários algoritmos, revelando distinções consideráveis a cada nova simulação, o que indica que os algoritmos nem sempre apresentam uma estabilidade na precisão das soluções geradas. Os valores de desvio padrão obtidos pelo *Circle-Inspired Optimization Algorithm* em cada função otimizada são compatíveis com os valores produzidos pelos demais algoritmos.

Tabela 5.6 – Desvio padrão para as funções analisadas

f	PSO	HS	FA	SGA	WOA	CIOA
f_1	**0**	6,12E-05	0,0104	1,86E-04	**0**	1,32E-04
f_2	1,7655	6,3286	3,1127	**0,6464**	9,7448	1,7100
f_3	**0**	3,59E-09	0,0105	8,42E-06	8,83E-12	3,39E-08
f_4	2,25E-04	**2,14E-04**	0,0242	0,2846	2,41E-04	3,06E-04
f_5	301,63	**1,72E-10**	211,42	156,80	361,00	143,23
f_6	4,3294	**3,27E-07**	3,6416	1,0299	7,1477	0,4124
f_7	1,38E-15	1,04E-05	0,0270	3,37E-04	**0**	2,76E-05
f_8	0,0502	0,0222	0,0703	6,56E-03	0,1122	**7,89E-07**
f_9	**0**	**0**	**0**	**0**	**0**	**0**
f_{10}	**0**	**0**	**0**	**0**	**0**	**0**
f_{11}	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	**0**	**0**	1,12E-03	4,07E-05	1,53E-04	**0**
f_{13}	**0**	**0**	5,16E-03	1,31E-04	4,12E-04	**0**
f_{14}	**0**	1,43E-09	0,0135	1,02E-05	**0**	3,88E-08
f_{15}	**0**	**0**	**0**	**0**	**0**	**0**

No Quadro 5.2 apresenta-se um ranking dos algoritmos, levando em consideração a ordem crescente entre os valores de desvio padrão obtidos para cada função. Nota-se que o CIOA ocupou, na grande maioria das funções, posições intermediárias no ranking, enquanto grande parte dos demais algoritmos ora ficou entre as duas primeiras posições, ora entre as duas últimas.

Quadro 5.2 – Ranking do valor de desvio padrão

Tipo de Problema	f	Ranking
Minimização	f_1	[WOA = PSO] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_2	[SGA] < [**CIOA**] < **[**PSO**]** < [FA] < [HS] < [WOA]
	f_3	[PSO] < [WOA] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_4	[HS] < [PSO] < [WOA] < [**CIOA**] < [FA] < [SGA]
	f_5	[HS] < [**CIOA**] < [SGA] < **[**FA**]** < [PSO] < [WOA]
	f_6	[HS] < [**CIOA**] < [SGA] < [FA] < [PSO] < [WOA]
	f_7	[WOA] < [PSO] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_8	[**CIOA**] < [SGA] < [HS] < [PSO] < [FA] < [WOA]
	f_9	[**CIOA** = WOA = SGA = FA = HS = PSO]
	f_{10}	[**CIOA** = WOA = SGA = FA = HS = PSO]
	f_{11}	[**CIOA** = WOA = SGA = FA = HS = PSO]
Minimização com restrições	f_{12}	[**CIOA** = HS = PSO] < [SGA] < [WOA] < [FA]
	f_{13}	[**CIOA** = HS = PSO] < [SGA] < [WOA] < [FA]
Maximização	f_{14}	[WOA = PSO] < [HS] < [**CIOA**] < [SGA] < [FA]
	f_{15}	[**CIOA** = WOA = SGA = FA = HS = PSO]

Foi realizada uma avaliação do coeficiente de variação entre as 50 simulações de cada algoritmo, na qual os valores percentuais de coeficiente de variação foram calculados através da Equação 5.1

$$Coeficiente\ de\ Variação = \frac{Desvio\ Padrão}{Média} . 100 \tag{5.1}$$

Para esta análise foram consideradas apenas as funções *benchmark* cujo valor ótimo exato seja diferente de zero, visando evitar casos em que o valor da média seja quase nulo, pois estes tendem a causar distorções nos valores de Coeficiente de Variação. Na Tabela 5.7 são apresentados os coeficientes obtidos para as funções analisadas.

Tabela 5.7 – Coeficiente de variação para as funções com ótimo diferente de zero (%)

f	PSO	HS	FA	SGA	WOA	CIOA
f_5	12,016	**0**	6,690	4,621	10,119	3,564
f_9	**0**	**0**	**0**	**0**	**0**	**0**
f_{10}	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	**0**	**0**	0,500	0,018	0,069	**0**
f_{13}	**0**	**0**	1,940	0,052	0,162	**0**
f_{15}	**0**	**0**	**0**	**0**	**0**	**0**

Observa-se que os coeficientes de variação obtidos na maior parte dos problemas são relativamente baixos, com exceção da função f_5, onde alguns algoritmos apresentam coeficientes de variação superiores a 10%. Todavia, o CIOA teve um ótimo desempenho em todas as funções analisadas.

No Quadro 5.3 apresenta-se um ranking dos algoritmos, levando em consideração a ordem crescente entre os valores de desvio padrão obtidos para cada função.

Quadro 5.3 – Ranking do coeficiente de variação

Tipo de Problema	f	Ranking
Minimização	f_5	[HS] < [**CIOA**] < [SGA] < **[**FA**]** < [WOA] < [PSO]
	f_9	[**CIOA** = WOA = SGA = FA = HS = PSO]
	f_{10}	[**CIOA** = WOA = SGA = FA = HS = PSO]
Minimização com restrições	f_{12}	[**CIOA** = HS = PSO] < [SGA] < [WOA] < [FA]
	f_{13}	[**CIOA** = HS = PSO] < [SGA] < [WOA] < [FA]
Maximização	f_{15}	[**CIOA** = WOA = SGA = FA = HS = PSO]

Verifica-se que o CIOA apresentou a melhor ou segunda melhor performance em todas as funções.

5.1.3 Tempo Computacional

Ao validar e avaliar o desempenho de um novo algoritmo meta-heurístico, um aspecto que pode assumir grande relevância é o tempo computacional necessário para cada operação. É importante ressaltar que este tempo operacional pode variar de acordo com a plataforma computacional utilizada. No presente trabalho, todas as análises foram realizadas utilizando um processador Intel Core i5 – 8ª geração com 8GB de RAM e sistema Windows 10 Home.

Todavia, devem ser mencionadas algumas **ressalvas** na análise de tempo computacional, uma vez que o tempo de operação de cada algoritmo é altamente dependente de inúmeros fatores, tais quais, estilo de programação, organização das variáveis. Outro aspecto importante que deve ser levado em consideração é que, embora todos os algoritmos tenham resolvido todos os problemas com o mesmo número de avaliações da função objetivo, as notáveis diferenças nas formulações de cada algoritmo podem demandar diferentes faixas de tempo computacional para cada avaliação em cada algoritmo analisado.

Na Tabela 5.7 são apresentados os tempos computacionais médios por execução de cada algoritmo nas 15 funções otimizadas. Salienta-se novamente que o número total de avaliações utilizadas por cada algoritmo foi o mesmo: 200000. O algoritmo CIOA apresentou o menor tempo computacional nas funções de f_9 a f_{13} e na função f_{15} que são as funções com dimensões fixas. Nas demais funções, sem dimensões fixas, o tempo computacional do CIOA não superou o baixo tempo apresentado pelo SGA, mas seus valores ficaram relativamente próximos entre si e muito abaixo dos tempos dos demais algoritmos.

Tabela 5.8 – Tempo computacional de operação (s)

f	PSO	HS	FA	SGA	WOA	CIOA
f_1	9,60	10,99	7,09	**2,00**	5,63	3,19
f_2	10,08	11,09	7,46	**2,28**	5,86	3,32
f_3	9,40	10,90	7,17	**2,00**	5,59	3,07
f_4	9,89	11,51	7,57	**2,30**	5,85	3,31
f_5	11,00	11,68	7,69	**2,44**	6,12	3,49
f_6	9,47	11,08	7,27	**2,17**	5,76	3,27
f_7	10,83	12,07	7,61	**2,45**	5,97	3,58
f_8	9,52	10,92	7,52	**2,33**	5,73	3,42
f_9	9,16	5,51	6,84	1,86	2,08	**1,17**
f_{10}	9,27	5,59	6,96	1,92	2,24	**1,33**
f_{11}	10,01	5,92	7,14	1,84	2,16	**1,17**
f_{12}	9,29	5,50	6,94	1,85	2,15	**1,19**
f_{13}	9,14	5,55	6,96	1,88	2,14	**1,19**
f_{14}	9,90	10,64	7,20	**2,10**	5,67	3,09
f_{15}	9,07	5,61	7,10	1,94	2,27	**1,27**

No Quadro 5.4 é apresentado um ranking levando-se em conta o tempo computacional de cada algoritmo na otimização das funções *benchmark* e das funções com restrições.

Quadro 5.4 – Ranking do tempo computacional

Tipo de Problema	f	Ranking
Minimização	f_1	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_2	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_3	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_4	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_5	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_6	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_7	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_8	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_9	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]
	f_{10}	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]
	f_{11}	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]
Minimização com restrições	f_{12}	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]
	f_{13}	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]
Maximização	f_{14}	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]
	f_{15}	[**CIOA]** < [SGA] < [WOA] < [HS] < [FA] < PSO]

Nota-se que o CIOA obteve sempre o melhor ou o 2º melhor desempenho, mostrando-se um algoritmo extremamente atrativo, uma vez que possui uma velocidade de execução superior a maior parte dos algoritmos com o qual foi comparado.

5.2 ANÁLISE DE SENSIBILIDADE DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM*

No Capítulo 4 do presente trabalho, ao longo da formulação do *Circle-Inspired Optimization Algorithm*, foram apresentados coeficientes com valores fixos e parâmetros que podem ser alterados pelo usuário do algoritmo. Nesta seção apresenta-se uma análise detalhada da sensibilidade do algoritmo CIOA mediante a alteração destes parâmetros e coeficientes.

Para cada parâmetro que pode ser alterado, verifica-se o comportamento do algoritmo, em termos de média, desvio padrão e tempo computacional, na otimização de três das funções *benchmark* utilizadas na validação do CIOA na seção 5.1. As funções selecionadas para estas análises são as funções f_3, f_5 e f_6, já apresentadas na Tabela 5.2 da subseção 5.1.1. A escolha destas funções justifica-se devido ao fato de elas possuírem características distintas, o que contribui para que o algoritmo tenha comportamentos diferentes mediante a otimização de cada

uma delas. A função f_3 é unimodal e apresenta valor ótimo exato igual zero; a função f_5 trata-se de uma função multimodal onde o valor ótimo é diferente de zero; já a função f_6 é uma função multimodal com valor ótimo igual a zero.

Os parâmetros ou coeficientes cuja variação é objeto de estudo em cada análise de sensibilidade são descritos na sequência.

a) Número de agentes de pesquisa e número de iterações;

b) Parâmetro θ;

c) Parâmetro $Glob_{It}$;

d) Coeficiente de atualização do vetor de raios;

e) Denominador de atualização dos limites de variáveis.

Na sequência são apresentados os resultados de cada uma das análises de sensibilidade realizadas neste trabalho, obtidos após 50 simulações do CIOA em cada função.

5.2.1 Análise de sensibilidade considerando a variação do número de agentes de pesquisa e do número de iterações

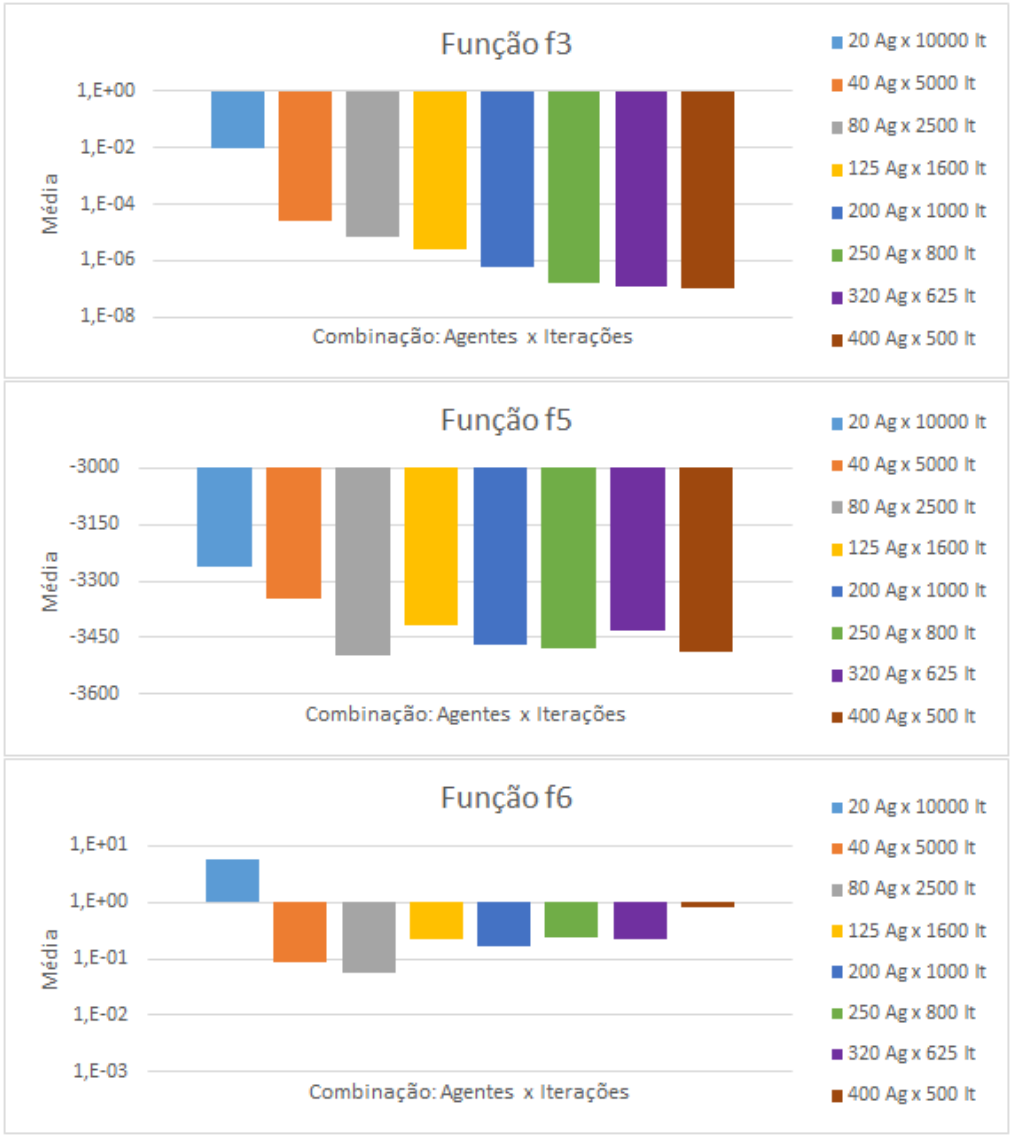
Nesta análise, verificou-se o comportamento do algoritmo mediante oito diferentes combinações de número de agentes de pesquisa e número de iterações, descritas na Tabela 5.9. Salienta-se que, em todas as combinações, a multiplicação entre o número de agentes de pesquisa e o número de iterações resulta no mesmo valor. Durante esta análise de sensibilidade, demais parâmetros foram fixados como $\theta = 17°$ e $Glob_{It} = 0{,}85$.

Tabela 5.9 – Combinações de número de agentes de pesquisa e número de iterações

Combinação	Número de Agentes de Pesquisa	Número de Iterações
Combinação 1	20	10000
Combinação 2	40	5000
Combinação 3	80	2500
Combinação 4	125	1600
Combinação 5	200	1000
Combinação 6	250	800
Combinação 7	320	625
Combinação 8	400	500

Na Figura 5.2 apresentam-se gráficos dos resultados obtidos em termos de média para cada combinação de agentes e iterações nas três funções analisadas. Para uma melhor visualização dos resultados, os gráficos das funções f_3 e f_6 estão em escala logarítmica, com origem das barras no valor de 1.

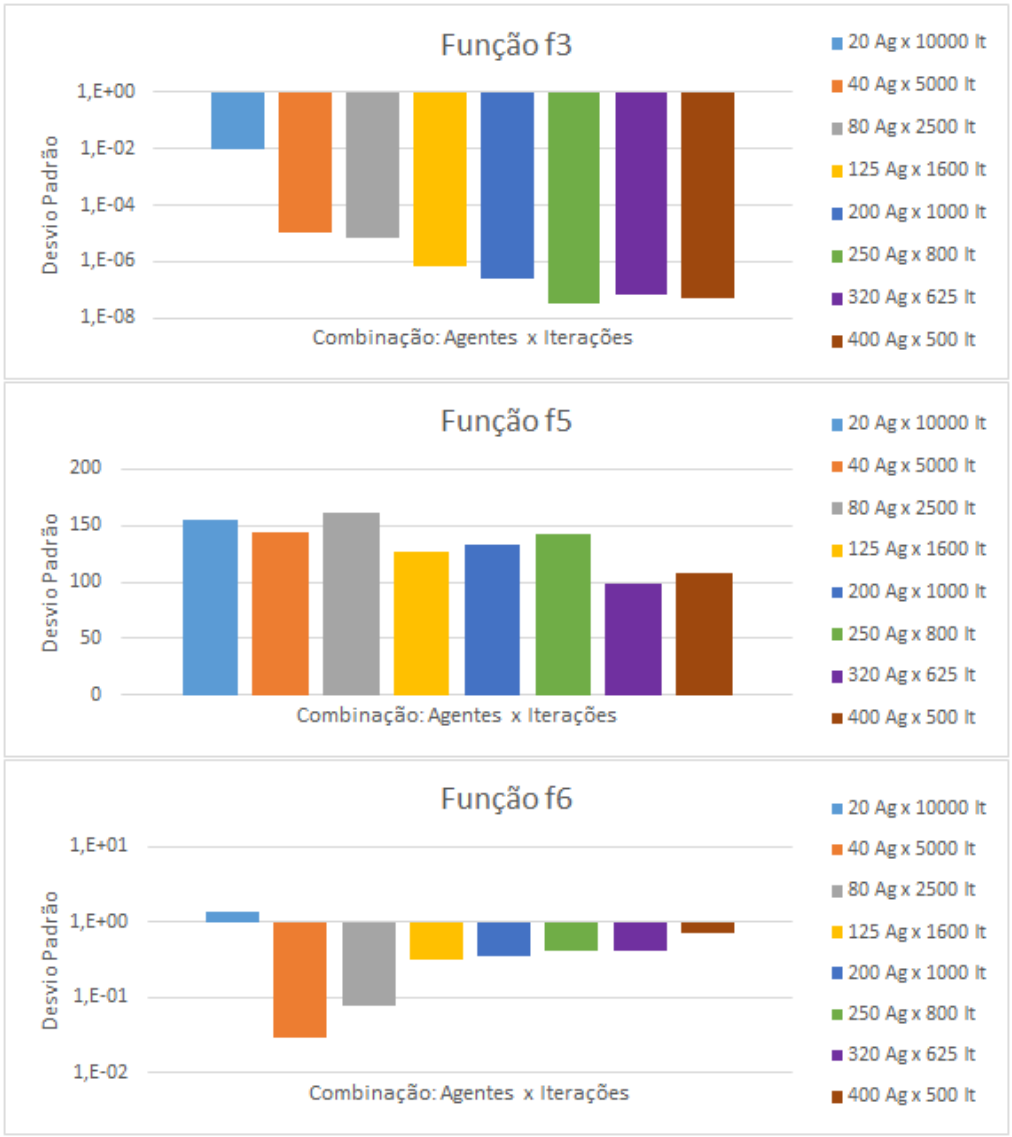
Figura 5.2 – Média do ótimo das funções considerando a variação do número de agentes de pesquisa e do número de iterações



Verifica-se que na Função f_3, o aumento do número de agentes de pesquisa promove melhorias no resultado, de forma que os menores valores de ótimo correspondem às três últimas combinações. Na função f_5, bons resultados são visíveis a partir de 80 agentes de pesquisa enquanto na função f_6 os melhores resultados (menores valores mínimos) encontram-se nas combinações intermediárias.

Na Figura 5.3 são apresentados os gráficos dos resultados obtidos em termos de desvio padrão para cada combinação de número de agentes e de iterações nas três funções analisadas. Para os gráficos das funções f_3 e f_6 utilizou-se a escala logarítmica.

Figura 5.3 – Desvio padrão das funções considerando a variação do número de agentes de pesquisa e do número de iterações



Verifica-se que nas funções f_3 e f_5 o aumento do número de agentes de pesquisa e consequente redução do número de iterações promove uma diminuição no valor do desvio padrão. Todavia, na função f_6, análises com um número elevado de agentes de pesquisa aumentaram o desvio padrão. Desta forma, combinações intermediárias dão ao algoritmo maior estabilidade, tendendo à produção de bons resultados em mais tipos de problemas distintos.

Na Figura 5.4 são apresentados os gráficos dos resultados obtidos em termos de tempo computacional para cada combinação de número de agentes e de iterações nas três funções analisadas.

Figura 5.4 – Tempo Computacional para as funções considerando a variação do número de agentes de pesquisa e do número de iterações

Verifica-se que a alteração do número de agentes de pesquisa bem como a consequente variação do número de iterações não provoca mudanças significativas de tempo computacional de operação do algoritmo CIOA.

Levando-se em conta todos os resultados apresentados nesta subseção, pode-se afirmar que o algoritmo CIOA apresenta ligeira sensibilidade a variações de número de agentes de pesquisa e número de iterações, especialmente quando avalia-se a média e o desvio padrão do valor

ótimo produzido. Comparando-se este comportamento nas três funções distintas na qual a análise foi realizada, pode-se dizer que, para o caso em questão, onde foram consideradas um número total de 200000 avaliações, as melhores configurações vão da combinação com 80 agentes de pesquisa e 2500 iterações até a combinação composta de 320 agentes de pesquisa e 625 iterações. Nesta faixa, as variações de resultado assumem valores de menor intensidade.

5.2.2 Análise de sensibilidade considerando a variação do parâmetro θ

Conforme já foi mencionado no capítulo 4, o ângulo θ é um dos únicos parâmetros próprios do algoritmo CIOA que precisa ser definido pelo operador. Nesta análise, estuda-se o comportamento do algoritmo CIOA mediante a alteração do ângulo θ, para isso, o algoritmo foi testado nas três funções *benchmark* já mencionadas, mediante 11 diferentes valores de θ, apresentados na Tabela 5.10. Nesta tabela também são apresentadas informações referentes ao valor de cada ângulo testado. Procurou-se realizar os testes em ângulos que correspondam a diferentes quadrantes da circunferência trigonométrica, alguns valores foram propositalmente selecionados devido ao fato de serem divisores $360°$, esta escolha é feita pois o algoritmo CIOA tende a apresentar melhor desempenho em ângulo não-divisores de $360°$. Durante esta análise de sensibilidade, fixou-se $Glob_{It} = 0{,}85$ e foram utilizadas 800 iterações de 250 agentes de pesquisa.

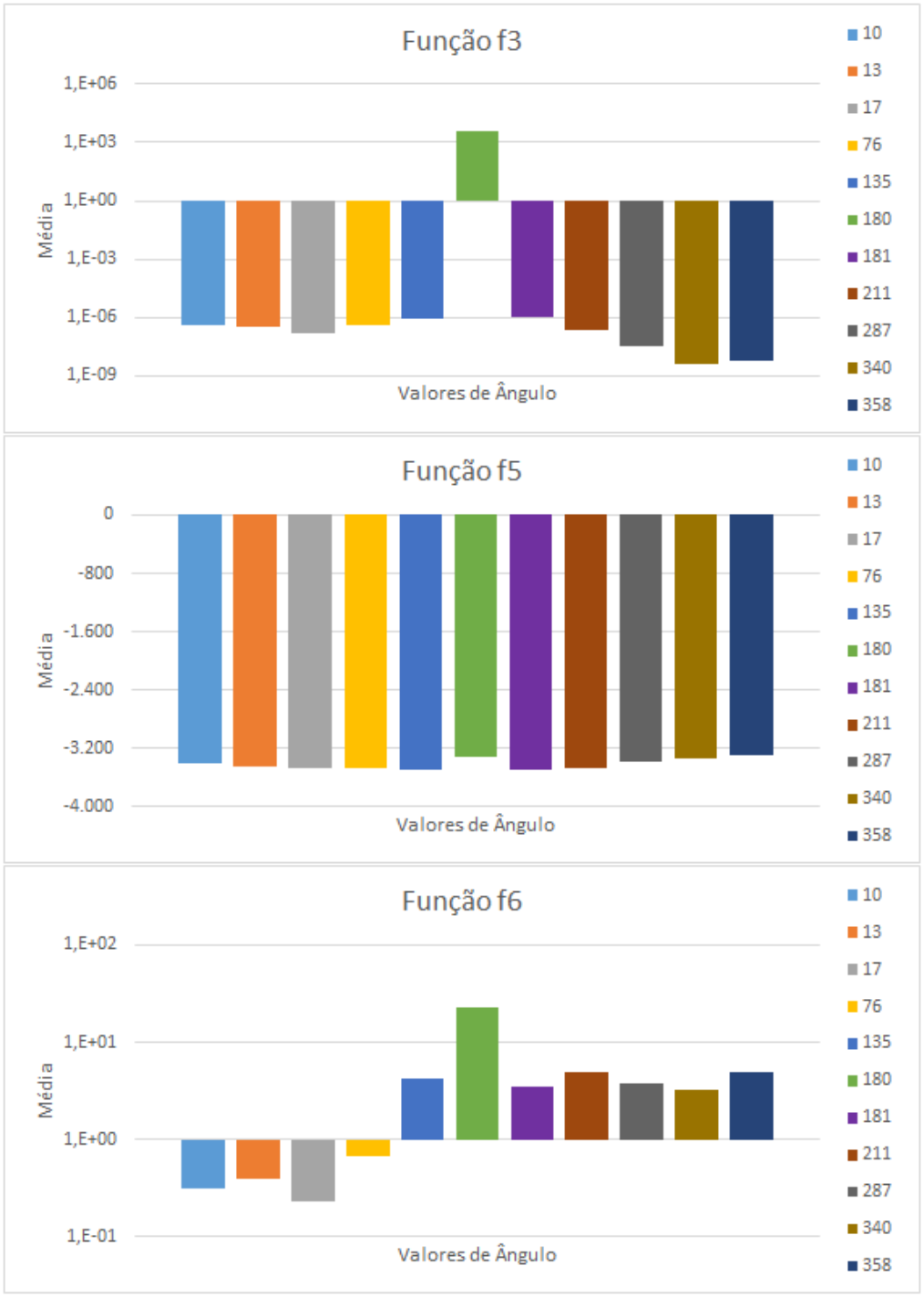
Tabela 5.10 – Valores do ângulo θ

Valor de θ	Informações
$10°$	Ângulo do 1º quadrante e divisor de $360°$
$13°$	Ângulo do 1º quadrante e não-divisor de $360°$
$17°$	Ângulo do 1º quadrante e não-divisor de $360°$
$76°$	Ângulo do 1º quadrante e não-divisor de $360°$
$135°$	Ângulo do 2º quadrante e não-divisor de $360°$
$180°$	Ângulo do 2º quadrante e divisor de $360°$
$181°$	Ângulo do 3º quadrante e não-divisor de $360°$
$211°$	Ângulo do 3º quadrante e não-divisor de $360°$
$287°$	Ângulo do 4º quadrante e não-divisor de $360°$
$340°$	Ângulo do 4º quadrante e não-divisor de $360°$
$358°$	Ângulo do 4º quadrante e não-divisor de $360°$

Os resultados obtidos em cada uma das três funções *benchmark* em termos de média do valor ótimo global são apresentados na Figura 5.5. Para melhor visualização, os gráficos das funções f_3 e f_6 estão em escala logarítmica, onde a origem das barras localiza-se no valor de 1. Verifica-

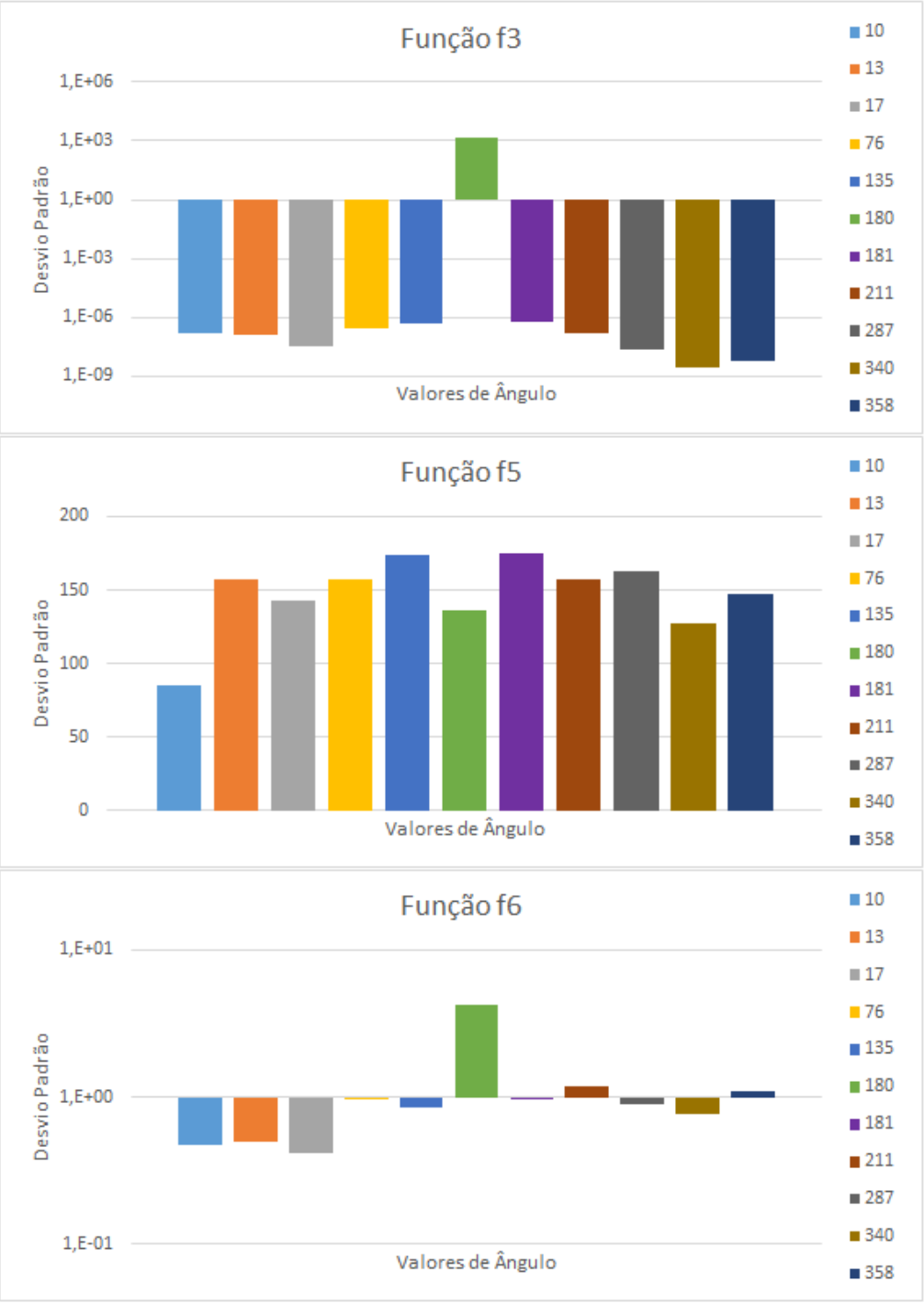
se que, nas funções f_3 e f_5, não há grande sensibilidade do algoritmo mediante a variação do ângulo θ. Todavia, o valor de 180° produz péssimos resultados na função f_3. Este fato referente ao ângulo de 180° se repete na função f_6, onde valores de ângulo fora do primeiro quadrante foram responsáveis por produzir médias elevadas, distantes do valor mínimo exato da função. De modo geral, o ângulo de 17° competiu uma performance estável ao algoritmo, com bons desempenhos nas três funções analisadas.

Figura 5.5 – Média das funções considerando a variação do ângulo θ



Na Figura 5.6 apresentam-se os resultados em termos de desvio padrão mediante a variação do ângulo θ. Nas funções f_3 e f_6 os gráficos são apresentados em escala logarítmica, com origem das barras no valor de 1.

Figura 5.6 – Desvio padrão das funções considerando a variação do ângulo θ

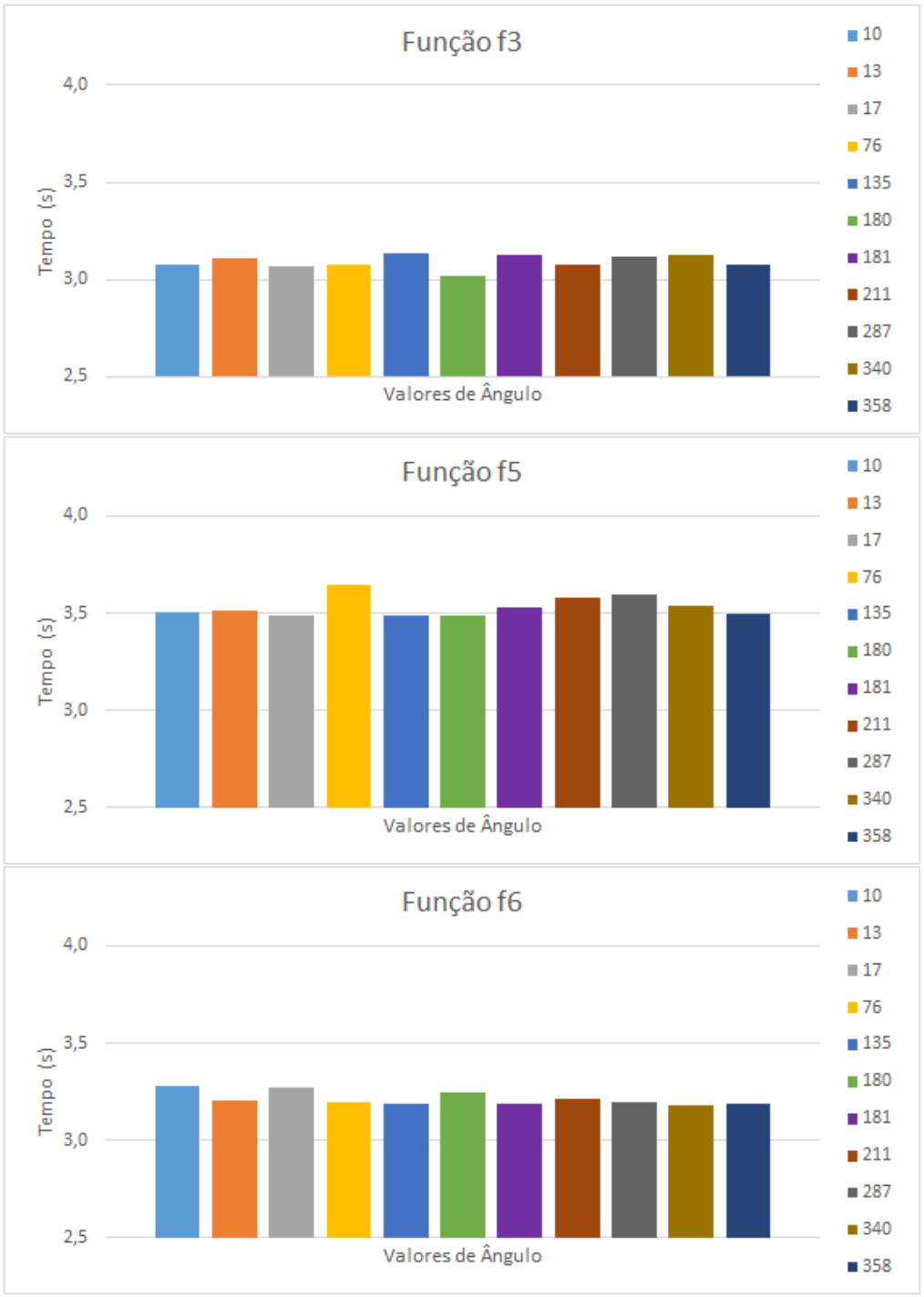


Verifica-se que a influência da escolha do ângulo no desvio padrão é relativamente baixa na função f_3, com exceção do ângulo de 180°, no qual há um aumento considerável de desvio

padrão. Na função f_6 os menores valores de desvio padrão são obtidos por valores de θ relativamente baixos, pertencentes ao primeiro quadrante da circunferência trigonométrica.

Na Figura 5.7 são apresentados os resultados em termos de tempo computacional considerando a variação do ângulo θ.

Figura 5.7 – Tempo Computacional para as funções considerando a variação do ângulo θ



É possível notar na Figura 5.7 que o valor selecionado para o ângulo θ causa pouca influência no tempo computacional de operação do CIOA. As ligeiras oscilações de tempo ilustradas na Figura 5.7 podem se dever às características randômicas do algoritmo.

Com base nos resultados apresentados nesta subseção, pode-se afirmar que os valores de ângulo θ sugeridos no Capítulo 4, durante a apresentação da formulação do CIOA, são justamente aqueles que conferem maior estabilidade ao algoritmo, ou seja, produzem bons resultados para um amplo tipo de problemas.

O valor de $\theta = 17°$, utilizado constantemente neste trabalho conferiu um dos desempenhos mais estáveis ao algoritmo.

5.2.3 Análise de sensibilidade considerando a variação do parâmetro $Glob_{It}$

O parâmetro $Glob_{It}$, apresentado pela primeira vez neste trabalho no Capítulo 4, define a proporção de iterações do algoritmo CIOA dedicadas às pesquisas global e local. Quanto maior o valor de $Glob_{It}$, menor será o número de iterações dedicadas exclusivamente à pesquisa local. Na seção 4.5 do presente trabalho, salientou-se que o algoritmo apresenta um melhor desempenho com valores de $Glob_{It}$ entre $0{,}75$ e $0{,}95$.

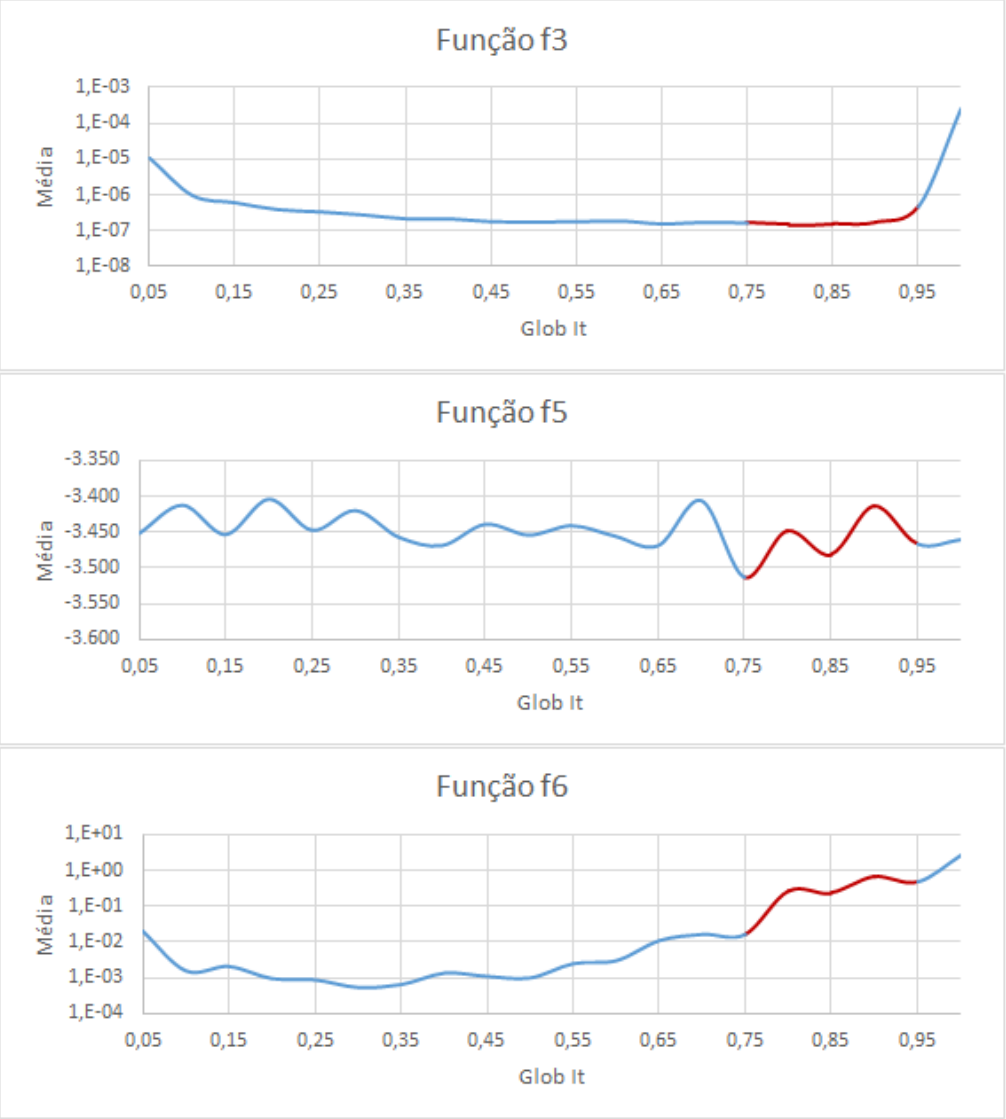
Na presente subseção, são apresentados resultados da análise do comportamento do CIOA mediante a variação do parâmetro $Glob_{It}$ nas três funções *benchmark* selecionadas para as análises de sensibilidade. Para isso, foram realizados testes variando-se o parâmetro $Glob_{It}$ de $0{,}05$ a $1{,}00$. Durante as análises, fixou-se $\theta = 17°$, foram consideradas 800 iterações e 250 agentes de pesquisa.

Na Figura 5.8 são apresentados os resultados obtidos em cada uma das três funções *benchmark* em termos de média do valor ótimo global mediante a variação de $Glob_{It}$, destaca-se no gráfico a faixa de valores sugerida para este parâmetro. Para uma melhor visualização, foi utilizada a escala logarítmica nos gráficos de f_3 e f_5.

Verifica-se que na função f_3 o algoritmo tem um mal desempenho para valores muito baixos ou muito altos de $Glob_{It}$ enquanto o melhor desempenho, ou seja, aquele que gera menores valores para a média do ótimo global, ocorre justamente na faixa de valores destacada. Na função f_5 não é possível apontar um padrão claro de influência entre o valor de $Glob_{It}$ e o desempenho do algoritmo, de modo que as oscilações do valor médio para o ótimo da função

podem ser oriundas das características randômicas do CIOA. Na função f_6 ocorre o oposto do que foi observado em f_3, isto é, com exceção de valores extremos, o melhor desempenho do algoritmo é obtido com valores relativamente baixos de $Glob_{It}$.

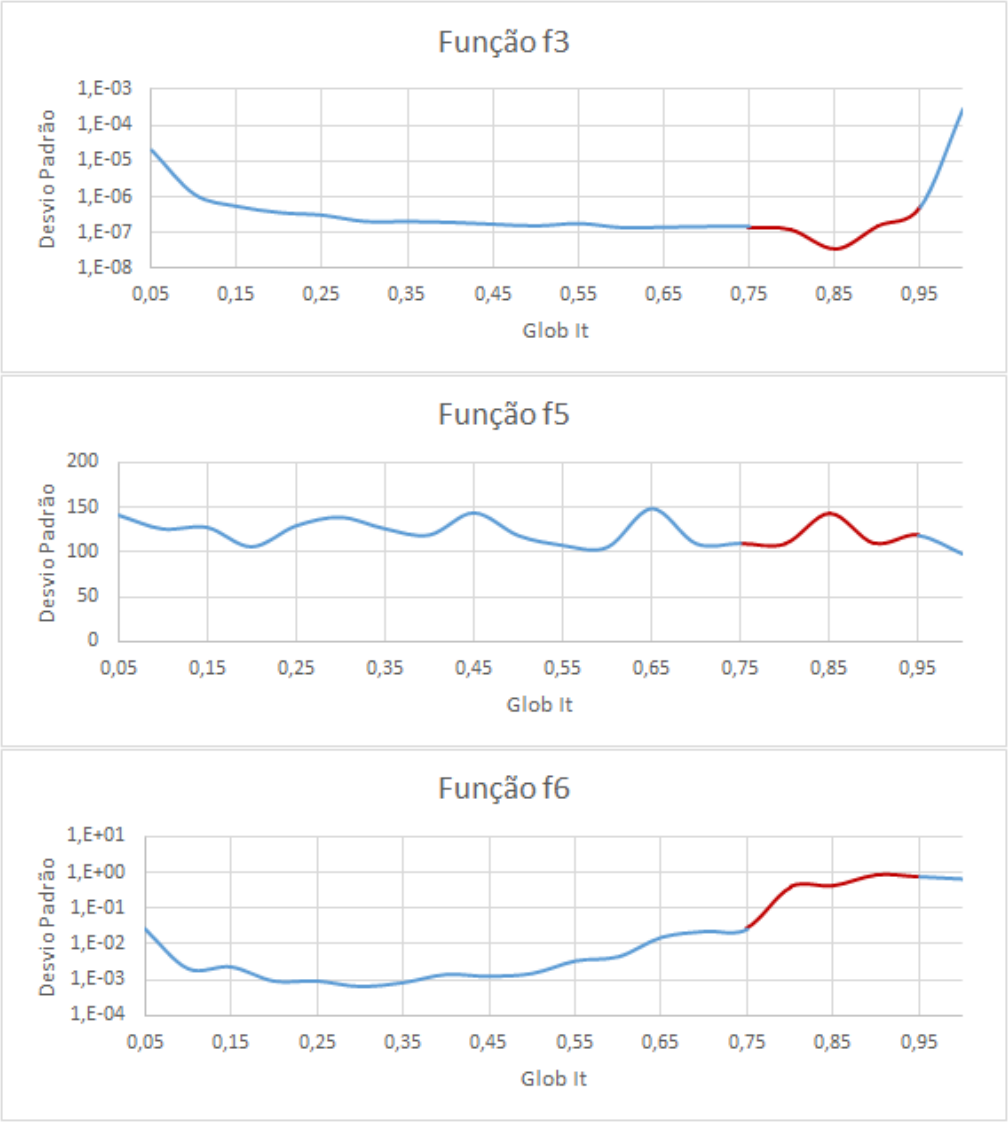
Figura 5.8 – Média das funções considerando a variação do parâmetro $Glob_{It}$



Na Figura 5.9 apresentam-se os resultados em termos de desvio padrão mediante a variação do parâmetro $Glob_{It}$. Para uma melhor visualização, nas funções f_3 e f_6, os gráficos são apresentados em escala logarítmica. De modo geral, o comportamento do algoritmo em relação ao desvio padrão foi similar ao analisado na média das funções. Na função f_3 os menores valores de desvio padrão são obtidos na faixa de valores recomendados para $Glob_{It}$, que vai de 0,75 a 0,95. Na função f_5 não se pode mencionar uma influência clara entre os valores de $Glob_{It}$ e os valores de desvio padrão, as oscilações observadas podem ser resultado das

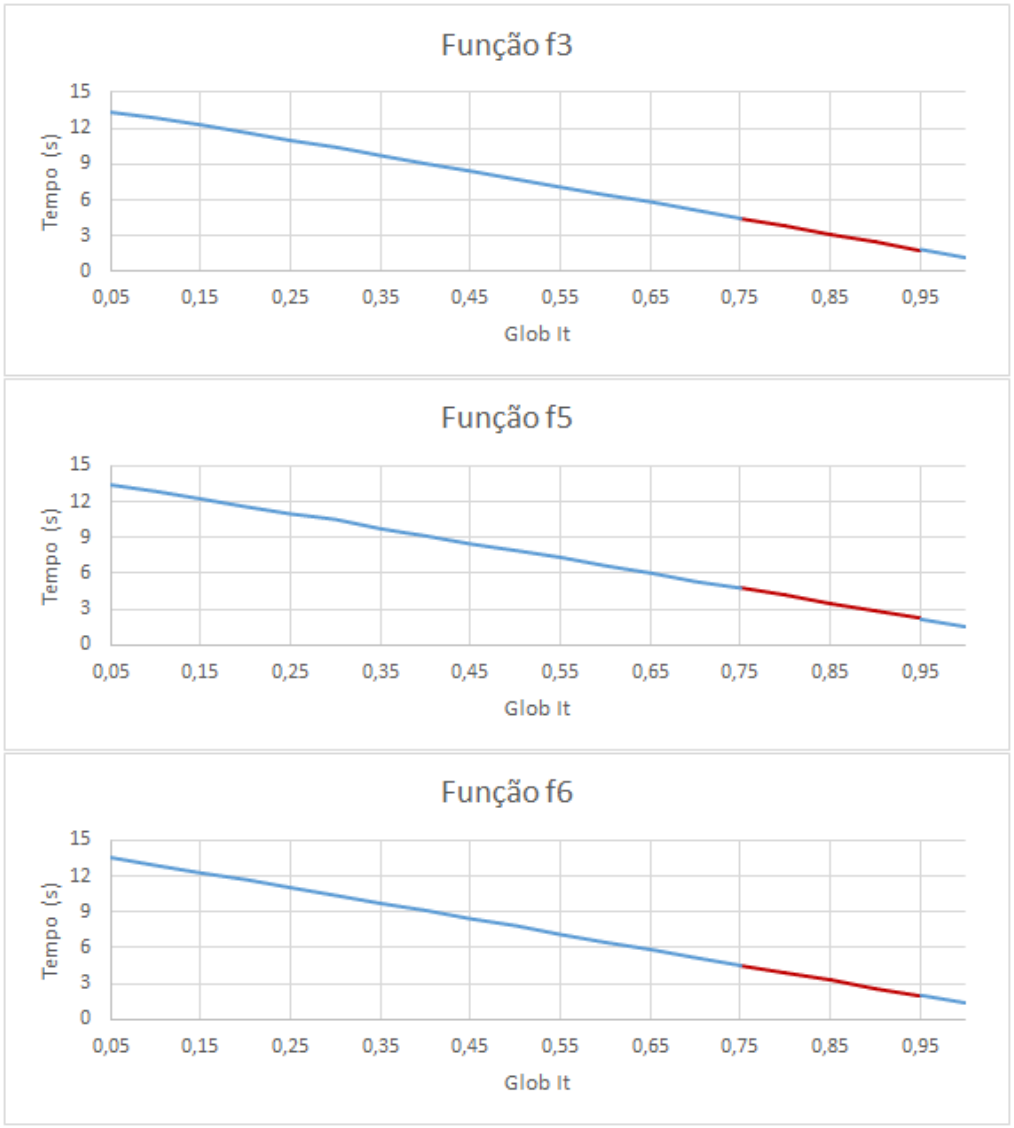
características randômicas do CIOA, presentes também em outros algoritmos meta-heurísticos. Na função f_6, o melhor desempenho é atingido em valores relativamente baixos de $Glob_{It}$, especificamente entre 0,20 e 0,35.

Figura 5.9 – Desvio padrão das funções considerando a variação do parâmetro $Glob_{It}$



Na Figura 5.10 são apresentados os resultados da influência do parâmetro $Glob_{It}$ sobre o tempo computacional do algoritmo CIOA. Observa-se que, em todas as funções analisadas, quando menor o valor de $Glob_{It}$, maior será o tempo computacional necessário para o algoritmo solucionar o problema. Desta forma, em termos de custo computacional, torna-se altamente atrativo que o valor a ser definido pelo usuário para o parâmetro $Glob_{It}$ seja o mais próximo possível de 1.

Figura 5.10 – Tempo Computacional para as funções considerando a variação do parâmetro $Glob_{It}$



Considerando-se todos os resultados apresentados nesta subseção, seja em termos de média, desvio padrão e tempo computacional, verifica-se que a faixa de valores $0{,}75 \leq Glob_{It} \leq 0{,}95$ é a que promove um maior equilíbrio entre resultados estatísticos e tempo computacional, tornando o algoritmo capaz de produzir bons resultados em diferentes tipos de problemas. Salienta-se ainda que, o parâmetro $Glob_{It}$ apresenta influência relevante ao tempo computacional de operação do algoritmo, onde há uma relação direta, constante e explícita entre o aumento do valor de $Glob_{It}$ e a diminuição do tempo computacional.

5.2.4 Análise de sensibilidade considerando a variação do coeficiente de atualização do vetor de raios

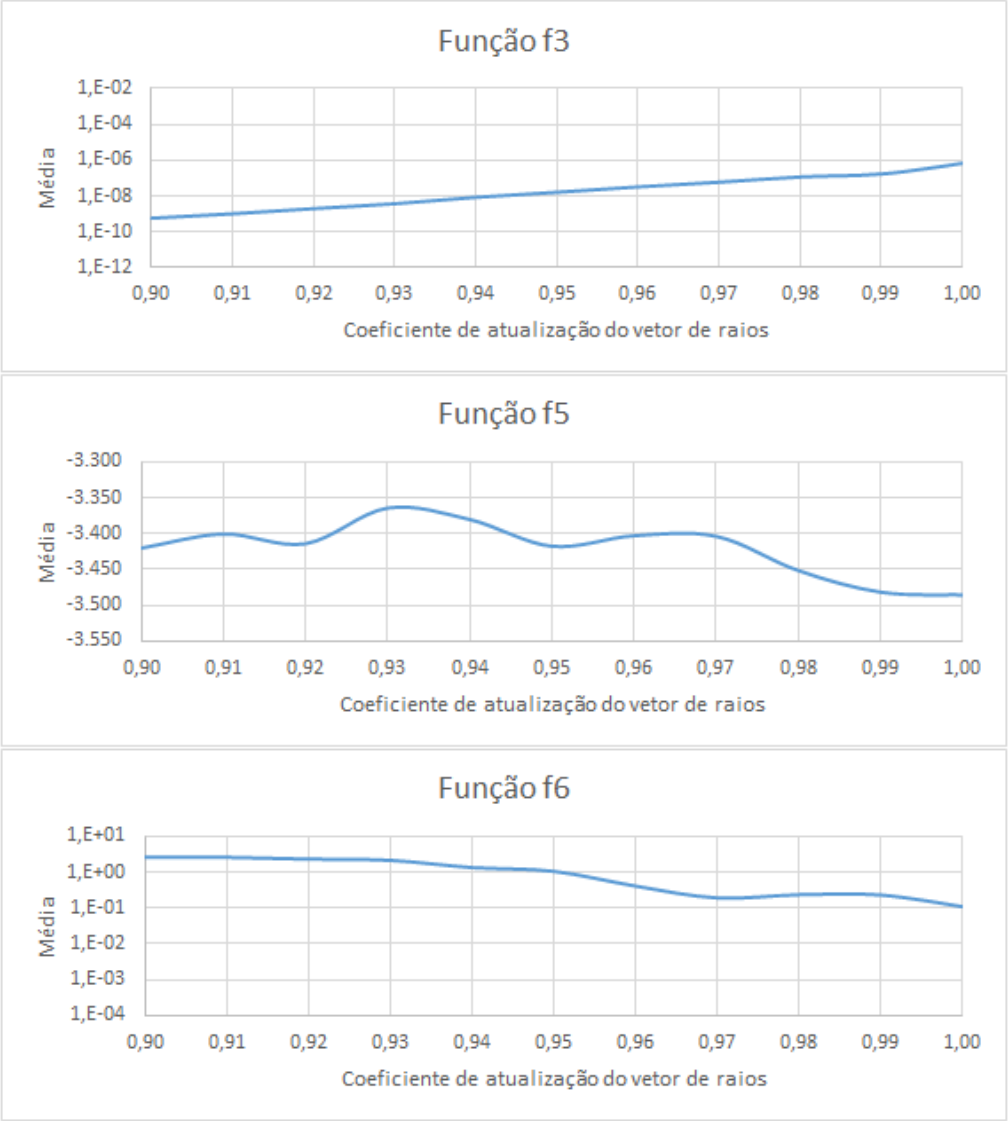
Na Equação 4.10 da seção 4.3 é proposta uma redução do vetor de raios do algoritmo CIOA, onde utiliza-se um coeficiente de valor fixo igual a $0{,}99$. Embora exista a possibilidade deste valor ser alterado no código computacional do CIOA, recomenda-se fortemente que o valor indicado seja mantido fixo, pois este promove estabilidade ao algoritmo, tornando-o capaz de operar com eficiência em diferentes tipos de problemas de otimização.

Na análise desta subseção, são apresentados os resultados da influência do valor do coeficiente de atualização do vetor de raios no comportamento do CIOA. Para isto, testes foram realizados variando-se o coeficiente entre valores que vão de 0,90 até 1. Durante esta análise, os parâmetros do algoritmo foram fixados em $\theta = 17°$ e $Glob_{It} = 0{,}85$. 800 iterações e 250 agentes de pesquisa foram considerados. Os resultados obtidos são apresentados na sequência.

Na Figura 5.11 apresenta-se a variação da média do ótimo global das três funções *benchmark* selecionadas para a análise de sensibilidade mediante variações do valor do coeficiente de atualização do vetor de raios. Para uma melhor visualização dos resultados, os gráficos das funções f_3 e f_5 estão em escala logarítmica.

É possível observar um comportamento distinto em cada função analisada. Na função f_3, unimodal e com ótimo exato igual a zero, os melhores resultados são obtidos quando o coeficiente de atualização de raios apresenta valores mais baixos; já nas funções f_5 e f_6, ambas multimodais, o comportamento inverso ocorre, ou seja, os valores de ótimo global são reduzidos conforme o coeficiente de atualização de raios se aproxima do valor de 1. Desta forma, o valor indicado de 0,99 dá ao algoritmo alta eficiência em problemas similares aos de f_5 e f_6, e uma eficiência reduzida em problemas similares ao de f_3. Todavia, é importante ressaltar que no caso específico da função f_3, a variação do coeficiente de atualização de raios promove influência a partir da 7ª casa decimal de precisão, uma vez que o ótimo global desta função varia da faixa de 10^{-10} a 10^{-7}. Em outras palavras, ainda que um coeficiente de 0,99 atribua ao CIOA um desempenho levemente inferior na função f_3, o resultado obtido apresenta uma alta precisão, correspondente à 6 casas decimais. A redução do coeficiente para 0,90 aumentaria a precisão do CIOA na função f_3 para 10 casas decimais, em contrapartida, seu desempenho nas funções multimodais cairia drasticamente, apresentando resultados visivelmente afastados do valor exato do ótimo global.

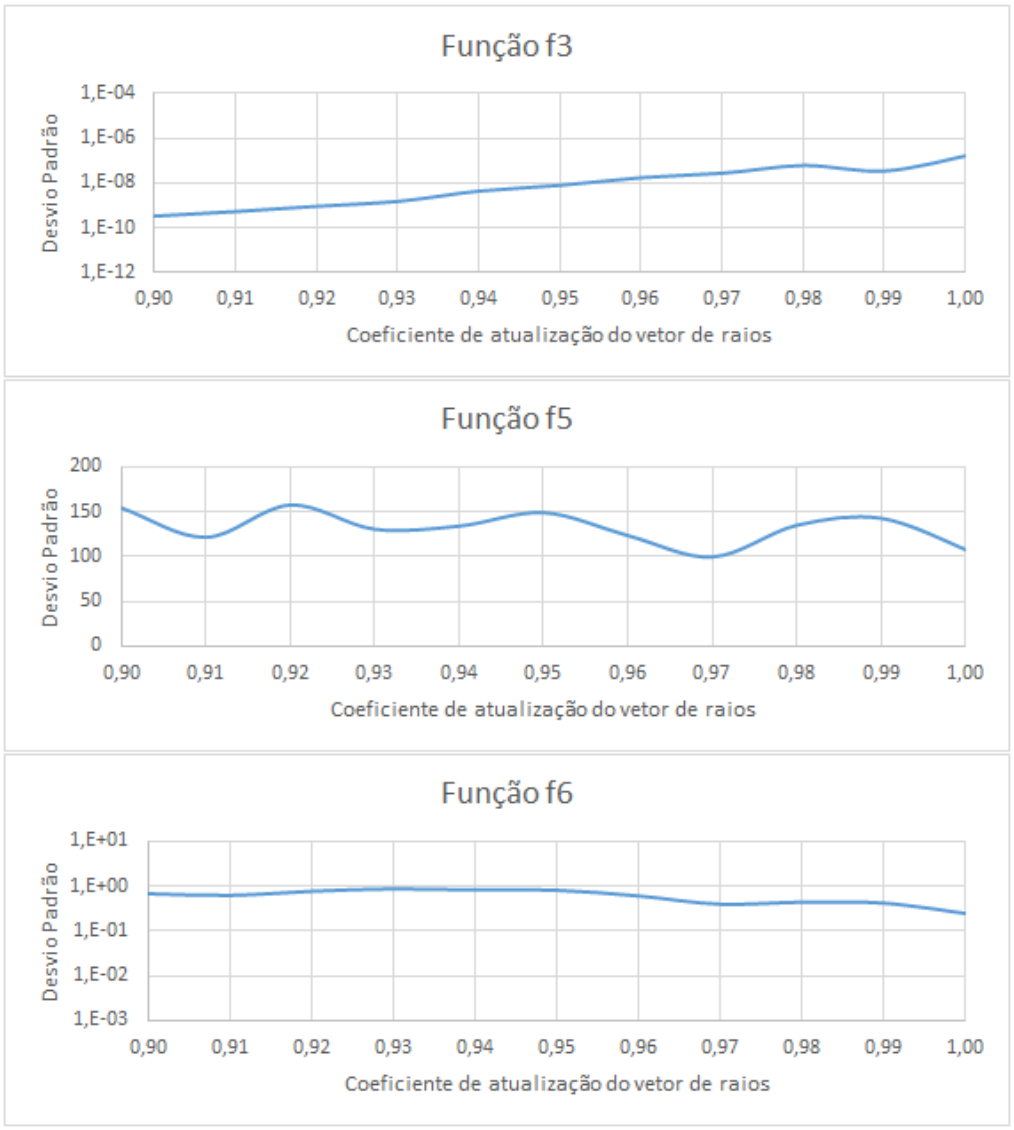
Figura 5.11 – Média das funções considerando a variação do coeficiente de atualização do vetor de raios



Na Figura 5.12 são apresentados os resultados para a variação do desvio padrão entre as simulações de cada uma das três funções *benchmark* utilizadas na análise de sensibilidade do CIOA quando são alterados os valores de coeficiente de atualização do vetor de raios. Para uma melhor visualização dos resultados, os gráficos das funções f_3 e f_5 são apresentados em escala logarítmica.

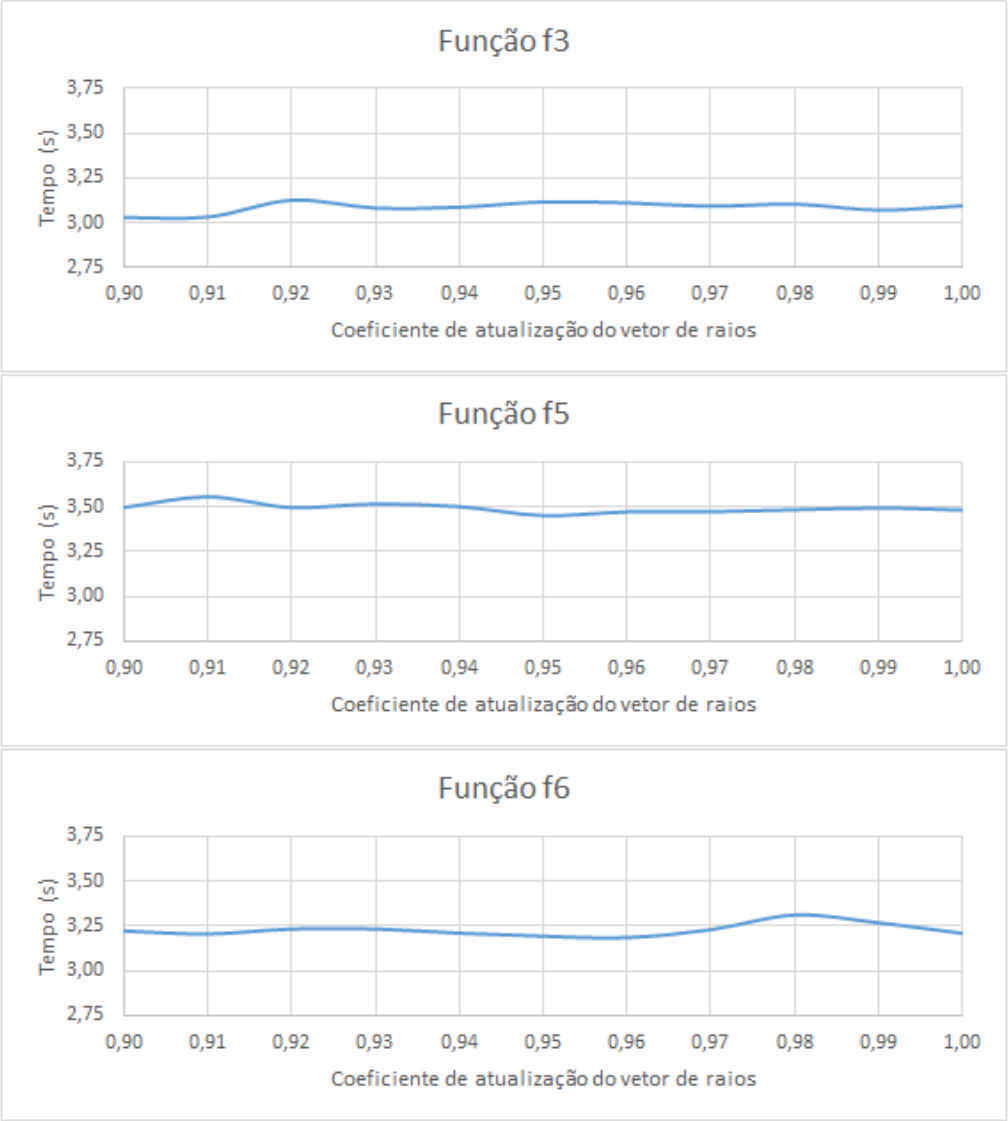
Observa-se no CIOA, para o desvio padrão, um comportamento similar ao ocorrido na avaliação da média das funções, na qual o valor de 0,99 para o coeficiente de atualização de raios promove uma estabilidade de desempenho em diferentes tipos de problemas.

Figura 5.12 – Desvio padrão das funções considerando a variação do coeficiente de atualização do vetor de raios



Na Figura 5.13 são apresentados os resultados em termos de tempo computacional para cada uma das três funções analisadas quando varia-se entre 0,90 a 1 o coeficiente de atualização do vetor de raios. Observa-se que não há significativa influência deste parâmetro no tempo computacional. As ligeiras oscilações apresentadas podem ser oriundas das características randômicas do CIOA.

Figura 5.13 – Tempo Computacional para as funções considerando a variação do coeficiente de atualização do vetor de raios



Levando-se em conta a média, o desvio padrão e o tempo computacional de operação, pode-se concluir que o valor de 0,99 para o coeficiente de atualização de raios é adequado. Uma vez que valores inferiores promovem um aumento de média e desvio padrão, especialmente nas funções f_5 e f_6 enquanto valores superiores a 0,99 promovem este aumento na função f_6.

5.2.5 Análise de sensibilidade considerando a variação do denominador de redução do limite de variáveis

Nesta subseção analisa-se a influência do valor do denominador da redução do limite de variáveis no comportamento do algoritmo CIOA. Para isso, foram utilizados valores de diferentes ordens de grandeza, que vão desde 10 até 100000. Durante esta análise, os parâmetros

do algoritmo foram fixados em $\theta = 17°$ e $Glob_{It} = 0{,}85$. 800 iterações e 250 agentes de pesquisa foram considerados.

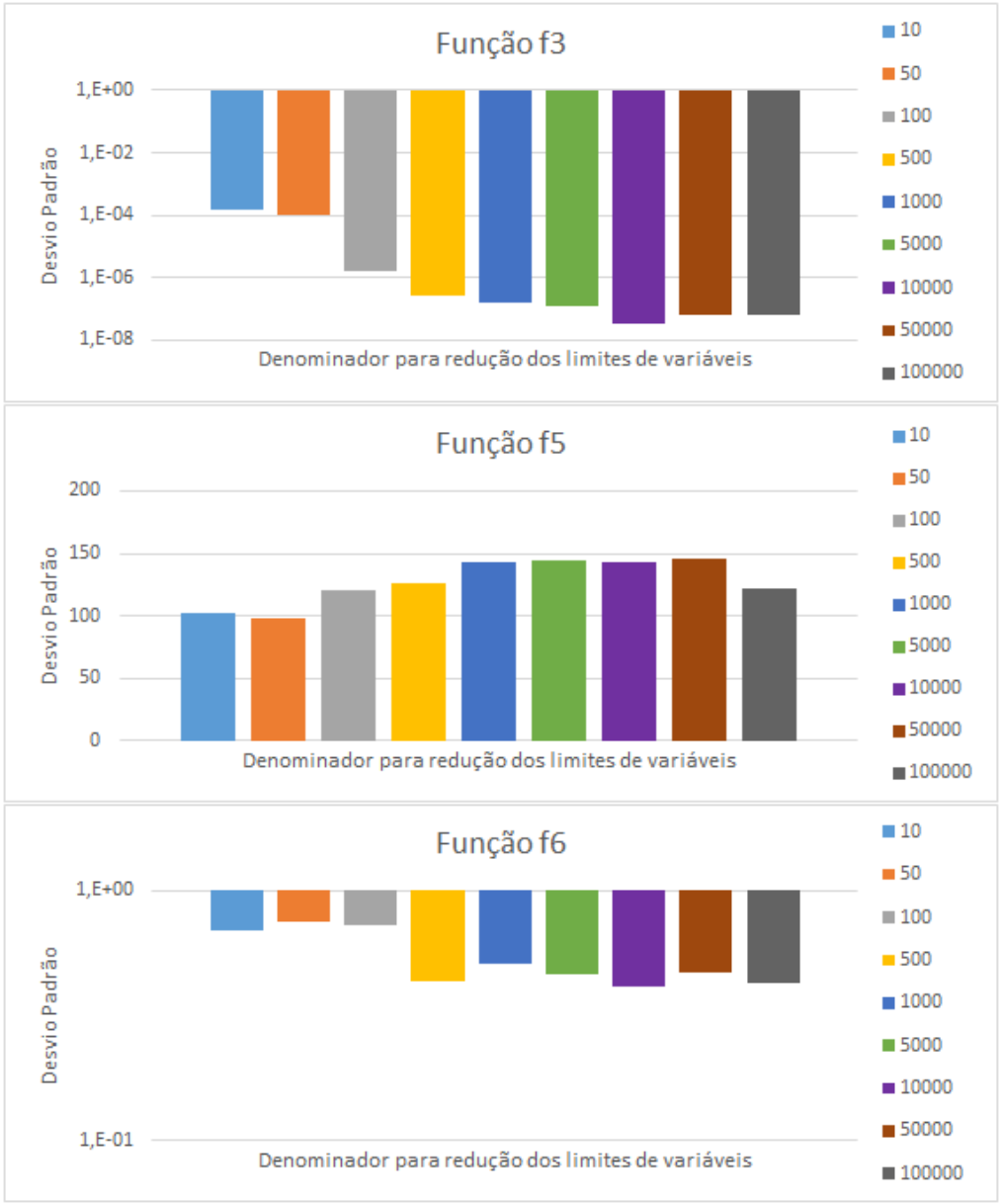
Na Figura 5.14 ilustra-se a média das funções analisadas diante da alteração deste parâmetro. Para uma melhor visualização, os gráficos das funções f_3 e f_6 estão em escala logarítmica.

Figura 5.14 - Média das funções considerando a variação do denominador de redução do limite de variáveis

Verifica-se que as funções assumem os menores valores para a média do ótimo global quando o denominador de redução do limite de variáveis assume valores relativamente altos. De modo geral, a menor média ocorre quando o denominador é fixado com o valor de 10000.

Na Figura 5.15 são apresentados os resultados para a variação do desvio padrão entre as simulações de cada uma das três funções *benchmark* utilizadas na análise de sensibilidade quando são alterados os valores do denominador de redução do limite de variáveis. Nas funções f_3 e f_6 utiliza-se a escala logarítmica de modo a facilitar a visualização dos resultados.

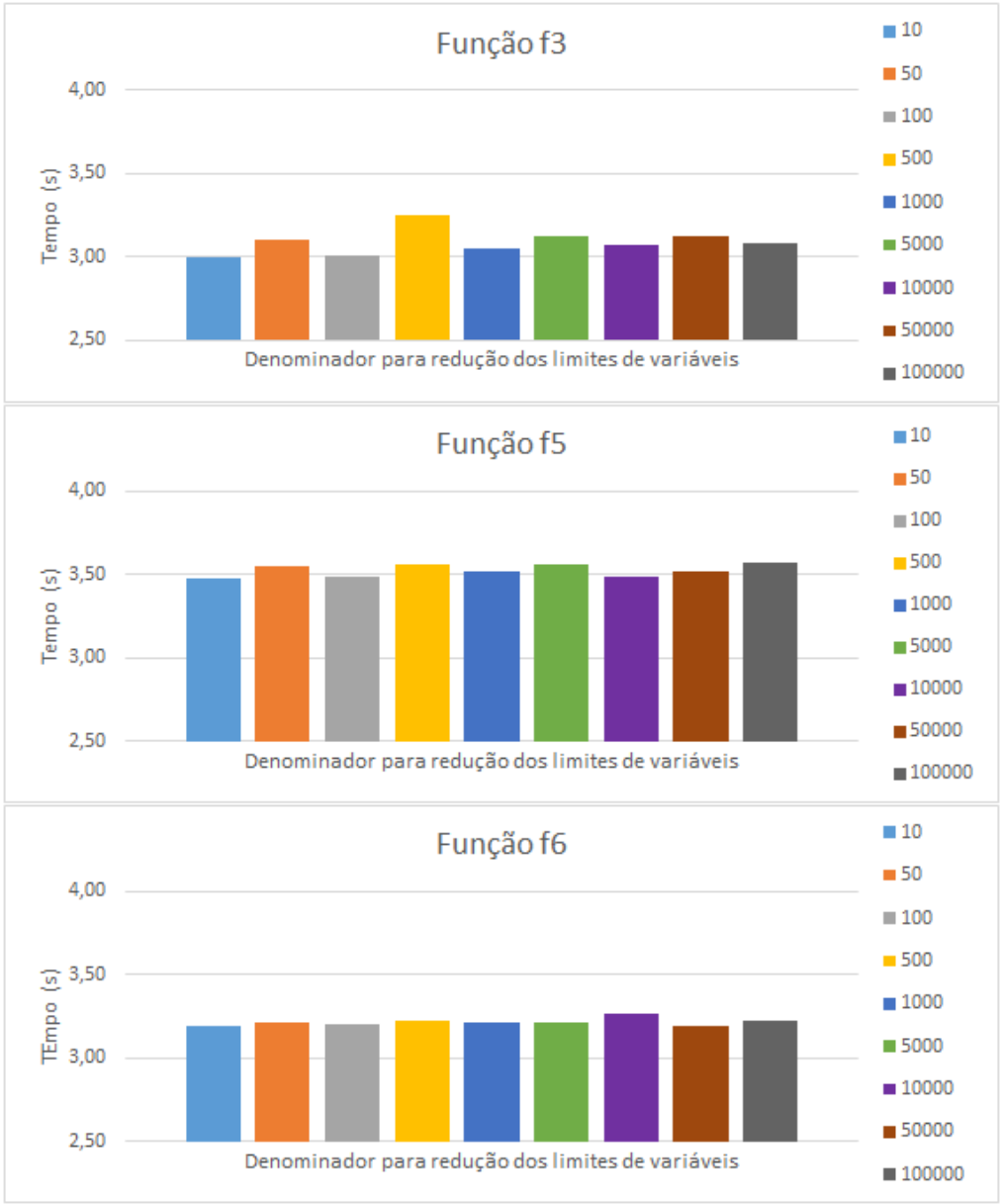
Figura 5.15 – Desvio padrão das funções considerando a variação do denominador de redução do limite de variáveis



Verifica-se que tanto na função f_3 como na função f_6, os menores valores de desvio padrão são obtidos quando o denominador que reduz o limite das variáveis de projeto é fixado com o valor de 10000. Na função f_5 ocorre o comportamento oposto, onde os menores valores de desvio padrão ocorrem para valores pequenos para este denominador.

Na Figura 5.16 avalia-se a influência que o valor do denominador de redução para o limite de variáveis exerce sobre o tempo computacional de operação do algoritmo.

Figura 5.16 – Tempo Computacional para as funções considerando a variação do denominador de redução do limite de variáveis



Observa-se que não há uma relação direta entre o denominador e o tempo computacional de operação do algoritmo.

Os resultados desta subseção corroboram o valor de 10000, no qual o denominador para a redução dos limites de variáveis foi fixado.

6. APLICAÇÃO DO *CIRCLE-INSPIRED OPTIMIZATION ALGORITHM* EM PROBLEMAS CLÁSSICOS DE ENGENHARIA ESTRUTURAL

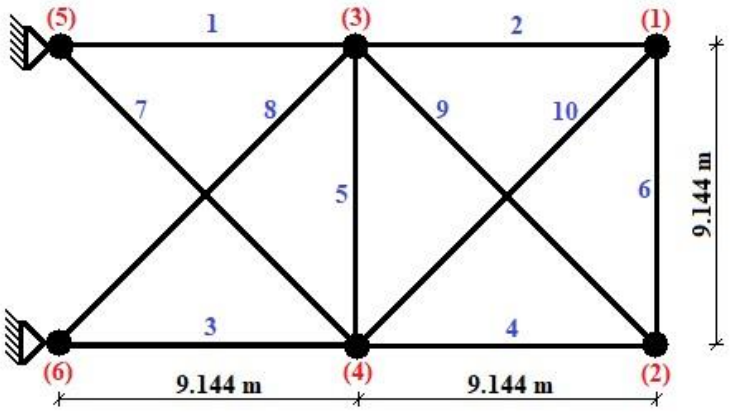
Neste capítulo aplica-se o *Circle-Inspired Optimization Algorithm* em problemas de otimização clássicos da engenharia que consistem na otimização paramétrica e/ou de forma de treliças sujeitas a diferentes ações e restrições. Estes problemas já foram amplamente estudados por diversos pesquisadores, especialmente com a finalidade de avaliar a eficiência de algoritmos meta-heurísticos quando aplicados à problemas de otimização estrutural.

Os resultados obtidos pelo CIOA foram gerados utilizando-se os parâmetros $Glob_{It} = 0{,}85$ e $\theta = 17°$ em todas as simulações. Estes valores conferem ao algoritmo uma maior estabilidade na busca de boas soluções, conforme foi observado na análise de sensibilidade apresentada na Seção 5.2 do presente trabalho. Novamente, o desempenho do *Circle-Inspired Optimization Algorithm* foi comparado com o desempenho dos cinco algoritmos meta-heurísticos apresentados na Seção 3.3 e já utilizados para a validação do CIOA no Capítulo 5.

6.1 OTIMIZAÇÃO PARAMÉTRICA DE TRELIÇA PLANA DE DEZ BARRAS SUJEITA A RESTRIÇÕES DE TENSÃO E DESLOCAMENTO

Neste problema, é realizada a otimização paramétrica de uma treliça plana de dez barras, mostrada na Figura 6.1. Assim, as variáveis de projeto são contínuas e representam as seções transversais de cada uma das dez barras que compõem a estrutura.

Figura 6.1 – Treliça plana de 10 barras sujeita a restrições de tensão e deslocamento



Fonte: Baseado em BORGES, 2013

A treliça é constituída de alumínio, tem módulo de Young $E = 68{,}95\ GPa$, massa específica $\rho = 2767{,}99\ kg/m^3$ e está sujeita a cargas verticais de $-444{,}82\ kN$ aplicadas nos nós 2 e 4. As restrições de tensão são $\pm 517{,}11\ MPa$ para o membro 9 e $\pm 172{,}37\ MPa$ para os demais membros. As restrições de deslocamento são $\pm 5{,}08\ cm$ na direção y para os nós 1, 2, 3 e 4 (nós livres). O intervalo das variáveis é de $0{,}645\ cm^2 \leq x_i \leq 200\ cm^2$.

Esse problema já foi estudado por outros autores. Lee e Geem (2004) utilizaram o HS para a otimização da treliça, Borges (2013) otimizou a estrutura utilizando o HS e o FA, com a finalidade de comparar o desempenho de ambos os algoritmos.

Neste trabalho, foram efetuadas dez simulações através de cada algoritmo. Nas análises com o CIOA, WOA, SGA e PSO foram utilizadas 800 iterações e 250 agentes de pesquisa; no algoritmo FA foram empregadas 5000 iterações e 40 agentes de pesquisa; a resolução com o HS foi feita utilizando-se de 200000 iterações. Desta forma, todos os algoritmos minimizaram a massa da treliça após um número total de 200000 avaliações. Os resultados obtidos para a massa mínima (kg) e para a seção transversal (cm²) de cada elemento da treliça são apresentados na Tabela 6.1, onde são ainda comparados com os valores que Borges (2013) obteve em seu trabalho utilizando o HS e o FA. Observa-se que o CIOA produziu resultados similares aos demais algoritmos com o qual foi comparado.

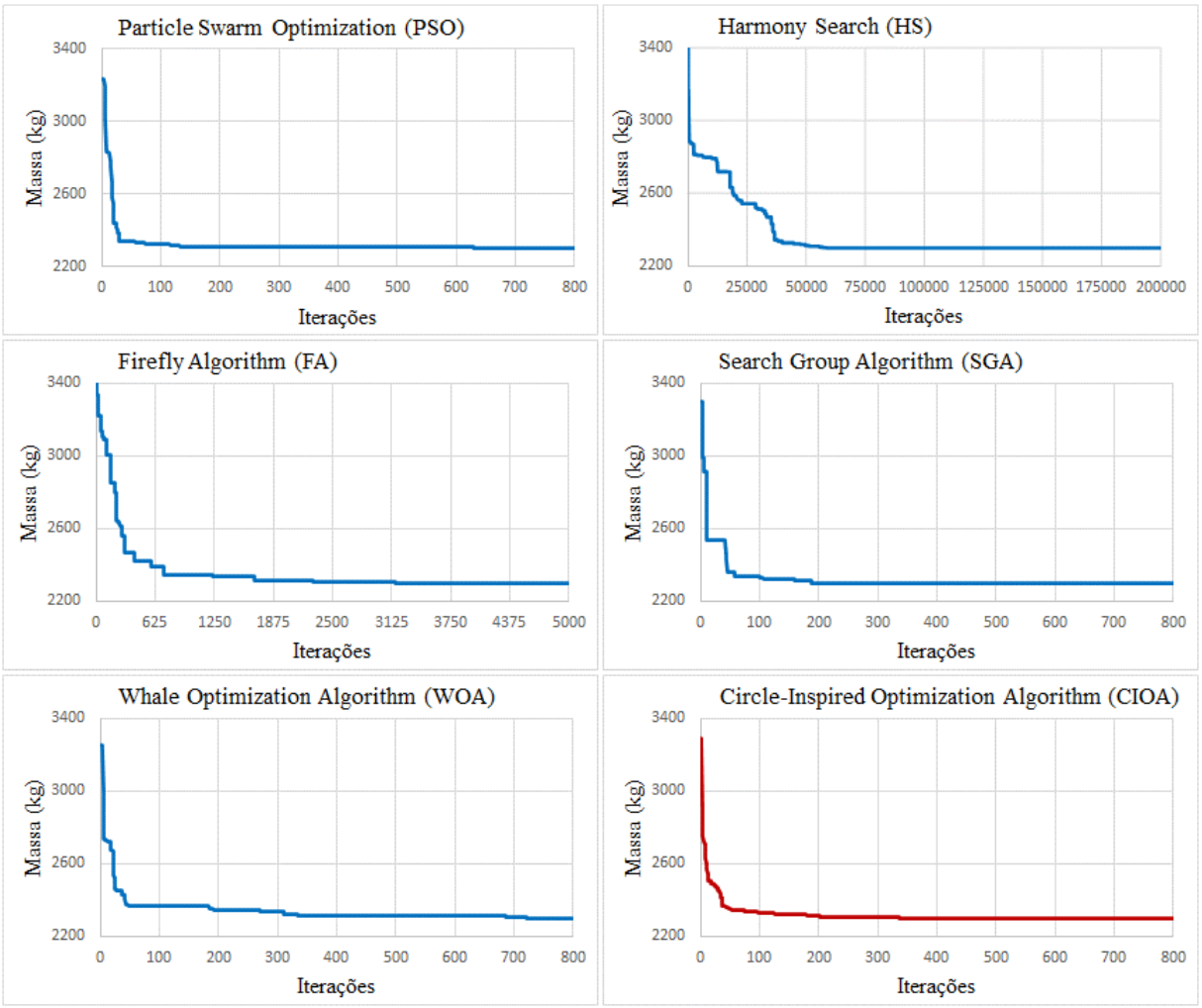
Tabela 6.1 – Projeto ótimo para a treliça plana de 10 barras sujeita a restrições de tensão e deslocamento

	Borges (2013)			Presente Trabalho				
Membro	HS	FA	PSO	HS	FA	SGA	WOA	CIOA
1	196,180	186,680	192,110	196,515	184,723	195,315	199,570	194,467
2	1,128	0,645	0,645	0,721	0,669	0,645	0,650	0,645
3	144,560	164,350	164,060	153,598	154,066	149,747	156,170	147,909
4	103,170	94,527	96,143	94,073	98,249	98,782	97,578	103,639
5	0,645	0,645	0,645	0,686	0,645	0,645	0,650	0,653
6	3,560	4,103	3,016	3,702	3,763	3,613	2,661	4,023
7	48,884	46,884	47,329	48,580	48,075	48,132	48,150	47,740
8	138,040	137,160	138,580	137,696	140,671	136,409	134,230	132,745
9	138,020	139,300	133,200	138,089	140,365	138,813	135,620	141,474
10	0,673	0,645	0,645	0,744	0,645	0,645	0,649	0,657
Massa (kg)	2302,60	2301,08	2300,29	2298,77	2299,36	**2295,54**	2297,99	2297,14

Os resultados obtidos no presente trabalho são compatíveis com estudos realizados por outros autores. Além dos valores obtidos por Borges (2013) apresentados na Tabela 6.1, Lee e Geem (2004) encontraram uma massa mínima de $2294{,}23\ kg$ utilizando o HS. Ressalta-se que resultados obtidos por um mesmo algoritmo podem sofrer variações em cada estudo devido as características randômicas, próprias dos algoritmos meta-heurísticos; devido ao número de avaliações totais utilizado por cada autor e à maneira como o problema foi implementado.

Na Figura 6.2 são apresentadas as curvas de convergência para a melhor execução de cada algoritmo. O CIOA apresentou rápida convergência, alcançando o resultado da massa ótima ainda antes da etapa de iterações onde todos os agentes dedicam-se exclusivamente à pesquisa local.

Figura 6.2 – Curvas de convergência na otimização de treliça plana de 10 barras sujeita a restrições de tensão e deslocamento



A Tabela 6.2 mostra os valores médios de massa mínima, valores de desvio padrão e de coeficiente de variação obtidos após dez execuções de cada algoritmo além do tempo computacional médio necessário para cada execução obtidos no presente trabalho. Embora não tenha superado o desempenho do algoritmo SGA, o *Circle-Inspired Optimization Algorithm* (CIOA) apresentou valores competitivos, superando grande parte dos demais algoritmos em todos os parâmetros avaliados.

Tabela 6.2 – Análise estatística e de tempo computacional para a treliça plana de 10 barras sujeita a restrições de tensão e deslocamento

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	2303,30	2300,65	2302,45	**2295,68**	2302,72	2298,36
Desvio padrão (kg)	1,55	1,52	1,99	**0,10**	3,16	0,65
Coef. de variação (%)	0,067	0,066	0,086	**0,005**	0,137	0,028
Tempo (s)	217,99	216,12	214,82	**185,01**	204,85	199,84

No Quadro 6.1 apresenta-se um ranking onde os algoritmos são classificados de acordo com a ordem crescente de seus resultados, uma vez que, para todos os parâmetros avaliados, o melhor resultado corresponde ao menor valor obtido.

Quadro 6.1 – Ranking para a treliça plana de 10 barras sujeita a restrições de tensão e deslocamento

Parâmetro	Ranking
Massa mínima	[SGA] < [**CIOA**] < [WOA] < [HS] < [FA] < [PSO]
Massa média	[SGA] < [**CIOA**] < [HS] < [FA] < [WOA] < [PSO]
Desvio padrão	[SGA] < [**CIOA**] < [HS] < [PSO] < [FA] < [WOA]
Coeficiente de variação	[SGA] < [**CIOA**] < [HS] < [PSO] < [FA] < [WOA]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [FA] < [HS] < [PSO]

Observa-se que o CIOA ocupou a segunda posição em todos os parâmetros analisados, superando o desempenho de quatro dos cinco algoritmos com o qual foi comparado.

É importante ressaltar que as restrições de tensão e deslocamento não foram violadas durante o processo de otimização, como pode ser visto nas tabelas 6.3 e 6.4, que apresentam respectivamente, as tensões e deslocamentos obtidas na configuração final após a otimização por cada algoritmo. Em negrito, destacam-se as restrições ativas, ou seja, aquelas que atingiram o limite estabelecido no problema de otimização.

Tabela 6.3 – Tensões (MPa) obtidas após a otimização da treliça plana de 10 barras sujeita a restrições de tensão e deslocamento

Membro	PSO	HS	FA	SGA	WOA	CIOA
1	46,914	45,890	48,791	46,148	45,154	46,359
2	-7,781	-7,798	-8,802	-9,107	-7,590	-10,189
3	-53,518	-57,127	-56,989	-58,628	-56,229	-59,343
4	-46,319	-47,344	-45,335	-45,090	-45,637	-42,983
5	172,330	169,260	171,280	173,337	169,456	-172,160
6	-1,664	-1,519	-1,565	-1,626	-1,854	-1,634
7	129,440	125,950	127,430	127,258	127,268	128,245
8	-46,580	-46,936	-45,889	-47,330	-48,078	-48,657
9	47,281	45,613	44,876	45,378	46,436	44,531
10	11,003	10,687	12,912	12,879	10,751	14,145

Tabela 6.4 - Deslocamentos (cm) na direção y obtidos após a otimização da treliça plana de 10 barras sujeita a restrições de tensão e deslocamento

Nó	PSO	HS	FA	SGA	WOA	CIOA
1	**-5,08**	**-5,08**	**-5,08**	**-5,08**	**-5,08**	**-5,08**
2	-5,06	-5,06	-5,06	-5,06	-5,06	-5,06
3	-1,86	-1,85	-1,86	-1,87	-1,87	-1,91
4	-4,14	-4,10	-4,14	-4,15	-4,12	-4,49

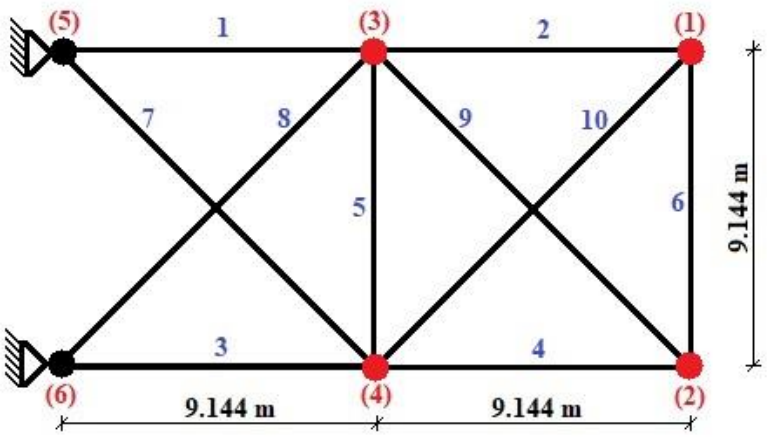
Verifica-se que a restrição de deslocamento vertical do nó 1 se tornou ativa após o processo de otimização em todos os algoritmos analisados.

6.2 OTIMIZAÇÃO PARAMÉTRICA DE TRELIÇA PLANA DE DEZ BARRAS SUJEITA A RESTRIÇÕES DE FREQUÊNCIAS NATURAIS

Neste problema, a estrutura a ser otimizada é a mesma do problema anterior: Uma treliça plana, de alumínio, composta por 10 barras, com módulo de Young $E = 68{,}95\ GPa$ e massa específica $\rho = 2767{,}99\ kg/m^3$. A diferença, neste caso, é que são utilizadas como restrições as múltiplas frequências naturais da treliça. As variáveis são novamente as seções transversais das barras, que podem variar em um intervalo de $0{,}645\ cm^2 \leq x_i \leq 200\ cm^2$.

Em cada um dos nós 1 ao 4 é fixada uma massa não estrutural concentrada de $453{,}6\ kg$, conforme pode ser observado na Figura 6.3. As restrições de frequência natural, que devem ser atendidas simultaneamente, são $f_1 \geq 7\ Hz$, $f_2 \geq 15\ Hz$ e $f_3 \geq 20\ Hz$.

Figura 6.3 – Treliça plana de 10 barras sujeita a restrições de frequência



Fonte: Adaptado de Miguel e Fadel Miguel, 2013

Este problema já foi estudado por alguns autores: Gomes (2011) utilizou o algoritmo PSO para minimizar a massa da treliça; Borges (2013) comparou o desempenho dos algoritmos HS e FA; Miguel e Fadel Miguel (2013) otimizaram a estrutura através dos algoritmos HS, ABC (*Artificial Bee Colony*) e FA, comparando os resultados obtidos em cada algoritmo.

No presente trabalho foram efetuadas dez execuções em cada algoritmo. Nas análises com o CIOA, WOA, SGA e PSO foram utilizadas 800 iterações e 250 agentes de pesquisa; no algoritmo FA foram empregadas 5000 iterações e 40 agentes de pesquisa; a resolução com o HS foi feita utilizando-se de 200000 iterações. Desta forma, todos os algoritmos minimizaram a massa da treliça após um número total de 200000 avaliações. Os resultados para a massa da treliça (kg) e a seção transversal de cada barra (cm²) são apresentados na Tabela 6.5 e comparados com aqueles encontrados por Miguel e Fadel Miguel (2013). Nota-se que o *Circle-Inspired Optimization Algorithm* produziu resultados compatíveis com os algoritmos com o qual foi comparado, mostrando-se eficiente na busca pela massa mínima da treliça deste problema.

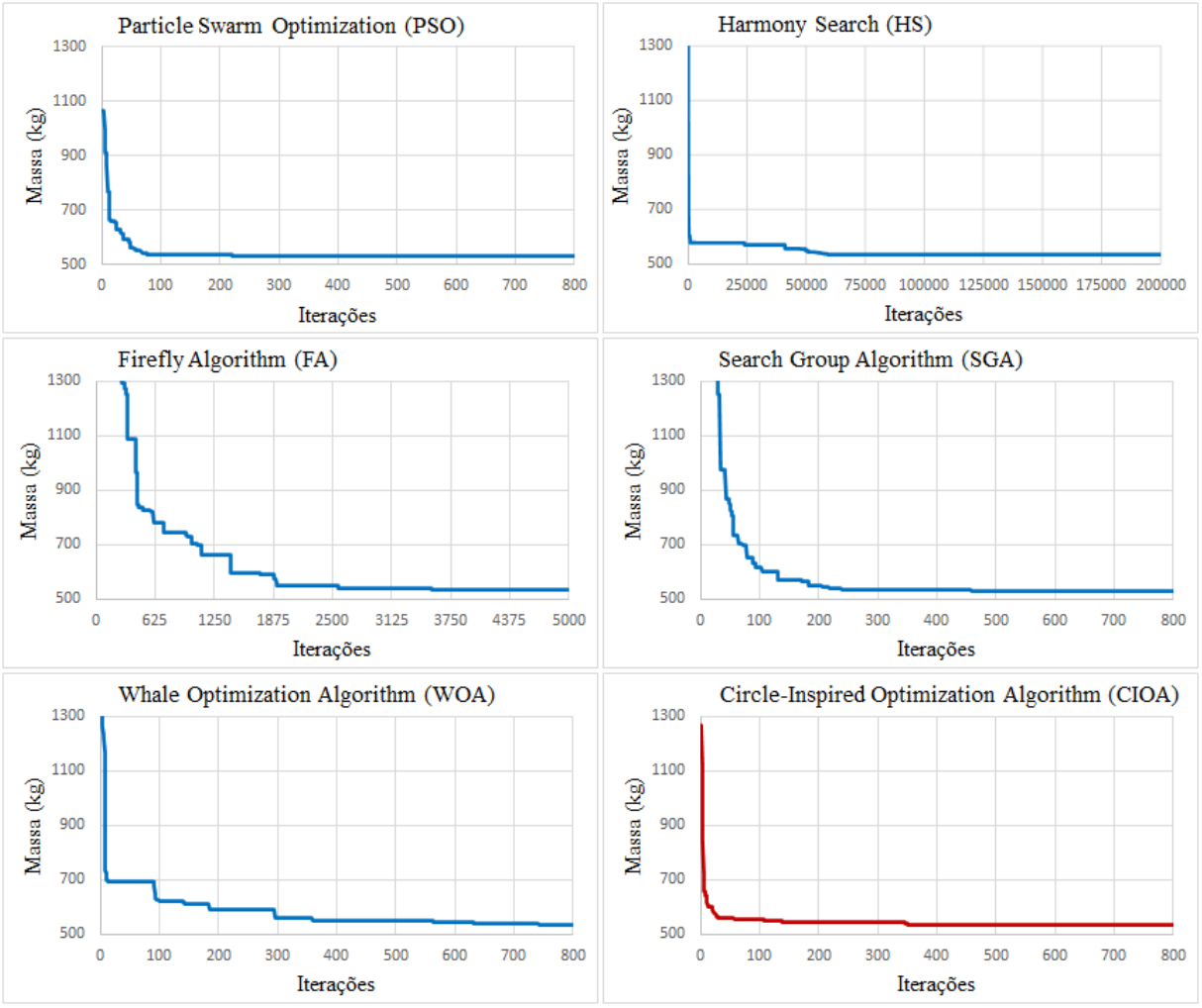
Tabela 6.5 – Projeto ótimo para a treliça plana de 10 barras sujeita a restrições de frequência

	Miguel e Fadel Miguel (2013)			Presente Trabalho					
Membro	HS	ABC	FA	PSO	HS	FA	SGA	WOA	CIOA
1	34,282	35,062	36,198	37,882	34,232	36,800	35,391	35,528	36,286
2	15,653	12,291	14,030	12,334	14,988	14,577	14,879	16,845	14,591
3	37,641	32,827	34,754	35,864	35,536	35,884	35,875	37,381	35,050
4	16,058	17,420	14,900	16,809	15,341	14,902	15,013	13,044	14,720
5	1,069	0,645	0,654	0,645	0,733	0,922	0,645	0,751	0, 776
6	4,740	4,828	4,672	4,714	4,800	4,614	4,613	4,909	4,730
7	22,505	19,588	23,467	19,751	24,041	23,813	23,844	27,227	23,608
8	24,603	28,710	25,508	29,379	24,600	24,889	24,125	21,127	25,288
9	12,867	15,782	12,707	13,256	13,657	12,747	12,528	10,686	12,341
10	12,099	13,055	12,351	10,315	12,235	11,143	12,533	13,367	12,384
Massa (kg)	534,99	536,98	531,28	534,21	534,14	532,38	**530,75**	533,69	532,20

Além de Miguel e Fadel Miguel (2013), cujos resultados obtidos fazem parte da Tabela 6.5, Gomes (2011) encontrou um projeto ótimo com massa de $537{,}98\ kg$ utilizando o PSO; Borges (2013) obteve massa mínima de $531{,}40\ kg$ com o FA e de $535{,}13\ kg$ através do HS.

Na Figura 6.4 são apresentadas as curvas de convergência de cada algoritmo. Nota-se que *Circle-Inspired Optimization Algorithm* (CIOA) apresentou convergência extremamente rápida. quando comparado aos outros algoritmos estudados neste trabalho.

Figura 6.4 – Curvas de convergência na otimização de treliça plana de 10 barras sujeita a restrições de frequência



A Tabela 6.6 mostra os valores médios de massa mínima, valores de desvio padrão e de coeficiente de variação obtidos após dez execuções de cada algoritmo além do tempo computacional médio necessário para cada execução obtidos no presente trabalho.

Tabela 6.6 – Análise estatística e de tempo computacional para a treliça plana de 10 barras sujeita a restrições de frequência

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	536,61	537,29	536,13	**530,83**	539,32	535,89
Desvio padrão (kg)	1,86	2,18	3,68	**0,07**	2,62	3,55
Coef. de variação (%)	0,346	0,405	0,686	**0,014**	0,486	0,663
Tempo (s)	213,84	204,55	212,35	**176,02**	195,67	190,31

O Quadro 6.2 ilustra o ranking onde os algoritmos são ordenados de acordo com os resultados apresentados nesta seção.

Quadro 6.2 – Ranking para a treliça plana de 10 barras sujeita a restrições de frequência

Parâmetro	Ranking
Massa mínima	[SGA] < [**CIOA**] < [FA] < [WOA] < [HS] < [PSO]
Massa média	[SGA] < [**CIOA**] < [FA] < [PSO] < [HS] < [WOA]
Desvio padrão	[SGA] < [PSO] < [HS] < [WOA] < [**CIOA**] < [FA]
Coeficiente de variação	[SGA] < [PSO] < [HS] < [WOA] < [**CIOA**] < [FA]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [HS] < [FA] < [PSO]

O CIOA obteve um bom desempenho em aspectos como identificação da massa mínima, massa média entre as dez execuções e tempo computacional de operação, superando nestes parâmetros quatro dos cinco algoritmos analisados. Contudo, divergências relativamente altas nos resultados obtidos a cada simulação influenciaram negativamente o desvio padrão e, por consequência, o coeficiente de variação. Ainda assim, ressalta-se que os valores obtidos nestes parâmetros estatísticos onde o CIOA apresentou um desempenho inferior são totalmente aceitáveis, uma vez que o coeficiente de variação após dez simulações apresentou um valor muito baixo, inferior a 1%.

É importante ressaltar que nenhuma restrição foi violada durante o processo de otimização. Na Tabela 6.7 são apresentadas as três primeiras frequências naturais de vibração obtidas por cada configuração final gerada pelos algoritmos de otimização. Em negrito, são indicados os casos em que a restrição se tornou ativa.

Tabela 6.7 – Primeiras frequências naturais obtidas após a otimização da treliça plana de 10 barras sujeitas a restrições de frequência

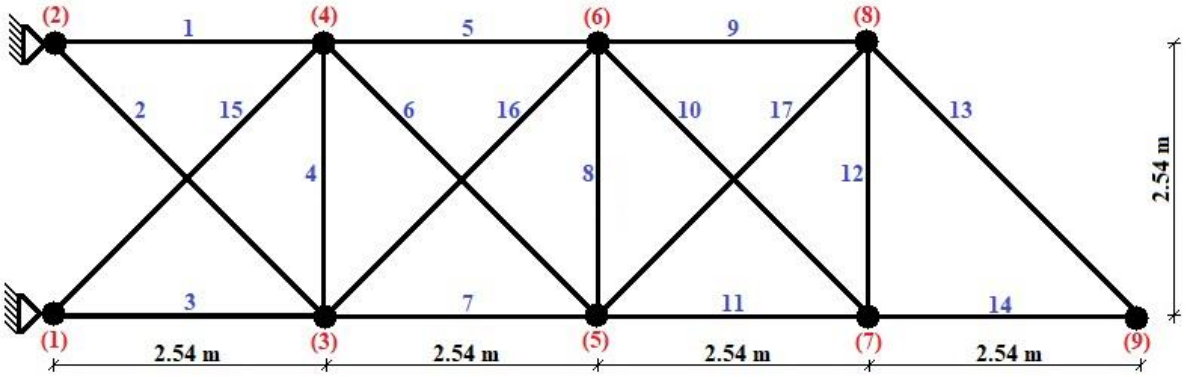
Frequências	PSO	HS	FA	SGA	WOA	CIOA
1	**7,000**	7,003	7,001	**7,000**	7,001	7,004
2	16,027	16,283	16,560	16,186	16,282	16,370
3	**20,000**	20,135	20,010	20,001	**20,000**	20,006

Verifica-se que o algoritmo PSO tornou ativas as restrições da 1ª e 3ª frequências naturais. No algoritmo SGA, tornou-se ativa apenas a restrição referente a 1ª frequência de vibração, enquanto no algoritmo WOA, a restrição da 3ª frequência natural de vibração ficou ativa ao final da otimização. O algoritmo CIOA, assim como o FA e o HS, não ativou nenhuma das restrições impostas neste problema.

6.3 OTIMIZAÇÃO PARAMÉTRICA DE TRELIÇA PLANA DE 17 BARRAS SUJEITA A RESTRIÇÕES DE TENSÃO E DESLOCAMENTO

Este problema consiste na otimização paramétrica de uma treliça plana de 17 barras, mostrada na Figura 6.5. A treliça tem módulo de Young $E = 206{,}84\ GPa$, massa específica $\rho = 7418{,}21\ kg/m^3$ e está sujeita a uma carga vertical de $-444{,}82\ kN$ no nó 9. As restrições de tensão são de $\pm\ 344{,}74\ MPa$ para todos os membros. As restrições de deslocamento são de $\pm\ 5{,}08\ cm$ nas direções x e y para todos os nós. O intervalo das variáveis é de $0{,}645\ cm^2 \leq x_i \leq 200\ cm^2$.

Figura 6.5 – Treliça plana de 17 barras



Fonte: Baseado em Miguel e Fadel Miguel, 2013

Apesar de possuir os mesmos tipos de restrições que a treliça da seção 6.1, este problema apresenta uma complexidade maior, devido ao maior número de variáveis de projeto e ao maior número de restrições, uma vez que, diferente da treliça de 10 barras, neste problema, as restrições de deslocamento estão presentes em ambas as direções de todos os nós.

Esse problema já foi estudado por outros autores. Lee e Geem (2004) otimizaram a treliça através do algoritmo *Harmony Search* (HS) enquanto Miguel e Fadel Miguel (2013) realizaram a otimização através do *Harmony Search* (HS), *Artificial Bee Colony* (ABC) e *Firefly Algorithm* (FA), comparando os resultados obtidos em cada um dos três algoritmos.

No presente trabalho, dez simulações foram realizadas por cada algoritmo. Nas análises com o CIOA, WOA, SGA e PSO foram utilizadas 1600 iterações e 250 agentes de pesquisa; o algoritmo FA realizou a otimização com 10000 iterações e 40 agentes de pesquisa; a resolução com o HS foi feita utilizando-se de 400000 iterações. Desta forma, todos os algoritmos

minimizaram a massa da treliça após um número total de 400000 avaliações. Os resultados obtidos para a massa mínima (kg) e para a seção de cada barra (cm²) são apresentados na Tabela 6.8 e comparados com os valores encontrados por Miguel e Fadel Miguel (2013).

O CIOA produziu resultados coerentes quando comparado aos demais algoritmos elencados neste estudo, com uma massa mínima próxima àquela obtida pelos algoritmos de melhor desempenho.

Tabela 6.8 - Projeto ótimo para a treliça plana de 17 barras

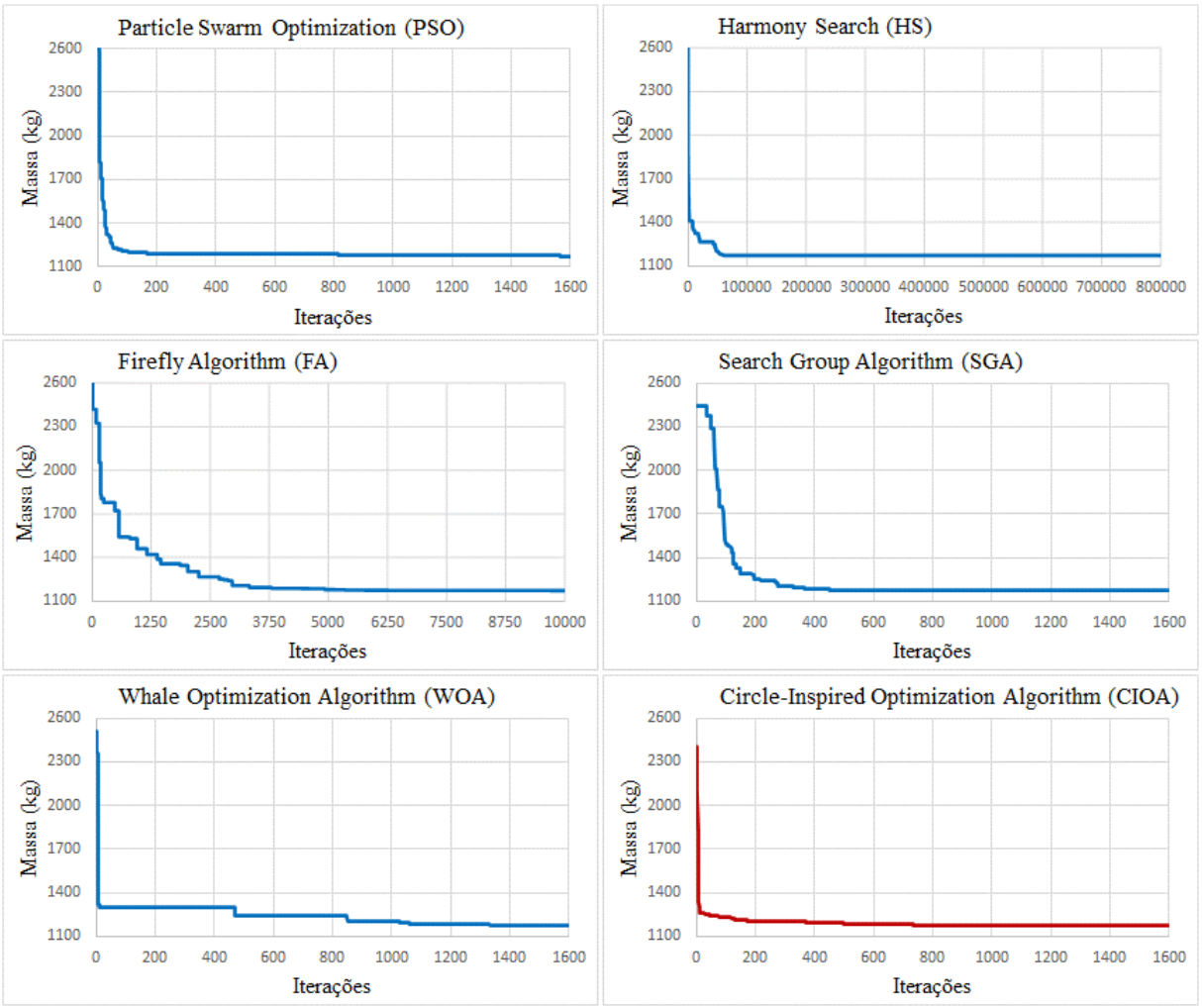
	Miguel e Fadel Miguel (2013)				Presente Trabalho				
Membro	HS	ABC	FA	PSO	HS	FA	SGA	WOA	CIOA
1	104,440	102,178	101,960	101,210	101,589	105,414	102,919	101,780	100,776
2	0,785	0,645	0,645	1,807	1,604	0,925	0,645	0,665	0,918
3	78,906	74,002	76,447	81,107	78,632	79,072	77,782	77,852	76,284
4	0,711	0,645	0,648	0,645	0,672	0,645	0,645	0,675	0,821
5	51,578	51,666	53,225	55,605	53,152	54,702	51,659	52,912	53,923
6	33,995	34,700	35,220	34,526	35,535	35,241	35,855	35,678	35,912
7	75,888	75,023	76,507	74,851	78,873	74,405	77,210	78,018	78,221
8	0,676	0,645	0,645	0,645	0,726	0,669	0,645	0,678	0,676
9	49,838	50,181	51,102	50,776	50,280	49,295	51,482	51,749	52,136
10	0,995	0,645	0,645	1,101	0,926	0,645	0,645	0,645	0,678
11	27,752	30,444	25,842	27,463	28,420	26,069	26,476	26,616	25,289
12	0,662	0,645	0,645	1,204	0,826	0,645	0,645	0,655	0,676
13	36,946	39,770	37,258	36,912	36,557	35,466	36,520	36,080	36,787
14	24,954	24,393	26,921	26,224	24,200	25,763	25,813	24,755	25,608
15	35,919	35,483	36,433	34,273	34,842	35,650	35,653	36,744	35,456
16	1,408	1,193	0,645	1,615	0,816	0,717	0,648	0,932	0,914
17	36,503	38,590	36,081	33,580	34,877	36,588	35,896	35,527	36,023
Massa (Kg)	1173,22	1174,63	1171,47	1173,64	1172,44	1172,11	**1171,15**	1171,67	1172,16

Os resultados obtidos no presente trabalho são similares àqueles obtidos por pesquisas de outros autores, que utilizaram diferentes algoritmos para a otimização. Além de Miguel e Fadel Miguel (2013), cujos resultados obtidos fazem parte da Tabela 6.8, Lee e Geem (2004) encontraram uma massa mínima de $1170{,}64\ kg$.

Na Figura 6.6 são ilustradas as curvas de convergência do projeto ótimo obtido por cada algoritmo onde verifica-se que o CIOA, bem como outros algoritmos como o PSO e HS

apresentam convergência extremamente rápida para este problema, ainda que não apresentem o melhor projeto entre os algoritmos estudados.

Figura 6.6 – Curvas de convergência para a treliça plana de 17 barras



A Tabela 6.9 apresenta, para cada algoritmo, dados estatísticos obtidos após as dez execuções realizadas: Massa ótima média, Desvio Padrão e Coeficiente de Variação, além do tempo computacional médio de cada execução.

Tabela 6.9 – Análise estatística e de tempo computacional para a treliça plana de 17 barras.

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	1177,28	1175,51	1172,92	**1171,18**	1173,68	1173,38
Desvio padrão (kg)	4,75	3,89	1,11	**0,02**	1,22	0,56
Coef. de variação (%)	0,403	0,331	0,095	**0,002**	0,104	0,048
Tempo (s)	408,02	431,56	339,43	**226,99**	335,43	292,02

Os valores gerados pelo CIOA são coerentes com os resultados produzidos pelos demais algoritmos. Além disso, o CIOA se mostra bastante estável, uma vez que o desvio padrão e o coeficiente de variação obtidos após as dez execuções apresentam valores extremamente baixos.

Por fim, apresenta-se no Quadro 6.3 o ranking onde os algoritmos são classificados de acordo com a ordem crescente dos resultados obtidos nesta seção.

Quadro 6.3 – Ranking para a treliça plana de 17 barras

Parâmetro	Ranking
Massa mínima	[SGA] < [WOA] < [FA] < [**CIOA**] < [HS] < [PSO]
Massa média	[SGA] < [FA] < [**CIOA**] < [WOA] < [HS] < [PSO]
Desvio padrão	[SGA] < [**CIOA**] < [FA] < [WOA] < [HS] < [PSO]
Coeficiente de variação	[SGA] < [**CIOA**] < [FA] < [WOA] < [HS] < [PSO]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [FA] < [PSO] < [HS]

O (CIOA) obteve resultados intermediários, quando comparado aos demais algoritmos. Os destaques positivos são em relação ao desvio padrão, coeficiente de variação e tempo computacional, nos quais o CIOA apresentou o segundo melhor desempenho, superando a performance de quatro dos cinco algoritmos com o qual foi comparado.

Nenhuma restrição foi violada durante a otimização da treliça. Nas tabelas 6.10 e 6.11 são apresentados, respectivamente, os deslocamentos e tensões obtidos através da configuração da estrutura obtida após o processo de otimização.

Tabela 6.10 – Deslocamentos (cm) obtidos após a otimização da treliça plana de 17 barras

Nó	Direção X						Direção Y					
	PSO	HS	FA	SGA	WOA	CIOA	PSO	HS	FA	SGA	WOA	CIOA
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	-0,21	-0,21	-0,21	-0,21	-0,21	-0,22	-0,60	-0,55	-0,60	-0,63	-0,65	-0,63
4	0,21	0,21	0,21	0,21	0,21	0,22	-0,64	-0,64	-0,63	-0,64	-0,63	-0,64
5	-0,42	-0,42	-0,42	-0,42	-0,42	-0,42	-1,71	-1,69	-1,69	-1,70	-1,68	-1,70
6	0,41	0,42	0,41	0,42	0,42	0,42	-1,61	-1,66	-1,65	-1,67	-1,63	-1,64
7	-0,63	-0,61	-0,64	-0,63	-0,63	-0,64	-3,07	-3,04	-3,04	-3,06	-3,04	-3,05
8	0,63	0,64	0,63	0,64	0,63	0,63	-3,20	-3,18	-3,16	-3,18	-3,16	-3,18
9	-0,83	-0,84	-0,85	-0,84	-0,85	-0,86	**-5,08**	**-5,08**	**-5,08**	**-5,08**	-5,07	**-5,08**

Tabela 6.11 – Tensões (MPa) obtidas após a otimização da treliça plana de 17 barras

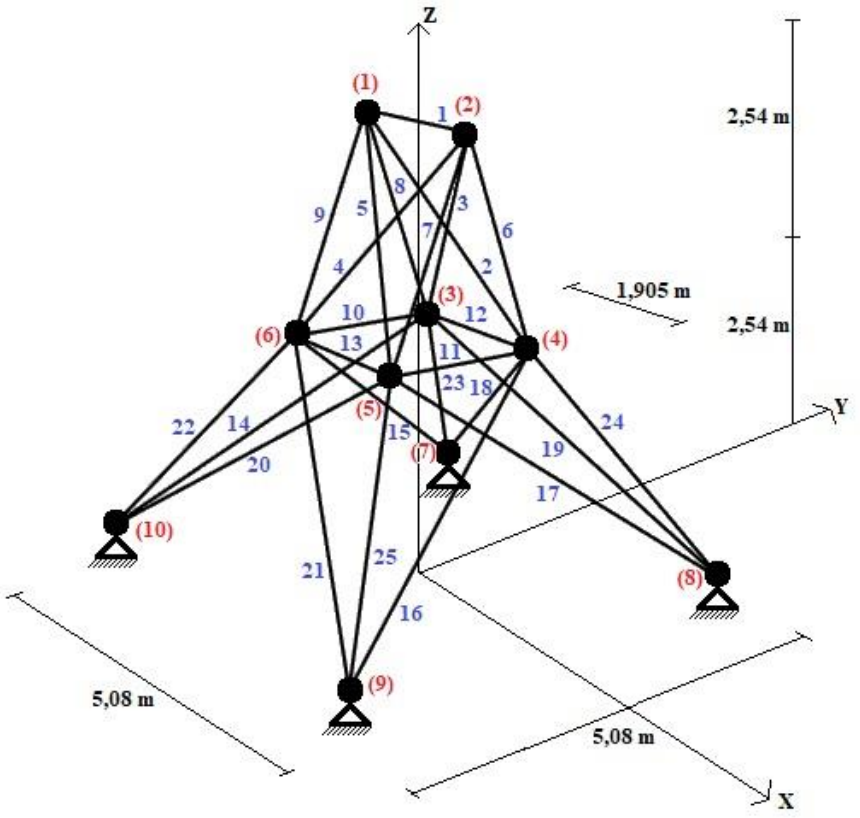
Membro	PSO	HS	FA	SGA	WOA	CIOA
1	173,773	173,582	167,801	172,124	173,998	175,481
2	160,635	139,999	159,322	170,861	177,073	167,238
3	-167,061	-171,729	-170,083	-172,565	-172,479	-176,356
4	-35,597	-70,064	-23,721	-4,388	-16,629	-10,833
5	163,271	169,478	164,259	173,666	169,922	166,831
6	174,735	172,583	174,937	172,490	172,573	171,245
7	-175,847	-167,775	-178,156	-171,864	-169,834	-169,327
8	78,391	26,335	36,765	20,355	41,676	47,402
9	172,614	175,096	179,168	171,604	170,634	169,341
10	169,202	141,398	141,010	135,656	145,221	141,047
11	-166,767	-159,774	-173,099	-170,346	-169,613	-178,568
12	-109,408	-112,008	-99,709	-95,923	-101,119	-100,031
13	170,424	172,079	177,373	172,254	174,354	171,003
14	-169,623	-183,810	-172,658	-172,324	-179,689	-173,704
15	-175,078	-174,104	-172,323	-173,351	-167,999	-173,093
16	-159,626	-193,595	-175,363	-163,681	-143,377	-154,209
17	-181,787	-176,614	-169,448	-172,811	-174,432	-171,976

Verifica-se na Tabela 6.10 que todos os algoritmos, exceto o WOA, ativaram a restrição de deslocamento na direção vertical do nó 9. As restrições de tensão, na Tabela 6.11 apresentaram valores distantes dos limites impostos pela restrição para este problema de otimização.

6.4 OTIMIZAÇÃO PARAMÉTRICA DE TRELIÇA ESPACIAL DE 25 BARRAS SUJEITA A RESTRIÇÕES DE TENSÃO E DESLOCAMENTO

Neste problema, é estudada a otimização paramétrica da treliça espacial mostrada na Figura 6.7. As variáveis de projeto são contínuas e representam as seções transversais das 25 barras que constituem a estrutura. Em relação às propriedades dos materiais, a treliça, constituída de alumínio, apresenta módulo de Young $E = 68{,}95\ GPa$ e massa específica $\rho = 2767{,}99\ Kg/m^3$. O intervalo das variáveis de projeto é de $0{,}0645\ cm^2 \leq x_i \leq 20\ cm^2$.

Figura 6.7 – Treliça espacial de 25 barras



Fonte: Baseado em Borges, 2013

Com a finalidade de manter a simetria da estrutura, as barras são agrupadas em oito grupos, conforme a Tabela 6.12, nos quais os elementos de cada grupo possuem a mesma seção transversal. Desta maneira, oito são as variáveis de projeto desse problema de otimização.

Tabela 6.12 – Detalhe do agrupamento de barras para a treliça espacial de 25 barras

Grupo	Membros
1	1
2	2-5
3	6-9
4	10-11
5	12-13
6	14-17
7	18-21
8	22-25

Fonte: Baseado em Borges, 2013

A estrutura deve ser otimizada de modo a suportar dois casos de carregamento independentes, isto é, dois carregamentos que **não** ocorrem simultaneamente. No primeiro caso, as cargas atuam nos nós 1, 2, 3 e 6 enquanto no segundo caso somente os nós 1 e 2 estão sujeitos a cargas, conforme pode ser visto na Tabela 6.13.

Tabela 6.13 – Componentes de cargas nodais (kN) para a treliça espacial de 25 barras

Caso	Nó	X	Y	Z
1	1	4,4482	44,482	-22,241
	2	0	44,482	-22,241
	3	2,2241	0	0
	6	2,2241	0	0
2	1	0	88,964	-22,241
	2	0	-88,964	-22,241

Fonte: Baseado em Borges, 2013

As restrições que devem ser atendidas referem-se às tensões em todas as barras, cujos limites de tração e compreensão são apresentados na Tabela 6.14. As restrições de deslocamentos são de $\pm 0{,}889\ cm$ nos nós 1 e 2 para as direções X, Y e Z.

Tabela 6.14 – Restrições de tensão (MPa) para a treliça espacial de 25 barras

Restrição	Valor
Tensão de tração para todos os membros	275,79
Tensão de compressão para o Grupo 1	-241,95
Tensão de compressão para o Grupo 2	-79,91
Tensão de compressão para o Grupo 3	-119,31
Tensão de compressão para o Grupo 4	-241,95
Tensão de compressão para o Grupo 5	-241,95
Tensão de compressão para o Grupo 6	-46,60
Tensão de compressão para o Grupo 7	-47,98
Tensão de compressão para o Grupo 8	-76,41

Fonte: Baseado em Borges, 2013

Este problema já foi estudado por diversos autores: Lee e Geem (2004) efetivaram a otimização através do algoritmo HS; Borges (2013) otimizou a treliça por meio dos algoritmos HS e FA, enquanto Miguel e Fadel Miguel (2013) além de utilizarem o HS e o FA, também usaram o

Artificial Bee Colony (ABC); Bekdas et al. (2015) realizaram a otimização através do *Buterfly Optimization Algorithm* (BOA).

Dez simulações foram realizadas para cada algoritmo. Nas análises com o CIOA, WOA, SGA e PSO foram utilizadas 800 iterações e 250 agentes de pesquisa; no algoritmo FA foram empregadas 5000 iterações e 40 agentes de pesquisa; a resolução com o HS foi feita utilizando-se de 200000 iterações. Desta forma, todos os algoritmos minimizaram a massa da treliça espacial fazendo uso de 200000 avaliações. Os resultados obtidos para a seção transversal (cm^2) de cada grupo de barras, bem como a massa da treliça (kg) são apresentados na Tabela 6.15 e comparados com os obtidos por Borges (2013). O *Circle-Inspired Optimization Algorithm* encontrou uma massa ótima muito próxima ao melhor valor obtido entre todos os algoritmos estudados no presente trabalho.

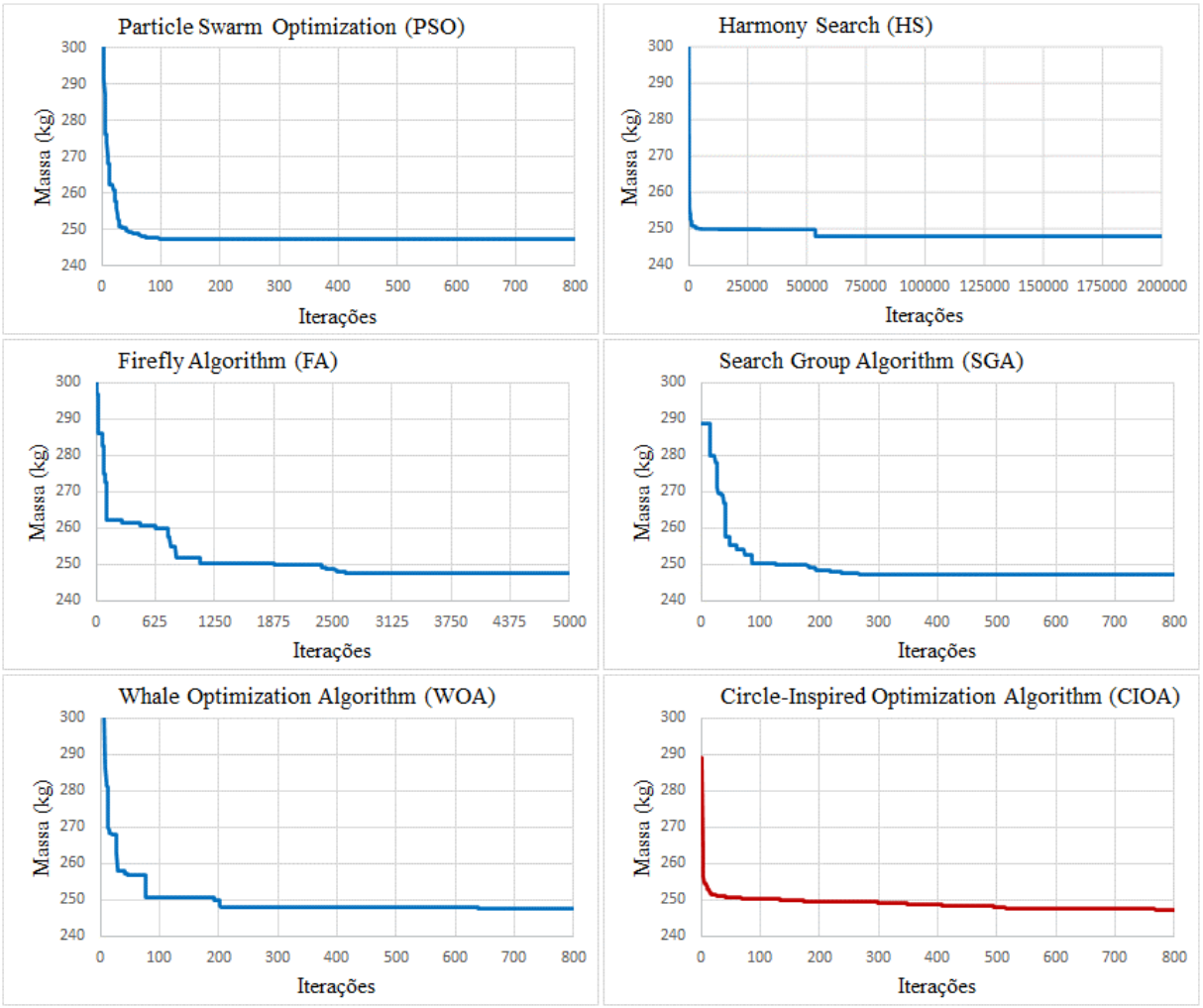
Tabela 6.15 – Projeto ótimo para a treliça espacial de 25 barras

	Borges (2013)			Presente Trabalho				
Grupo	HS	FA	PSO	HS	FA	SGA	WOA	CIOA
1	0,201	0,065	0,224	0,068	0,065	0,067	0,229	0,072
2	12,409	12,473	12,983	12,774	13,077	12,799	12,919	13,017
3	19,625	19,760	19,118	18,213	18,668	19,345	19,134	19,022
4	0,081	0,065	0,065	0,066	0,074	0,065	0,066	0,065
5	0,516	0,089	0,065	0,096	0,065	0,065	0,065	0,071
6	4,743	4,411	4,585	5,029	4,500	4,408	4,565	4,406
7	11,065	10,919	10,755	11,520	10,920	10,821	10,810	10,787
8	16,804	17,032	17,029	16,896	17,222	17,170	17,036	17,267
Massa (Kg)	248,82	247,35	247,39	247,95	247,44	**247,27**	247,41	247,29

Os resultados obtidos são coerentes com aqueles obtidos por outros autores que estudaram este mesmo problema de otimização através de vários algoritmos. Além de Borges (2013), cujos resultados obtidos fazem parte da Tabela 6.15, Lee e Geem (2004) obtiveram uma massa mínima de $246{,}93\ kg$ com o algoritmo HS; Borges (2013) encontrou massas de $247{,}35\ kg$ com o FA e $248{,}82\ kg$ com o HS; Miguel e Fadel Miguel (2013) obtiveram massas de $247{,}31\ kg$ com o FA, $248{,}09\ kg$ com o *Artificial Bee Colony* (ABC) e $247{,}84\ kg$ com o HS; Bekdas et al. (2015) utilizaram o *Flower Pollination Algorithm* (FPA) e obtiveram uma massa ótima de $247{,}28\ kg$.

Na Figura 6.8 encontram-se as curvas de convergência geradas por cada algoritmo em sua melhor execução: aquela na qual foi obtido o projeto ótimo. O CIOA apresenta elevada taxa de convergência ainda nas primeiras iterações, tendo seu resultado refinado nas iterações seguintes.

Figura 6.8 – Curvas de convergência para a treliça espacial de 25 barras



Na Tabela 6.16 é apresentada a análise estatística com valores de massa média, desvio padrão e coeficiente de variação entre os resultados obtidos após dez execuções, além do tempo computacional médio por execução. Os resultados obtidos pelo CIOA são muito similares àqueles obtidos pelos demais algoritmos e próximos do melhor resultado para cada parâmetro, obtido pelo SGA.

Tabela 6.16 – Análise estatística e de tempo computacional para a treliça espacial de 25 barras

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	247,748	248,713	248,465	**247,281**	247,632	247,315
Desvio padrão (kg)	0,397	0,664	1,210	**0,007**	0,190	0,025
Coef. de variação (%)	0,160	0,267	0,487	**0,003**	0,077	0,010
Tempo (s)	237,23	226,20	248,02	**182,72**	200,97	189,65

No Quadro 6.4 é apresentado o ranking, em ordem crescente, referente aos parâmetros avaliados nesta seção.

Quadro 6.4 – Ranking para a treliça espacial de 25 barras

Parâmetro	Ranking
Massa mínima	[SGA] < [**CIOA**] < [PSO] < [WOA] < [FA] < [HS]
Massa média	[SGA] < [**CIOA**] < [WOA] < [PSO] < [FA] < [HS]
Desvio padrão	[SGA] < [**CIOA**] < [WOA] < [PSO] < [HS] < [FA]
Coeficiente de variação	[SGA] < [**CIOA**] < [WOA] < [PSO] < [HS] < [FA]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [HS] < [PSO] < [FA]

Observa-se que o CIOA apresentou a segunda melhor performance em todas as análises realizadas neste problema de otimização estrutural.

Nenhuma restrição foi violada durante a solução deste problema de otimização. Nas Tabela 6.17 e 6.18 são apresentadas as tensões de cada barra, resultantes após o processo de otimização para os casos de carregamento 1 e 2, respectivamente. Já a Tabela 6.19 apresenta os deslocamentos para os nós 1 e 2 obtidos após a otimização, considerando-se os dois casos distintos de carregamento.

Tabela 6.17 - Tensões (MPa) obtidas após a otimização da treliça espacial de 25 barras no caso de carregamento 1

Grupo	Membro	PSO	HS	FA	SGA	WOA	CIOA
1	1	22,384	24,092	25,353	24,562	22,150	25,070
2	2	-21,053	-20,837	-20,898	-20,903	-21,074	-20,910
	3	-18,072	-17,844	-17,939	-17,880	-18,079	-17,936
	4	17,400	17,621	17,719	17,656	17,395	17,695
	5	14,420	14,591	14,760	14,632	14,400	14,722
3	6	-29,301	-31,058	-30,009	-29,202	-29,322	-29,498
	7	17,250	18,144	17,411	17,044	17,286	17,146
	8	-27,644	-29,319	-28,313	-27,565	-27,667	-27,833
	9	18,906	19,883	19,108	18,682	18,941	18,811
4	10	-10,208	-8,851	-10,638	-11,046	-10,269	-11,074
	11	-12,528	-10,034	-13,271	-13,924	-12,644	-14,012
5	12	-25,376	-18,889	-24,676	-26,879	-25,133	-26,431
	13	0,642	0,367	0,107	0,755	0,659	0,168
6	14	-33,186	-30,270	-33,788	-34,507	-33,328	-34,522
	15	25,393	23,137	25,898	26,424	25,507	26,437
	16	-36,085	-32,927	-36,732	-37,515	-36,239	-37,531
	17	22,494	20,481	22,953	23,416	22,597	23,428
7	18	-25,263	-23,875	-24,887	-25,379	-25,187	-25,225
	19	-25,887	-24,458	-25,502	-26,000	-25,809	-25,847
	20	14,739	13,828	14,248	14,645	14,720	14,484
	21	14,114	13,245	13,634	14,024	14,099	13,861
8	22	28,486	28,680	28,287	28,252	28,448	28,189
	23	-34,245	-34,366	-33,863	-33,841	-34,207	-33,758
	24	-38,609	-38,761	-38,180	-38,170	-38,569	-38,064
	25	24,123	24,284	23,971	23,923	24,086	23,884

Verifica-se na Tabela 6.17 que nenhuma tensão tornou a restrição ativa para o primeiro caso de carregamento.

Tabela 6.18 - Tensões (MPa) obtidas após a otimização da treliça espacial de 25 barras no caso de carregamento 2

Grupo	Membro	PSO	HS	FA	SGA	WOA	CIOA
1	1	32,926	36,151	37,156	36,415	32,670	36,791
2	2	-48,073	-48,169	-47,614	-48,208	-48,232	-47,761
	3	47,085	47,834	47,293	47,876	47,224	47,407
	4	-48,073	-48,169	-47,614	-48,208	-48,232	47,407
	5	47,085	47,834	47,293	47,876	47,224	-47,761
3	6	33,885	35,581	34,389	33,425	33,908	33,807
	7	-45,760	-48,430	-46,928	-45,525	-45,766	-46,096
	8	-46,760	-48,430	-46,928	-45,525	-45,766	-46,096
	9	33,885	35,581	34,389	33,425	33,908	33,807
4	10	-11,267	-9,407	-11,922	-12,451	-11,354	-12,508
	11	-11,267	-9,407	-11,922	-12,451	-11,354	-12,508
5	12	-11,838	-11,108	-12,701	-12,885	-11,699	-12,944
	13	-11,838	-11,108	-12,701	-12,885	-11,699	-12,944
6	14	-17,175	-15,993	-17,252	-18,072	-17,332	-17,724
	15	6,481	6,203	6,416	6,980	6,598	6,629
	16	6,481	6,203	6,416	6,980	6,598	6,629
	17	-17,175	-15,993	-17,252	-18,072	-17,332	-17,724
7	18	37,024	34,533	36,174	36,689	36,876	36,675
	19	**-47,980**	-45,101	-47,366	-47,979	-47,768	-47,969
	20	**-47,980**	-45,101	-47,366	-47,979	-47,768	-47,969
	21	37,024	34,533	36,174	36,689	36,876	36,675
8	22	-8,458	-8,022	-8,527	-8,045	-8,356	-8,409
	23	-1,753	-2,090	-1,394	-1,902	-1,856	-1,497
	24	-8,458	-8,022	-8,527	-8,045	-8,356	-8,409
	25	-1,753	-2,090	-1,394	-1,902	-1,856	-1,497

Verifica-se na Tabela 6.18 que, na otimização via PSO, as tensões nas barras 19 e 20 atingiram o limite para este grupo de barras, tornando a restrição ativa.

Tabela 6.19 – Deslocamentos (cm) obtidos após a otimização da treliça espacial de 25 barras

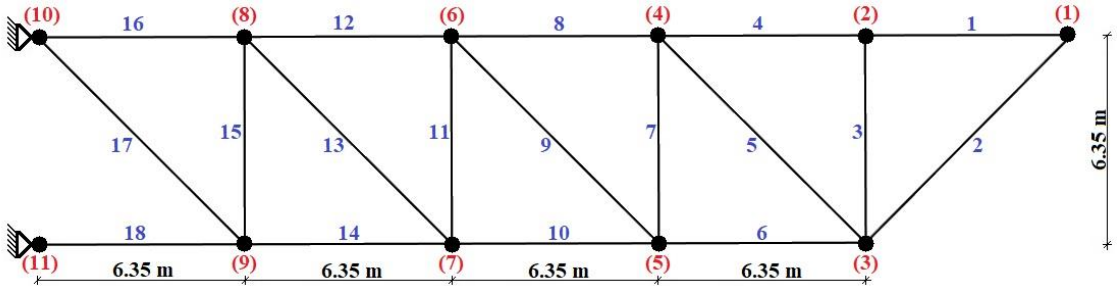
DIREÇÃO X							
Caso	Nó	PSO	HS	FA	SGA	WOA	CIOA
1	1	0,019	0,017	0,015	0,017	0,020	0,016
	2	0,081	0,084	0,085	0,084	0,081	0,085
2	1	-0,045	-0,050	-0,051	-0,050	-0,045	-0,051
	2	0,045	0,050	0,051	0,050	0,045	0,051
DIREÇÃO Y							
Caso	Nó	PSO	HS	FA	SGA	WOA	CIOA
1	1	**0,889**	0,883	**0,889**	**0,889**	**0,889**	**0,889**
	2	**0,889**	0883	**0,889**	**0,889**	**0,889**	**0,889**
2	1	**0,889**	0,887	**0,889**	**0,889**	**0,889**	**0,089**
	2	**-0,889**	-0,887	**-0,889**	**-0,889**	**-0,889**	**-0,889**
DIREÇÃO Z							
Caso	Nó	PSO	HS	FA	SGA	WOA	CIOA
1	1	-0,057	-0,057	-0,058	-0,058	-0,057	-0,058
	2	-0,082	-0,082	-0,083	-0,083	-0,082	-0,083
2	1	-0,073	-0,073	-0,074	-0,073	-0,072	-0,074
	2	-0,073	-0,073	-0,074	-0,073	-0,072	-0,074

Verifica-se na Tabela 6.19 que para os deslocamentos segundo a direção Y, todos os algoritmos exceto o HS tornaram a restrição ativa nos dois casos de carregamento. Isto é, com a configuração final de seções transversais obtidas após a otimização, os deslocamentos dos nós 1 e 2 atingiram o máximo valor permitido, de $\pm 0{,}889\ cm$.

6.5 OTIMIZAÇÃO PARAMÉTRICA E DE FORMA DE TRELIÇA PLANA DE 18 BARRAS SUJEITA A RESTRIÇÕES DE TENSÃO E FLAMBAGEM

Neste problema é realizada a otimização paramétrica e de forma da treliça plana de 18 barras apresentada na Figura 6.9. As variáveis de projeto são contínuas e representam as seções transversais das barras que constituem a estrutura e as coordenadas de alguns dos nós. A treliça é constituída de alumínio e tem as propriedades descritas a seguir: módulo de Young $E = 68{,}95\ GPa$ e massa específica $\rho = 2767{,}99\ Kg/m^3$. O intervalo das variáveis de projeto referente às áreas é de $22{,}5806\ cm^2 \leq x_i \leq 116{,}1288\ cm^2$.

Figura 6.9 – Treliça plana de 18 barras: Configuração inicial



Fonte: Baseado em Miguel e Fadel Miguel, 2013

A estrutura está sujeita a um carregamento de forças verticais de $-88{,}964\ kN$ nos nós do banzo superior da treliça (com exceção do nó 10). As áreas das seções transversais são divididas em quatro grupos, conforme a Tabela 6.20, de modo que todas as barras de um mesmo grupo deverão possuir a mesma área ao final da otimização

Tabela 6.20 - Detalhe do agrupamento de barras para a treliça plana de 18 barras

Grupo	Membros
1	1, 4, 8, 12 e 16
2	2, 6, 10, 14 e 18
3	3, 7, 11 e 15
4	5, 9, 13 e 17

Fonte: Baseado em Miguel e Fadel Miguel, 2013

Em relação à otimização de forma, os nós livres do banzo inferior (nós 3, 5, 7 e 9) podem se mover em qualquer direção do plano X-Y. Desta forma, o número total de variáveis do problema é doze, sendo quatro variáveis correspondentes a área das barras de cada grupo e oito variáveis correspondentes as coordenadas X e Y dos nós que podem ser movidos.

As restrições de tensão que devem ser atendidas nesse problema são de $\pm 137{,}8951\ MPa$ em todas as barras. Além disso, restrições de flambagem também devem ser atendidas, sendo que cada barra da treliça deve suportar um limite de tensão dado pela Equação 6.1

$$\sigma_f = \frac{-KEA}{L^2} \tag{6.1}$$

onde σ_f é a tensão de flambagem, K é uma constante determinada a partir da geometria da seção transversal da barra, E é o módulo de elasticidade do material, A e L são, respectivamente a área e o comprimento da barra analisada. Foi adotado para a constante um valor de $K = 4$.

Este problema já foi estudado por outros autores: Lee e Geem (2005) otimizaram a estrutura através do *Harmony Search* (HS); Borges (2013) utilizou para a otimização os algoritmos HS e FA; Miguel e Fadel Miguel (2013) utilizaram os algoritmos HS, ABC e FA.

No presente trabalho a otimização foi realizada utilizando-se 600000 avaliações. Estas avaliações são compostas da seguinte maneira: 2400 iterações e 250 agentes de pesquisa nos algoritmos CIOA, WOA, SGA e PSO; 15000 iterações e 40 agentes de pesquisa no algoritmo FA; 600000 iterações no algoritmo HS. Os resultados obtidos em termos de massa mínima (kg), seção transversal (cm²) e posição das coordenadas (m) são apresentados na Tabela 6.21 e comparados com os encontrados por Miguel e Fadel Miguel. Ressalta-se que, as variáveis referentes a áreas são apresentadas como A_i e as referentes a coordenadas horizontais e verticais dos nós como, respectivamente, X_j e Y_j. Os índices i se referem ao número do **grupo** das barras enquanto os índices j representam o número do **nó** da estrutura.

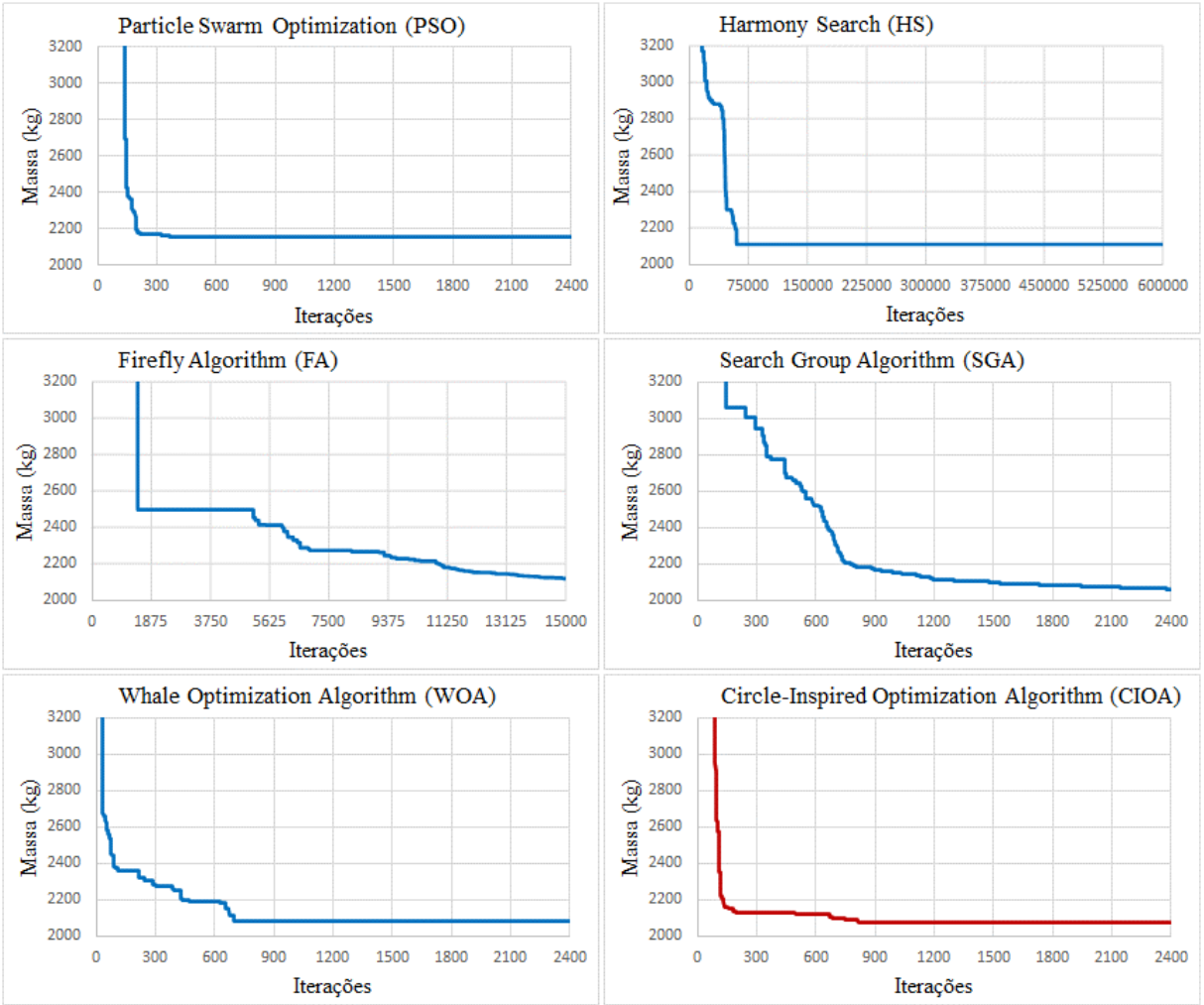
Tabela 6.21 – Projeto ótimo para a treliça plana de 18 barras

	Miguel e Fadel Miguel (2013)			Presente Trabalho					
Variável	HS	ABC	FA	PSO	HS	FA	SGA	WOA	CIOA
A_1	82,031	75,683	76,306	77,790	89,004	76,139	87,056	80,531	86,586
A_2	104,970	115,940	112,340	100,131	122,759	99,447	115,800	123,560	119,420
A_3	49,195	33,287	36,906	59,294	36,881	55,567	39,179	29,750	34,590
A_4	23,338	32,168	31,757	31,556	11,755	34,906	13,906	22,816	14,310
X_3	22,655	23,100	23,102	22,849	23,629	22,549	23,305	23,714	22,515
Y_3	3,890	4,710	4,550	2,607	4,994	3,370	4,809	5,000	4,455
X_5	15,523	16,476	16,202	15,480	16,563	15,260	16,392	17,172	16,555
Y_5	2,980	3,840	3,530	2,347	3,803	2,520	3,587	4,373	3,786
X_7	9,776	10,650	10,361	9,576	10,443	9,522	10,432	11,225	10,702
Y_7	1,930	2,400	2,220	1,482	2,752	1,454	2,554	2,285	2,741
X_9	4,679	5,189	5,076	4,534	4,870	4,515	5,064	5,502	5,223
Y_9	0,690	0,440	0,440	0,319	1,197	0,192	1,168	0,993	1,199
Massa (kg)	2072,5	2064,9	**2058,8**	2156,9	2109,9	2118,5	**2062,5**	2083,8	2075,4

Os resultados obtidos na Tabela 6.21 demonstram que, no presente trabalho, o CIOA obteve o segundo melhor resultado para a massa mínima. Nota-se ainda que todos os resultados neste trabalho são relativamente próximos aos encontrados por outros autores que também estudaram este problema: Lee e Geem (2005) encontraram uma massa de $2048{,}24\ kg$ com o HS; Borges (2013) obteve massas de $2073\ kg$ com o HS e $2055{,}41\ kg$ com o FA. É importante ressaltar que a alta complexidade deste problema, que envolve a otimização de forma além da otimização paramétrica, pode promover uma variação maior entre os resultados obtidos.

Na Figura 6.10 são apresentadas as curvas de convergência geradas por cada algoritmo neste problema. Verifica-se que o CIOA obteve convergência relativamente rápida.

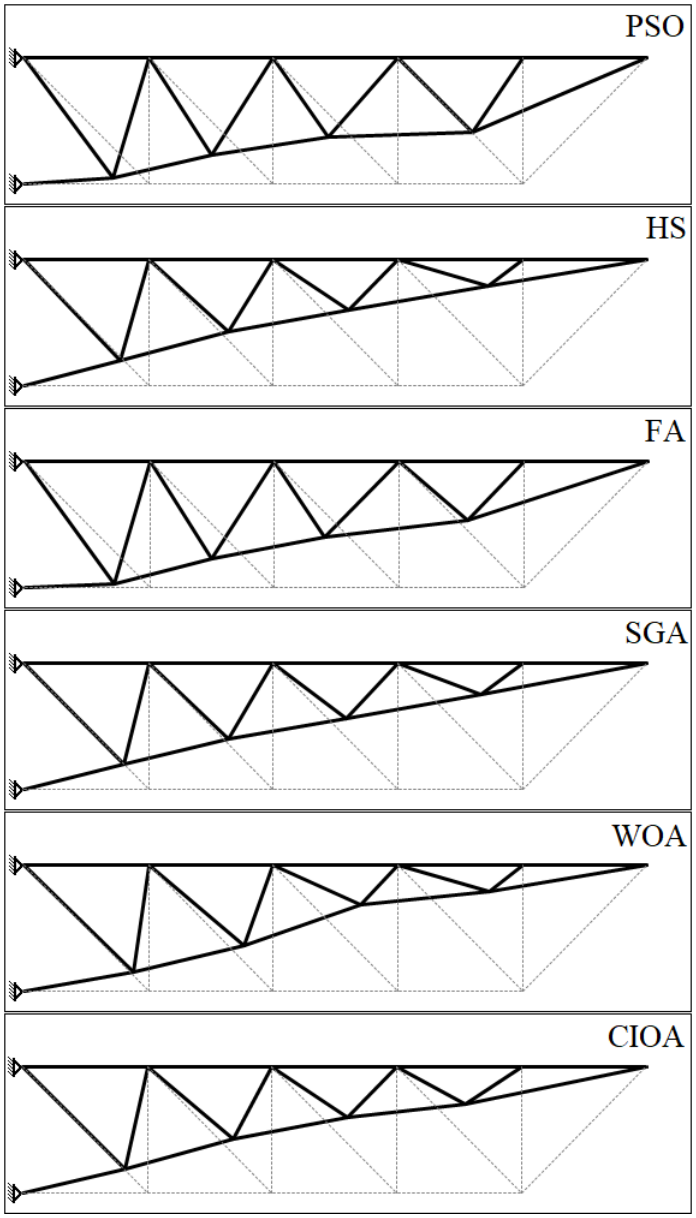
Figura 6.10 - Curvas de convergência para a treliça plana de 18 barras



Um fato que deve ser salientado é que as curvas apresentadas na Figura 6.10 parecem estar levemente deslocadas à direita no eixo X, isso ocorre devido ao fato dos resultados das primeiras iterações violarem algumas restrições e, por isso, sofrerem penalidades.

Na Figura 6.11 é apresentada a configuração da treliça obtida após o processo de otimização realizado por cada um dos algoritmos meta-heurísticos estudados neste trabalho.

Figura 6.11 – Configuração da treliça plana de 18 barras após a otimização de forma



Na Tabela 6.22 é apresentada uma análise estatística considerando as dez execuções de cada algoritmo, também é exposto o tempo computacional médio para cada execução. Verifica-se que o algoritmo CIOA apresentou desempenhos intermediários quando comparado aos demais

algoritmos. Salienta-se também que todos os algoritmos geraram um desvio padrão relativamente alto, isto é, os resultados de cada execução diferenciaram-se de forma considerável entre si, devido à alta complexidade deste problema de otimização.

Tabela 6.22 – Análise estatística e de tempo computacional para a treliça plana de 18 barras

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	2244,6	2233,6	2268,9	**2092,4**	2137,4	2146,3
Desvio padrão (kg)	64,834	106,119	98,922	**33,869**	54,174	38,276
Coef. de variação (%)	2,888	4,751	4,360	**1,619**	2,535	1,783
Tempo (s)	1029,45	1008,31	964,63	**855,12**	952,57	943,09

No Quadro 6.5 é apresentado o ranking, em ordem crescente, dos resultados desta seção.

Quadro 6.5 – Ranking para a treliça plana de 18 barras

Parâmetro	Ranking
Massa mínima	[SGA] < [**CIOA**] < [WOA] < [HS] < [FA] < [PSO]
Massa média	[SGA] < [WOA] < [**CIOA**] < [HS] < [PSO] < [FA]
Desvio padrão	[SGA] < [**CIOA**] < [WOA] < [PSO] < [FA] < [HS]
Coeficiente de variação	[SGA] < [**CIOA**] < [WOA] < [PSO] < [FA] < [HS]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [FA] < [HS] < [PSO]

Observa-se que o CIOA apresentou a segunda melhor performance em quatro das cinco análises comparativas realizadas neste problema de otimização.

Salienta-se que nenhuma restrição foi violada durante o processo de otimização. Como todas as barras possuem o mesmo limite de tração e de compressão, apresenta-se na Tabela 6.23 as tensões obtidas ao final do processo de otimização na barra mais tracionada e na barra mais comprimida da estrutura.

Tabela 6.23 – Tensões (MPa) nas barras com maiores esforços após a otimização da treliça plana de 18 barras

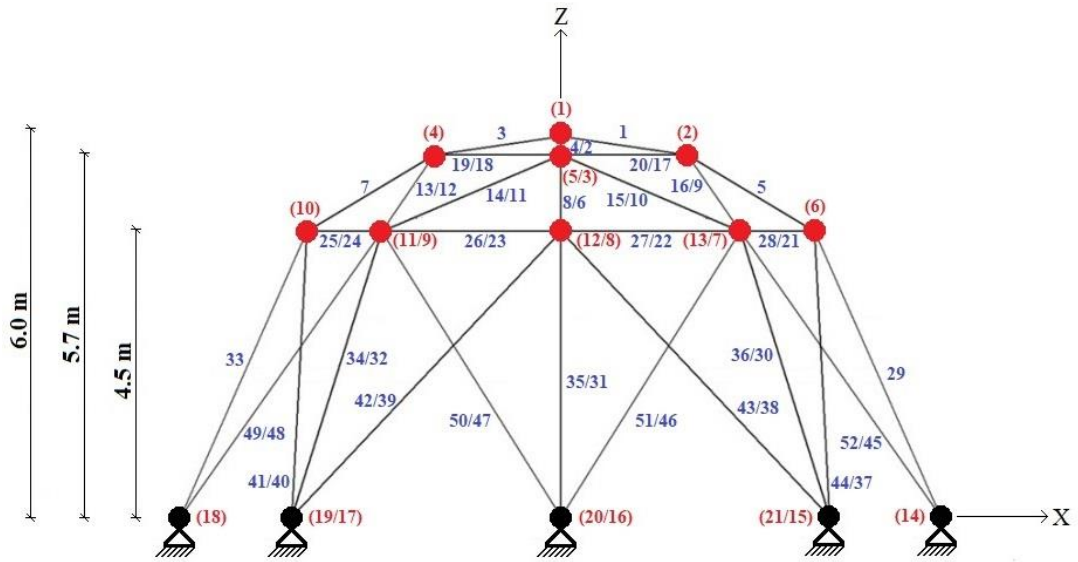
Tensão (MPa) na barra mais tracionada							
Grupo	Barra	PSO	HS	FA	SGA	WOA	CIOA
1	16	137,632	137,529	**137,895**	**137,895**	**137,895**	**137,895**
Tensão (MPa) na barra mais comprimida							
Grupo	Barra	PSO	HS	FA	SGA	WOA	CIOA
2	18	-133,602	-111,942	-134,310	-118,264	-109,746	-114,652

Verifica-se na Tabela 6.23 que nos algoritmos FA, SGA, WOA e CIOA a restrição de tensão da Barra 16 tornou-se ativa ao final do processo de otimização.

6.6 OTIMIZAÇÃO PARAMÉTRICA E DE FORMA DE TRELIÇA ESPACIAL DE 52 BARRAS SUJEITA A RESTRIÇÕES DE FREQUÊNCIA NATURAL

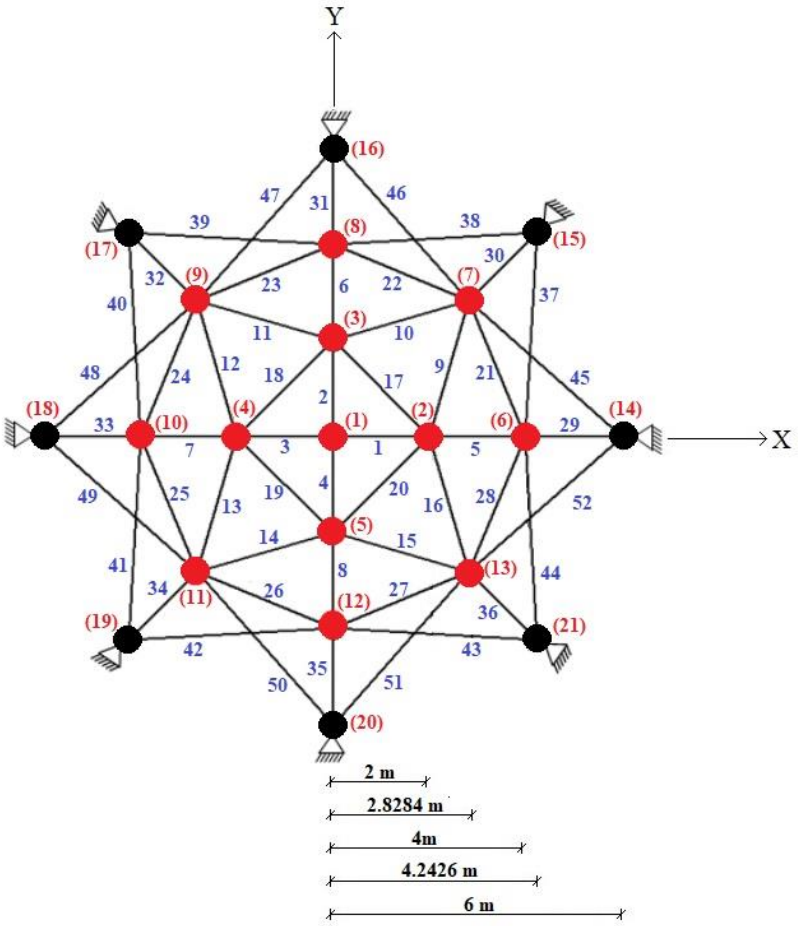
Neste problema é realizada a otimização paramétrica e de forma da treliça espacial de 52 barras apresentada nas Figuras 6.12 e 6.13. As variáveis de projeto são contínuas e representam as seções transversais das barras que compõem a estrutura e as coordenadas de alguns dos nós. A treliça é comporta de aço e apresenta, como propriedades, módulo de Young $E = 210\ GPa$ e massa específica $\rho = 7800\ kg/m^3$. O intervalo das variáveis de projeto referente às áreas é de $1\ cm^2 \leq x_i \leq 10\ cm^2$.

Figura 6.12 – Configuração inicial da treliça espacial de 52 barras (vista lateral)



Fonte: Baseado em Borges, 2013

Figura 6.13 – Configuração inicial da treliça espacial de 52 barras (vista superior)



Fonte: Baseado em Borges, 2013

Em cada um dos nós livres (nós de 1 a 13), é fixada uma massa não-estrutural concentrada de 50 kg. Para a otimização de forma, estes mesmos nós podem se mover até $\pm$ 2 m em qualquer direção X, Y e Z. As restrições de frequência natural da estrutura, que devem ser atendidas simultaneamente, são $f_1 \leq 15{,}9155\ Hz$ e $f_2 \geq 28{,}6479\ Hz$.

Para manter a simetria da estrutura, os 52 membros são divididos em oito grupos, conforme a Tabela 6.24, de modo que todas as barras de um mesmo grupo deverão possuir a mesma área ao final da otimização. Da mesma forma, as coordenadas dos nós que podem se mover são agrupadas em 5 variáveis de projetos que são apresentadas na Tabela 6.25

Tabela 6.24 – Detalhe do agrupamento de barras para a treliça espacial de 52 barras

Grupo	Membros
1	1-4
2	5-8
3	9-16
4	17-20
5	21-28
6	29-36
7	37-44
8	45-52

Tabela 6.25 – Variáveis de projeto de otimização de forma

Variável	Coordenadas
Z_1	Vertical no nó 1
X_2	Horizontal nos nós 2 a 5
Z_2	Vertical nos nós 2 a 5
X_3	Horizontal nos nós 6 a 13
Z_3	Vertical nos nós 6 a 13

Este problema já foi estudado por diversos autores: Gomes (2011) otimizou a estrutura utilizando o algoritmo PSO; Borges (2013) utilizou o HS e o FA para a otimização; Miguel e Fadel Miguel (2013) utilizaram os algoritmos HS, ABC e FA;

No presente trabalho foram realizadas 10 simulações com cada algoritmo, de modo que em cada simulação foi considerado um número total de 400000 avaliações. Para os algoritmos CIOA, WOA, SGA e PSO as avaliações foram compostas de 1600 iterações e 250 agentes de pesquisa; o algoritmo FA realizou a otimização com 10000 iterações e 40 agentes de pesquisa; para a resolução com o HS foram consideradas 400000 iterações. Os resultados obtidos para a seção transversal (cm²) de cada grupo de barras, para as coordenadas correspondentes à cada variável de forma (m) bem como a massa da treliça (kg) são apresentados na Tabela 6.26 e comparados com os obtidos por Borges (2013). Nota-se que, no presente trabalho, o CIOA obteve o segundo melhor resultado para a massa mínima da treliça, superando quatro dos cinco algoritmos com o qual foi comparado.

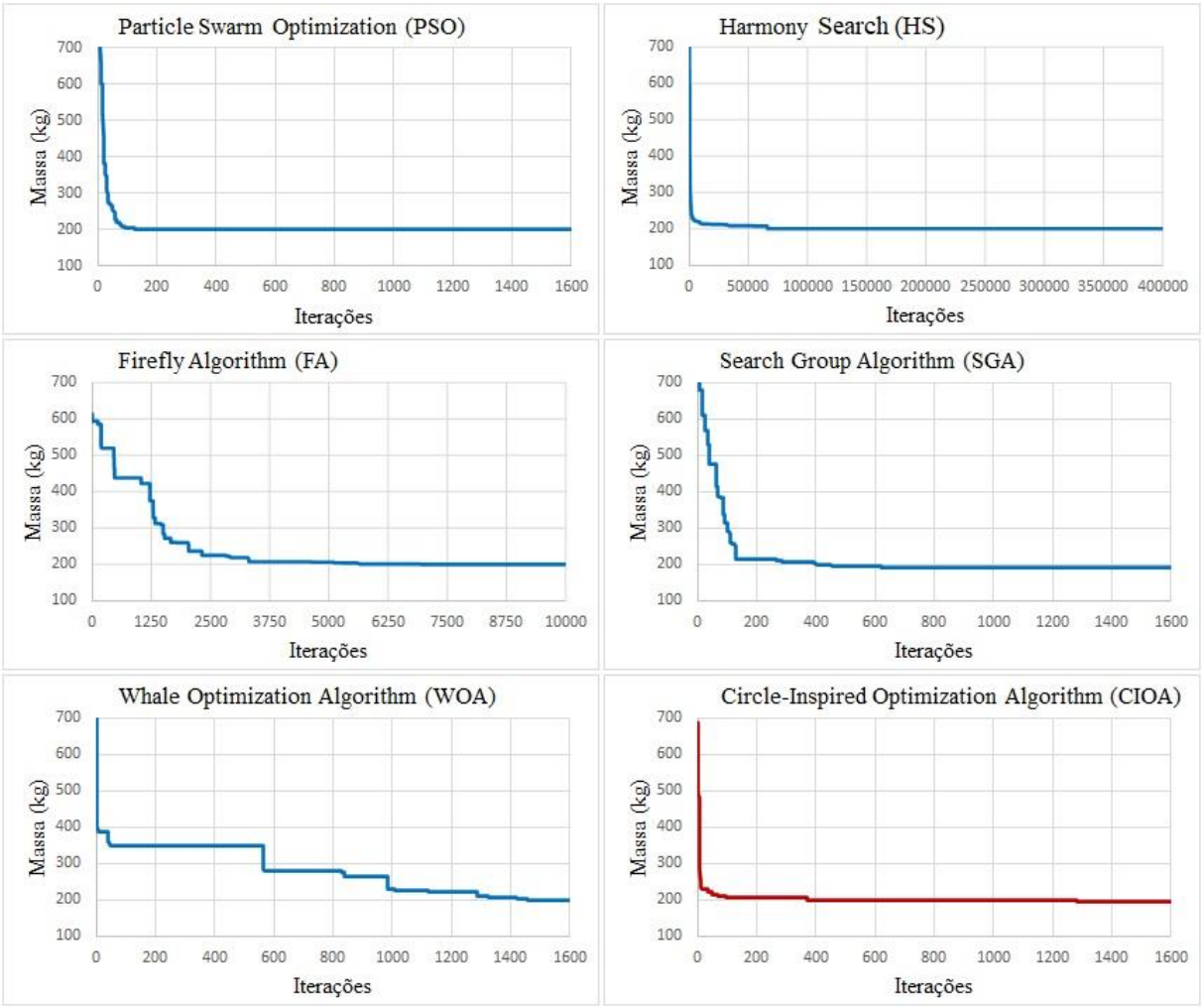
Tabela 6.26 – Projeto ótimo para a treliça espacial de 52 barras

Variável	Projeto Inicial	Borges (2013)		Presente Trabalho					
		HS	FA	PSO	HS	FA	SGA	WOA	CIOA
Z_1	6,0	4,555	5,682	6,342	5,461	5,780	6,101	5,487	5,967
X_2	2,0	1,547	2,044	2,134	2,050	2,017	2,382	2,008	2,232
Z_2	5,7	3,726	3,709	4,079	3,718	3,895	3,745	3,713	3,859
X_3	4,0	3,300	3,823	3,869	3,847	3,761	4,043	3,827	4,033
Z_3	4,5	2,734	2,520	2,711	2,500	2,694	2,500	2,503	2,520
A_1	2,0	1,090	1,000	1,000	1,015	1,000	1,000	1,001	1,000
A_2	2,0	1,488	1,298	1,073	1,241	1,092	1,029	1,241	1,127
A_3	2,0	1,751	1,280	1,129	1,575	1,163	1,197	1,157	1,328
A_4	2,0	1,468	1,507	1,246	1,552	1,452	1,365	2,175	1,363
A_5	2,0	1,498	1,389	1,525	1,362	1,470	1,433	1,404	1,445
A_6	2,0	1,186	1,000	1,000	1,006	1,000	1,000	1,033	1,000
A_7	2,0	1,618	1,651	1,443	1,766	1,610	1,570	1,533	1,659
A_8	2,0	1,718	1,364	1,771	1,495	1,586	1,385	1,510	1,323
Massa (kg)	338,69	213,24	194,59	202,13	200,11	199,54	**192,78**	198,46	196,82

Os resultados apresentados na Tabela 6.26 são similares aos obtidos por outros autores que também estudaram este problema de otimização estrutural: Gomes (2011) obteve uma massa de $228{,}38\ kg$ com o PSO; Miguel e Fadel Miguel (2013) encontraram massa mínima de $202{,}21\ kg$ com o HS, $198{,}16\ kg$ com o ABC e $193{,}99\ kg$ com o FA.

Na Figura 6.14 são apresentadas as curvas de convergência obtidas por cada algoritmo na solução deste problema. Enquanto algoritmos renomados, como o FA e o WOA, apresentaram uma taxa de convergência relativamente lenta, o CIOA foi um dos algoritmos que obteve a mais rápida taxa de convergência nas primeiras iterações, alcançando resultados próximos do ótimo ainda antes da iteração de número 200.

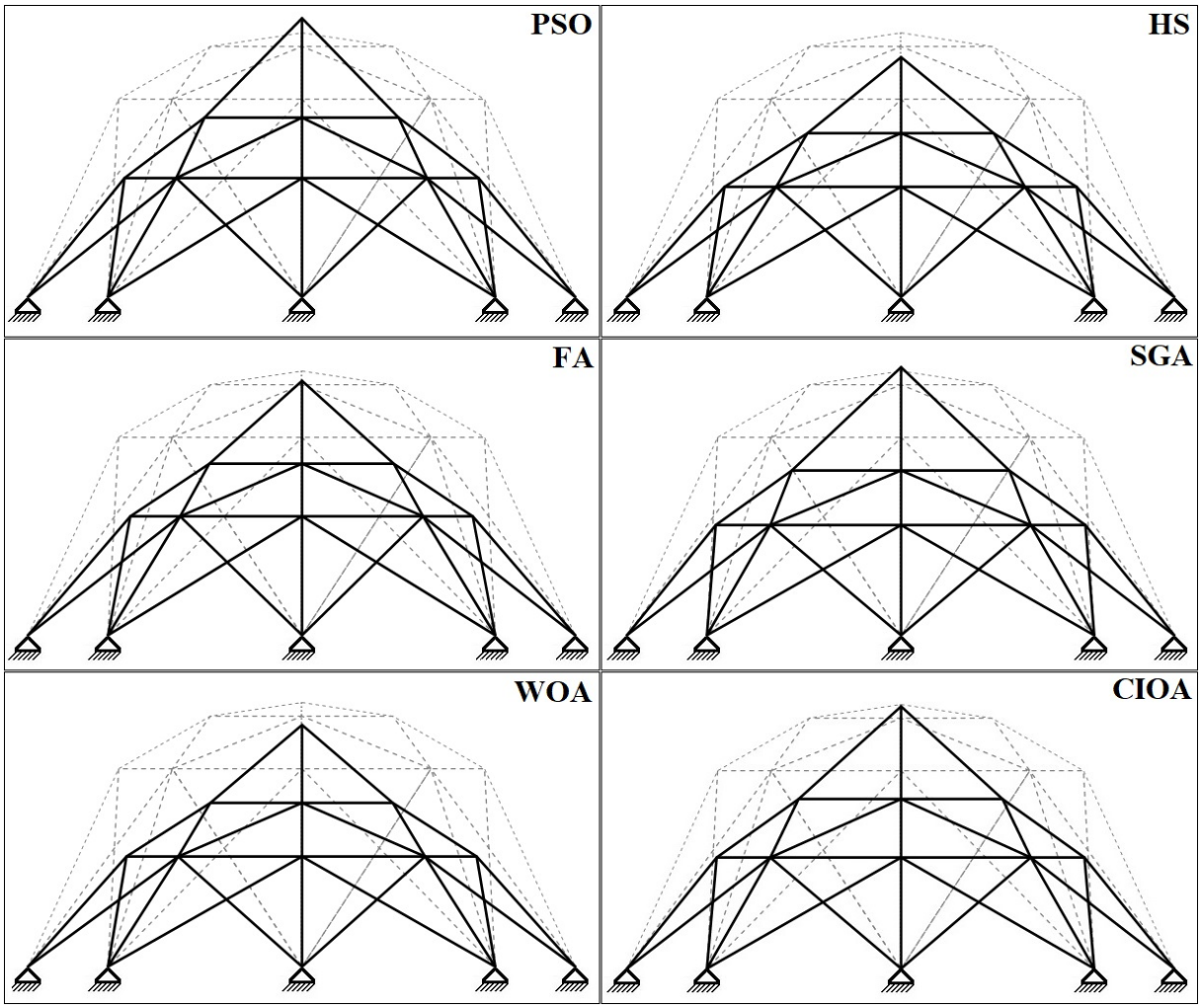
Figura 6.14 – Curvas de convergência para a treliça espacial de 52 barras



É importante ressaltar que as restrições impostas não foram violadas durante o processo de otimização, por nenhum algoritmo utilizado neste trabalho.

Na Figura 6.15 é apresentada a configuração final da treliça obtida após o processo de otimização realizado por cada um dos algoritmos estudados neste trabalho. É possível observar que as configurações apresentam similaridades evidentes, especialmente entre o CIOA e o SGA, que foram justamente os algoritmos que geraram os dois melhores projetos para este problema. Os algoritmos HS, WOA e FA, que produziram projetos de qualidade intermediária, conforme visto na Tabela 6.17, também geraram estruturas cujas configurações apresentam semelhanças notáveis entre si.

Figura 6.15 – Configuração da treliça espacial de 52 barras após a otimização de forma



Na Tabela 6.27 é apresentada a análise estatística básica deste problema e o tempo computacional necessário para cada algoritmo realizar a otimização. Conforme pode ser visto, o algoritmo CIOA apresentou o melhor desempenho em termos de massa média, desvio padrão e coeficiente de variação, considerando-se as 10 simulações realizadas com cada algoritmo.

Tabela 6.27 – Análise estatística e de tempo computacional para a treliça espacial de 52 barras

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	207,01	203,62	203,23	199,49	201,43	**199,00**
Desvio padrão (kg)	5,503	3,407	3,600	3,250	2,199	**1,276**
Coef. de variação (%)	2,658	1,673	1,722	1,629	1,092	**0,641**
Tempo (s)	1503,79	1312,42	1320,96	**1173,06**	1255,03	1216,62

No Quadro 6.6 é apresentado o ranking, em ordem crescente, referente aos resultados avaliados nesta seção.

Quadro 6.6 – Ranking para a treliça espacial de 52 barras

Parâmetro	Ranking
Massa mínima	[SGA] < [**CIOA**] < [WOA] < [FA] < [HS] < [PSO]
Massa média	[**CIOA**] < [SGA] < [WOA] < [FA] < [HS] < [PSO]
Desvio padrão	[**CIOA**] < [WOA] < [SGA] < [HS] < [FA] < [PSO]
Coeficiente de variação	[**CIOA**] < [WOA] < [SGA] < [HS] < [FA] < [PSO]
Tempo computacional	[SGA] < [**CIOA**] < [WOA] < [HS] < [FA] < [PSO]

Observa-se que o CIOA apresentou o melhor desempenho em dois dos cinco parâmetros avaliados: massa média, desvio padrão e coeficiente de variação. Nos demais parâmetros, o CIOA obteve a segunda melhor performance.

Ressalta-se que nenhuma restrição foi violada neste problema de otimização. As duas primeiras frequências naturais obtidas, com a configuração final da estrutura, são apresentadas na Tabela 5.28.

Tabela 6.28 – Primeiras frequências naturais (Hz) após a otimização da treliça espacial de 52 barras

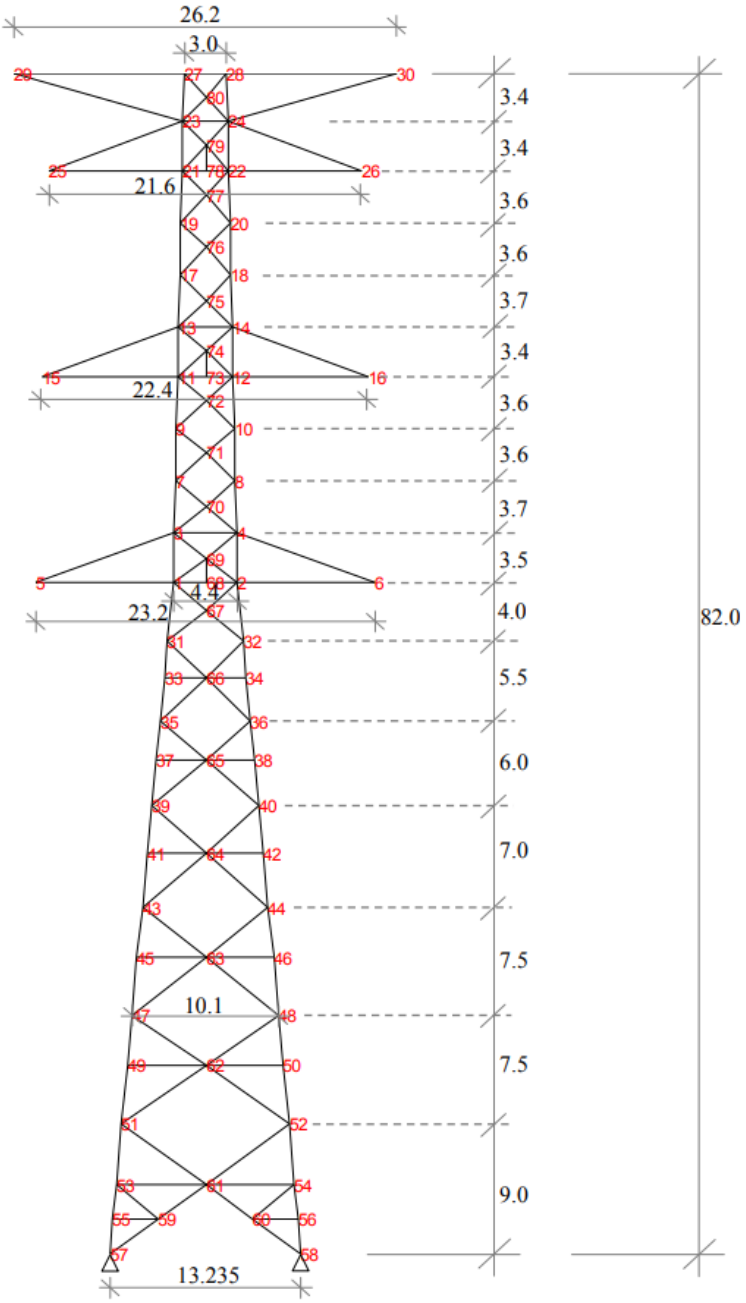
Frequências	PSO	HS	FA	SGA	WOA	CIOA
1	11,675	12,185	10,920	12,045	10,793	11,909
2	**28,648**	28,858	**28,648**	**28,648**	28,689	28,740

Verifica-se que a restrição referente à 2ª frequência natural de vibração da estrutura tornou-se ativa diante da otimização pelo PSO, FA e SGA.

7. APLICAÇÃO DO CIRCLE-INSPIRED OPTIMIZATION ALGORITHM EM UMA ESTRUTURA REALISTA

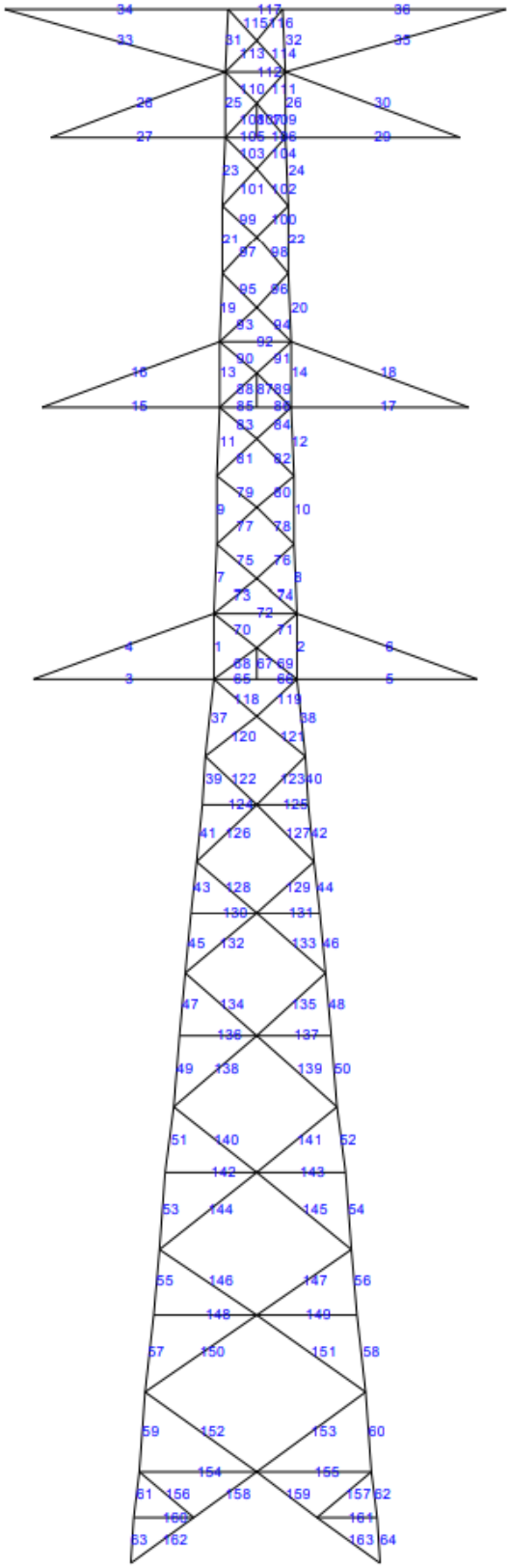
Como última avaliação, o *Circle-Inspired Optimization Algorithm* é aplicado para otimizar o projeto de uma torre de linha de transmissão de energia de aço de 82 metros de altura, construída no Japão, a qual é apresentada nas Figuras 7.1 e 7.2.

Figura 7.1 – Torre de transmissão realista (dimensões em metros) e número de nós



Fonte: Murotsu et al., 1994 apud Miguel e Fadel Miguel, 2013

Figura 7.2 – Torre de transmissão realista – número dos elementos



Fonte: Murotsu et al., 1994 apud Miguel e Fadel Miguel, 2013

Esta torre foi danificada por um tufão ocorrido no Japão em 1991. Para a otimização, a estrutura da torre é modelada como uma estrutura de treliça plana composta por 80 nós e 163 elementos. O material que constitui a estrutura é o aço, com Módulo de Young $E = 210\ GPa$ e massa específica $\rho = 7700\ kg/m^3$. As variáveis de projeto independentes são grupos de barras que deverão apresentar a mesma área de seção transversal, conforme apresentado na Tabela 7.1.

Tabela 7.1 – Agrupamento dos membros para a torre de transmissão

Grupo	Número do Elemento
1	1-8, 15-18, 27-30, 33-36
2	9-12, 65-67, 72, 85-87, 92, 105-107, 112, 117
3	13, 14, 19, 20
4	21-26, 31, 32
5	37-46
6	47-54
7	55-64
8	68-71, 73-84, 88-91, 93-104, 108-111, 113-116, 118-131, 136, 137, 142, 143, 148, 149, 156, 157
9	132-135, 138-141, 146, 147, 154, 155, 158, 159, 162, 163
10	144, 145, 150-153
11	160, 161

Fonte: Murotsu et al., 1994 apud Miguel e Fadel Miguel, 2013

Neste problema são considerados diferentes tipos de restrições, envolvendo tensão, deslocamento, flambagem e frequência natural. As restrições de **tensão** assumem valores de $\pm 490\ MPa$ em todas as barras; as restrições de **deslocamento** são de $\pm 0{,}82\ m$ nas direções X e Y de todos os nós; para as restrições de **flambagem**, o limite de tensão compressiva de flambagem de Euler é calculado pela Equação 6.1, apresentada na seção 6.6; as restrições de **frequência natural** são impostas de modo que a menor frequência natural da estrutura seja superior a $1\ Hz$. A área permitida das seções transversais é baseada em seções em 'L' e pode variar de $0{,}96\ cm^2$ até $96{,}77\ cm^2$.

Atuam na estrutura cargas que envolvem o vento e correspondem a um tufão com ventos de velocidade de $60\ m/s$, aplicadas na direção X. Também é considerado o peso próprio da torre, aplicado em forma de cargas na direção Y. Os dados referentes às cargas aplicadas diretamente

na torre e aos efeitos que as cargas aplicadas nos condutores (cabos) produzem na estrutura são apresentados na Tabela 7.2.

Tabela 7.2 – Cargas atuantes na torre de transmissão

	Nó	Carga de vento (kN)	Peso próprio (kN)
Torre	3, 4	8,1	-18,9
	9, 10	6,0	-5,9
	13, 14	7,1	-15,3
	19, 20	5,1	-3,3
	23, 24	7,1	-23,7
	31, 32	6,9	-10,6
	35, 36	6,0	0,0
	39, 40	6,9	-16,4
	43, 44	7,3	0,0
	47, 48	7,8	-22,5
	51, 52	14,1	-18,0
Cabos	5, 6	40,2	-15,2
	15, 16	42,9	-15,2
	25, 26	45,2	-15,2
	29, 30	5,7	-1,2

Fonte: Murotsu et al., 1994 apud Miguel e Fadel Miguel, 2013

Este problema já foi estudado por Miguel e Fadel Miguel 2013, onde a estrutura foi otimizada através dos algoritmos *Harmony Search* (HS), *Artificial Bee Colony* (ABC) e *Firefly Algorithm* (FA).

No presente trabalho, foram utilizadas 800 iterações e 250 agentes de pesquisa nos algoritmos CIOA, SGA, WOA e PSO; para o algoritmo FA, 5000 iterações e 40 agentes de pesquisa foram considerados; no HS foram realizadas 200000 iterações. Desta forma, todos os algoritmos solucionaram o problema utilizando um número total de 200000 avaliações. Os resultados obtidos em termos de massa mínima (kg), seção transversal (cm²) são apresentados na Tabela 7.3 e comparados tanto com o projeto original, da torre colapsada, quanto entre os algoritmos. Nota-se que na otimização através de algoritmos meta-heurísticos houve um acréscimo de massa total em relação ao projeto original. Todavia, em relação às seções transversais de cada grupo de barras, não houve um padrão, isto é, alguns grupos tiveram suas seções majoradas enquanto outros grupos tiveram as seções transversais minoradas em relação ao projeto original.

Verifica-se que o CIOA apresentou um excelente desempenho neste problema de otimização, obtendo o melhor projeto entre os algoritmos com o qual foi comparado.

Tabela 7.3 - Projeto ótimo para a torre de transmissão

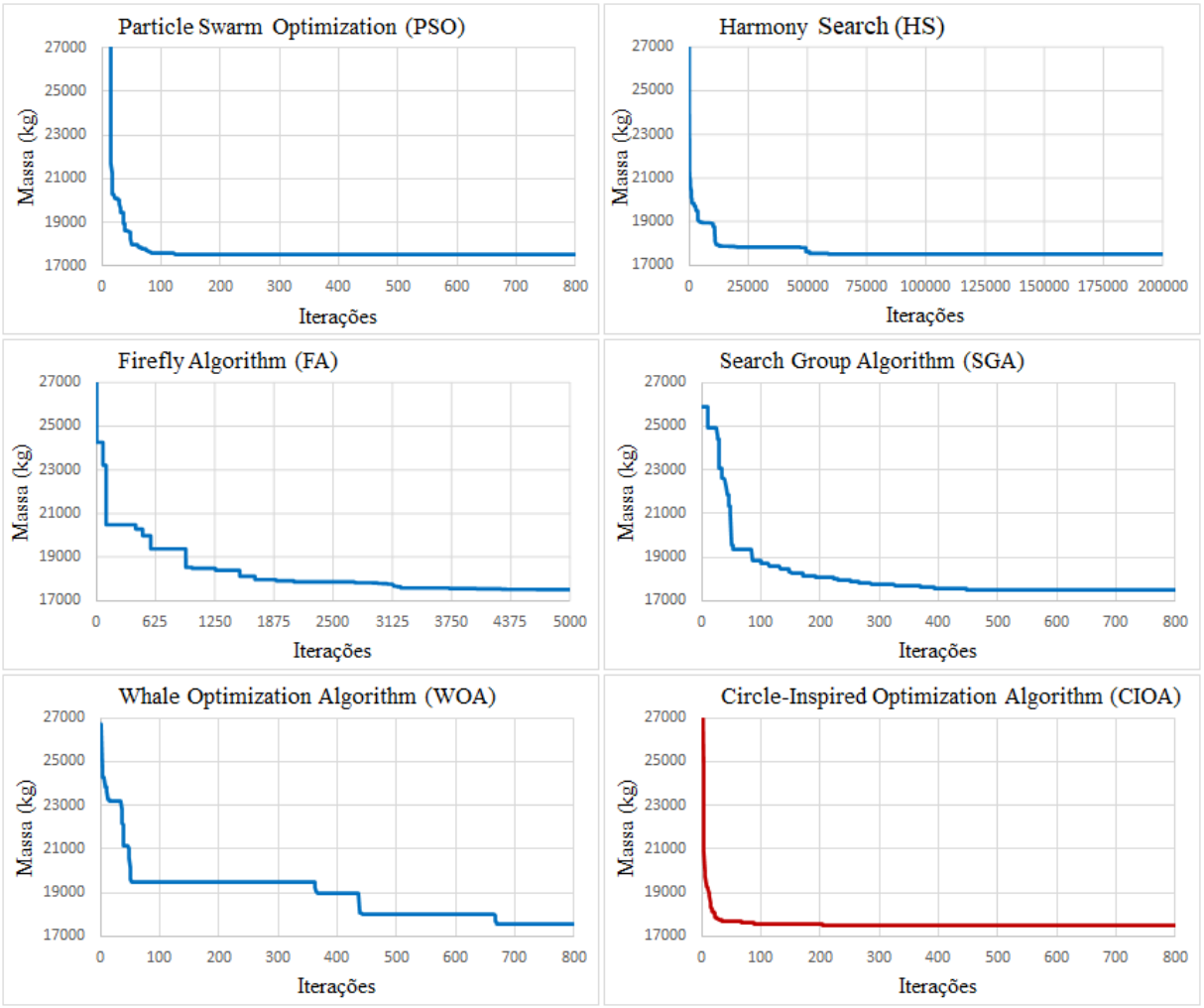
Grupo	Projeto Original	Presente Trabalho					
		PSO	HS	FA	SGA	WOA	CIOA
1	46,027	42,702	42,715	42,671	42,756	42,655	42,651
2	34,815	39,221	38,069	36,887	37,150	38,850	36,600
3	22,718	38,859	44,152	42,250	35,452	32,818	42,641
4	14,987	20,228	21,566	21,383	20,623	21,162	20,206
5	49,273	75,108	85,831	84,957	83,243	79,932	92,068
6	57,265	89,079	82,829	81,305	83,133	86,924	80,790
7	67,546	90,333	83,441	87,873	89,873	89,889	82,648
8	8,6356	11,787	11,775	11,767	11,761	11,799	11,758
9	9,8922	16,148	16,140	16,142	16,443	16,140	16,143
10	12,183	20,147	19,865	20,470	19,881	20,368	19,862
11	6,4654	0,979	0,960	1,569	1,216	1,072	0,960
Massa (kg)	14546,1	17531,4	17509,5	17506,3	17495,8	17530,9	**17485,0**

Os valores calculados para a massa ótima neste trabalho são ligeiramente melhores do que aqueles encontrados por Miguel e Fadel Miguel (2013), que obtiveram $17581{,}1\ kg$ com o HS, $17595{,}4\ kg$ com o ABC e $17557{,}5\ kg$ com o FA.

É importante ressaltar que alguns algoritmos podem ter maior dificuldade em trabalhar com esse problema devido ao elevado número de restrições que devem ser respeitadas. Estas restrições são dos mais diversos tipos: Restrições de tensão e de flambagem em todas as barras, restrições de deslocamento em todos os nós, além da restrição da primeira frequência natural de vibração da estrutura.

Na Figura 7.3 são apresentadas as curvas de convergência geradas por cada algoritmo na melhor solução obtida para este problema, onde observa-se que o CIOA apresentou rápida convergência.

Figura 7.3 – Curvas de Convergência para a torre de transmissão



Na Tabela 7.4 são apresentados os resultados da análise estatística e os valores para o tempo computacional. Apesar do baixo número de variáveis de projeto – apenas 11 – há um número muito elevado de restrições a serem atendidas neste problema, o que o torna bastante complexo. Devido a isto, o valor de desvio padrão entre execuções pode ser elevado em alguns algoritmos. O mesmo ocorre com o tempo computacional: Este problema foi o que exigiu maior tempo de operação. Verifica-se que CIOA apresentou boa performance na análise estatística para este problema.

Tabela 7.4 – Análise estatística e de tempo computacional para a torre de transmissão

	PSO	HS	FA	SGA	WOA	CIOA
Massa média (kg)	17610,5	17576,3	17655,8	17580,9	17625,2	**17521,2**
Desvio padrão (kg)	56,262	57,048	212,889	86,463	68,436	**25,420**
Coef. de variação (%)	0,319	0,325	1,206	0,492	0,388	**0,145**
Tempo (s)	3306,78	3345,16	3388,99	**3055,13**	3067,01	3259,35

No Quadro 7.1 é apresentado o ranking, em ordem crescente, referente aos parâmetros avaliados nesta seção.

Quadro 7.1 – Ranking para a torre de transmissão

Parâmetro	Ranking
Massa mínima	[**CIOA**] < [SGA] < [FA] < [HS] < [WOA] < [PSO]
Massa média	[**CIOA**] < [HS] < [SGA] < [PSO] < [WOA] < [FA]
Desvio padrão	[**CIOA**] < [PSO] < [HS] < [WOA] < [SGA] < [FA]
Coeficiente de variação	[**CIOA**] < [PSO] < [HS] < [WOA] < [SGA] < [FA]
Tempo computacional	[SGA] < [WOA] < [**CIOA**] < [PSO] < [HS] < [FA]

Observa-se que o CIOA apresentou ótimas performances, tendo o melhor desempenho em quatro dos cinco parâmetros avaliados, mostrando sua boa eficiência em problemas com elevado número de restrições, todavia, sua performance comparativa no tempo computacional foi levemente inferior do que a obtida nos problemas com um menor número de restrições, mas ainda assim apresentou o terceiro melhor tempo computacional.

Salienta-se que em todas as execuções realizadas neste trabalho, nenhuma das restrições impostas foi violada. Na Tabela 7.5 são apresentados dados de uma análise convencional da estrutura após o processo de otimização por cada algoritmo, onde são indicados: a barra mais tracionada, e o valor do esforço correspondente; a barra mais comprimida, e o valor do esforço correspondente; o nó com maior deslocamento, e o valor do referido deslocamento; a primeira frequência natural de vibração da estrutura. Restrições que se tornaram ativas são indicadas em negrito.

Tabela 7.5 – Restrições para o projeto ótimo da torre de transmissão

Tensão (MPa) na barra mais tracionada						
Barra	PSO	HS	FA	SGA	WOA	CIOA
1	241,74	241,67	241,91	241,43	242,01	242,02
Tensão (MPa) na barra mais comprimida						
Barra	PSO	HS	FA	SGA	WOA	CIOA
2	-291,99	-291,91	-292,21	-291,64	-292,31	-292,35
Deslocamento máximo da estrutura (cm)						
Nó	PSO	HS	FA	SGA	WOA	CIOA
30 (direção X)	**82,00**	**82,00**	81,89	81,99	81,89	**82,00**
Primeira frequência natural de vibração da estrutura (Hz)						
Frequência	PSO	HS	FA	SGA	WOA	CIOA
1	1,1418	1,1403	1,1435	1,1444	1,1434	1,1455

Verifica-se que, na otimização através do PSO, do HS e do CIOA, a restrição de deslocamento ficou ativa na direção X do nó 30.

8. ANÁLISE ESTATÍSTICA AVANÇADA ATRAVÉS DO TESTE DOS POSTOS SINALIZADOS DE WILCOXON

Neste capítulo são apresentados os resultados obtidos para a comparação do CIOA com cada um dos algoritmos estudados neste trabalho através de uma análise estatística utilizando o Teste dos Postos Sinalizados de Wilcoxon. Esta comparação foi realizada em todos os problemas de otimização abordados nos Capítulos 5, 6 e 7.

Detalhes da formulação do Teste de Wilcoxon foram apresentados na Seção 3.4, onde deu-se ênfase à importância da consideração deste teste em comparações de desempenho entre dois algoritmos, uma vez que o Teste e Wilcoxon leva em consideração o resultado individual de cada simulação do conjunto de simulações que compõem a amostra de resultados obtidos.

Desta forma, nos resultados apresentados ao longo deste capítulo, pode-se afirmar que o desempenho do CIOA é superior ao do algoritmo com o qual foi comparado sempre que $T^+ > T^-$. Em casos específicos onde ambos os apresentaram os mesmos resultados em todas as simulações consideradas, ocorre o que Derrac et al. (2011) e Civicioglu (2013) chamam de hipótese nula, ou seja, não é possível definir qual dos algoritmos apresenta o melhor desempenho.

8.1 COMPARAÇÃO ENTRE CIOA E PSO

Na Tabela 8.1 são apresentados os resultados obtidos na comparação através do Teste de Wilcoxon entre os algoritmos CIOA e PSO. Observa-se que o algoritmo PSO teve um desempenho levemente superior nas funções *benchmark*, apresentando a melhor performance em 5 funções enquanto o CIOA obteve o melhor desempenho em 4 funções. Todavia, o CIOA apresentou grande superioridade nos problemas de engenharia, obtendo a melhor performance em todos os problemas de otimização de treliças analisados.

Tabela 8.1 – Comparação entre CIOA e PSO usando o Teste dos Postos Sinalizados de Wilcoxon

Tipo de Otimização	Problema Analisado	T^+	T^-	Melhor Desempenho
Otimização de Funções	f_1	0	1275	PSO
	f_2	643	632	CIOA
	f_3	0	1275	PSO
	f_4	38	1237	PSO
	f_5	1275	0	CIOA
	f_6	1275	0	CIOA
	f_7	0	1275	PSO
	f_8	1275	0	CIOA
	f_9	0	0	Indefinido
	f_{10}	0	0	Indefinido
	f_{11}	0	0	Indefinido
	f_{12}	0	0	Indefinido
	f_{13}	0	0	Indefinido
	f_{14}	0	1275	PSO
	f_{15}	0	0	Indefinido
Otimização de Treliças	10 barras (caso 1)	55	0	CIOA
	10 barras (caso 2)	37	18	CIOA
	17 barras	53	2	CIOA
	25 barras	55	0	CIOA
	18 barras	52	3	CIOA
	52 barras	55	0	CIOA
	Torre de transmissão	55	0	CIOA

Considerando-se a totalidade dos problemas estudados neste trabalho: funções e treliças. O CIOA obteve o melhor desempenho sobre o PSO em onze problemas, o PSO apresentou melhor performance em cinco problemas. Em seis problemas não foi possível definir qual algoritmo é mais eficiente, já que obtiveram o mesmo desempenho em todas as simulações.

8.2 COMPARAÇÃO ENTRE CIOA E HS

Os resultados obtidos na comparação entre os algoritmos CIOA e HS através do Teste de Wilcoxon são apresentados na Tabela 8.2. Verifica-se que o algoritmo HS obteve superioridade na otimização das funções *benchmark* enquanto o CIOA apresentou melhor desempenho nos problemas de otimização estrutural.

Tabela 8.2 – Comparação entre CIOA e HS usando o Teste dos Postos Sinalizados de Wilcoxon

Tipo de Otimização	Problema Analisado	T^+	T^-	Melhor Desempenho
Otimização de Funções	f_1	918,5	356,5	CIOA
	f_2	901	374	CIOA
	f_3	0	1275	HS
	f_4	47.5	1227.5	HS
	f_5	0	1275	HS
	f_6	1	1274	HS
	f_7	0	1275	HS
	f_8	1275	0	CIOA
	f_9	0	0	Indefinido
	f_{10}	0	0	Indefinido
	f_{11}	0	0	Indefinido
	f_{12}	0	0	Indefinido
	f_{13}	0	0	Indefinido
	f_{14}	0	1275	HS
	f_{15}	0	0	Indefinido
Otimização de Treliças	10 barras (caso 1)	55	0	CIOA
	10 barras (caso 2)	37	18	CIOA
	17 barras	37	18	CIOA
	25 barras	55	0	CIOA
	18 barras	45	10	CIOA
	52 barras	55	0	CIOA
	Torre de transmissão	55	0	CIOA

Considerando-se todos os 22 problemas de otimização na comparação entre CIOA e HS, observa-se que o CIOA teve melhor desempenho em dez deles, enquanto o HS se sobressaiu em seis. Ambos os algoritmos apresentaram o mesmo comportamento em seis problemas, não sendo possível definir o melhor algoritmo nestes problemas específicos.

8.3 COMPARAÇÃO ENTRE CIOA E FA

Na Tabela 8.3 são apresentados os resultados obtidos através do Teste dos Postos Sinalizados de Wilcoxon na comparação entre os algoritmos CIOA e FA. Observa-se ampla superioridade do algoritmo CIOA.

Tabela 8.3 – Comparação entre CIOA e FA usando o Teste dos Postos Sinalizados de Wilcoxon

Tipo de Otimização	Problema Analisado	T^+	T^-	Melhor Desempenho
Otimização de Funções	f_1	1275	0	CIOA
	f_2	1273	2	CIOA
	f_3	1275	0	CIOA
	f_4	1275	0	CIOA
	f_5	1220	55	CIOA
	f_6	1275	0	CIOA
	f_7	1275	0	CIOA
	f_8	1275	0	CIOA
	f_9	0	0	Indefinido
	f_{10}	0	0	Indefinido
	f_{11}	0	0	Indefinido
	f_{12}	1275	0	CIOA
	f_{13}	1275	0	CIOA
	f_{14}	1275	0	CIOA
	f_{15}	0	0	Indefinido
Otimização de Treliças	10 barras (caso 1)	55	0	CIOA
	10 barras (caso 2)	28	27	CIOA
	17 barras	14	41	FA
	25 barras	55	0	CIOA
	18 barras	53	2	CIOA
	52 barras	53	2	CIOA
	Torre de transmissão	49	6	CIOA

Levando-se em consideração todos os problemas de otimização estudados no presente trabalho, verifica-se que o CIOA apresentou o melhor desempenho em dezessete problemas, enquanto o FA apenas em um. Em quatro problemas, ambos algoritmos apresentaram o mesmo desempenho.

8.4 COMPARAÇÃO ENTRE CIOA E SGA

Na Tabela 8.4 são apresentados os resultados obtidos no Teste dos Postos Sinalizados de Wilcoxon quando se comparam os algoritmos CIOA e SGA. Observa-se evidente superioridade do CIOA na otimização de funções *benchmark*, todavia, na maior parte nos problemas de otimização estrutural, o algoritmo SGA apresentou melhor desempenho.

Tabela 8.4 – Comparação entre CIOA e SGA usando o Teste dos Postos Sinalizados de Wilcoxon

Tipo de Otimização	Problema Analisado	T^+	T^-	Melhor Desempenho
Otimização de Funções	f_1	1275	0	CIOA
	f_2	749	526	CIOA
	f_3	1275	0	CIOA
	f_4	1275	0	CIOA
	f_5	819	456	CIOA
	f_6	1182	93	CIOA
	f_7	1275	0	CIOA
	f_8	1275	0	CIOA
	f_9	0	0	Indefinido
	f_{10}	0	0	Indefinido
	f_{11}	0	0	Indefinido
	f_{12}	1275	0	CIOA
	f_{13}	1275	0	CIOA
	f_{14}	1275	0	CIOA
	f_{15}	0	0	Indefinido
Otimização de Treliças	10 barras (caso 1)	0	55	SGA
	10 barras (caso 2)	0	55	SGA
	17 barras	0	55	SGA
	25 barras	0	55	SGA
	18 barras	5	20	SGA
	52 barras	35	20	CIOA
	Torre de transmissão	44	11	CIOA

Ao se tomar como referência a totalidade de problemas de otimização analisados nesta dissertação, ou seja, otimização de funções e otimização de treliças, verifica-se que o CIOA apresenta melhor desempenho em treze problemas, enquanto o SGA se mostrou mais eficiente em cinco problemas. Em quatro problemas, ambos algoritmos apresentaram o mesmo desempenho.

8.5 COMPARAÇÃO ENTRE CIOA E WOA

Na Tabela 8.5 são mostrados os resultados entre a comparação dos algoritmos CIOA e WOA, através de uma análise estatística avançada utilizando-se o Teste dos Postos Sinalizados de Wilcoxon. Na otimização de funções *benchmark*, observa-se ligeira superioridade do algoritmo WOA enquanto que, na otimização de treliças o algoritmo CIOA obteve o melhor desempenho.

Tabela 8.5 – Comparação entre CIOA e WOA usando o Teste dos Postos Sinalizados de Wilcoxon

Tipo de Otimização	Problema Analisado	T^+	T^-	Melhor Desempenho
Otimização de Funções	f_1	0	1275	WOA
	f_2	754	521	CIOA
	f_3	0	1275	WOA
	f_4	19	1206	WOA
	f_5	441	834	WOA
	f_6	1044	231	CIOA
	f_7	0	1275	WOA
	f_8	1275	0	CIOA
	f_9	0	0	Indefinido
	f_{10}	0	0	Indefinido
	f_{11}	0	0	Indefinido
	f_{12}	1128	0	CIOA
	f_{13}	1275	0	CIOA
	f_{14}	0	1275	WOA
	f_{15}	0	0	Indefinido
Otimização de Treliças	10 barras (caso 1)	55	0	CIOA
	10 barras (caso 2)	48	7	CIOA
	17 barras	34	21	CIOA
	25 barras	55	0	CIOA
	18 barras	26	29	WOA
	52 barras	52	3	CIOA
	Torre de transmissão	55	0	CIOA

Ao levar em consideração todos os problemas, observa-se melhor performance do CIOA em onze problemas, enquanto o WOA se mostrou mais eficiente em sete problemas. Em quatro problemas não foi possível determinar qual dos dois algoritmos foi o mais eficiente.

9. CONCLUSÕES

Neste capítulo constam as principais conclusões que podem ser elencadas através dos resultados produzidos neste trabalho, bem como sugestões para pesquisas futuras.

O algoritmo meta-heurístico de otimização produzido nesta pesquisa, denominado *Circle-Inspired Optimization Algorithm* (CIOA), se mostrou válido, uma vez que resolveu com sucesso os problemas que são amplamente utilizados na validação de novos algoritmos: a minimização de funções *benchmark*. Em vários destes problemas, o CIOA conseguiu obter resultados melhores do que aqueles produzidos por algoritmos já consagrados.

O CIOA também se mostrou altamente competitivo nos problemas aplicados à Engenharia Estrutural e que envolvem um número muito elevado de restrições que não podem ser violadas. Em grande parte destes problemas, o CIOA produziu resultados que o apontaram como mais eficiente do que a maioria dos algoritmos famosos com o qual foi comparado.

Por fim, uma análise estatística avançada através do Teste dos Postos Sinalizados de Wilcoxon, apontou que, de modo geral, o CIOA é capaz de superar o desempenho de cada um dos cinco algoritmos com o qual foi comparado, quando se considera a totalidade dos problemas estudados no presente trabalho.

Além da eficiência e do rápido tempo computacional de operação, evidenciado em todos os problemas no qual o CIOA foi utilizado, o algoritmo apresenta ainda como grande vantagem o reduzido número de parâmetros que precisam ser definidos pelo usuário, a saber, apenas dois: O ângulo θ e o parâmetro $Glob_{It}$. Além disso, é possível resolver um amplo leque de problemas mantendo-se um valor constante para estes parâmetros, independente das características do problema a ser resolvido, conforme foi feito neste trabalho, onde a otimização de todas as funções e de todas as estruturas foi realizada utilizando-se valores fixos de $\theta = 17°$ e $Glob_{It} = 0{,}85$.

Salienta-se ainda, como mais um ponto positivo, a fácil implementação do CIOA, cujo código computacional e formulações são extremamente simples e de fácil compreensão.

Alguns pontos que podem ainda ser aprimorados, ou que não foram estudados detalhadamente neste trabalho constam na Seção 8.1, em forma de sugestão para futuras pesquisas.

9.1 SUGESTÕES PARA FUTURAS PESQUISAS

Nesta seção, são apresentadas algumas sugestões para pesquisas que visam complementar ou aprimorar o presente trabalho:

a) Implementar uma nova versão do *Circle-Inspired Optimization Algorithm* utilizando programação em paralelo, de forma a reduzir ainda mais o tempo de execução;

b) Utilizar o CIOA em problemas de otimização multiobjetivo, fazendo as eventuais alterações necessárias para que estes problemas possam ser atendidos pelo algoritmo;

c) Utilizar o CIOA em problemas de otimização estrutural topológica;

d) Utilizar o CIOA em problemas de otimização estrutural com variáveis de projeto discretas, torando a análise mais realista;

e) Buscar possibilidades de eliminar o parâmetro $Glob_{It}$, fazendo com que o próprio algoritmo tenha autonomia e inteligência em definir qual é o melhor momento, ao longo das iterações, para dedicar-se exclusivamente à pesquisa local.

REFERÊNCIAS

ARORA, S.; SINGH, S. Butterfly optimization algorithm: a novel approach for global optimization. **Soft Computing.** v. 23, n. 3, p. 715-734, feb 2019.

BEKDAS, G.; NIGDELI, S. M.; YANG, X-S. Sizing optimization of truss structures using flower pollination algorithm. **Applied Soft Computing.** v. 37, p. 322-331, dec. 2015.

BORGES, A. A. **Otimização de forma e paramétrica de estruturas treliçadas através dos métodos meta-heurísticos Harmony Search e Firefly Algorithm.** 2013. Dissertação (Mestrado em Engenharia Mecânica) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

CIVICIOGLU, P. Backtracking Search Optimization Algorithm for numerical optimization problems. **Applied Mathematics and Computation.** v. 219, p. 8121-8144, apr. 2013.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation.** v. 1, p. 3-18, mar. 2011.

GEEM, Z. W.; KIM, J. H.; LOGANATHAN, G. V. A New Heuristic Optimization Algorithm: Harmony Search. **Simulation,** United States of America, v. 76, n. 2, p. 60-68, feb. 2001.

GANDOMI, A. H.; ALAVI, A. H. Krill herd: A new bio-inspired optimization algorithm. **Communications in Nonlinear Science and Numerical Simulation.** v. 17, p. 4831-4845, dec 2012.

GOMES, H. M. Truss optimization with dynamic constraints using a particle swarm algorithm. **Expert Systems with Applications.** v. 38, p. 957-968, jan. 2011.

GONÇALVES, M. S.; LOPEZ, R. H.; FADEL MIGUEL, L. F. Search group algorithm: A new metaheuristic method for the optimization of truss structures. **Computers and Structures.** v. 153, p. 165-184, june. 2015.

KARABOGA, D. An idea based on honey bee swarm for numerical optimization. Kayseri: Erciyes University, 2005, 10.

KARABOGA, D.; BASTURK, B. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. *In*: INTERNATIONAL FUZZY SYSTEMS ASSOCIATION WORLD CONGRESS, 12., 2007, Cancun. **Proceedings […]** Springer, 2007. p. 789-798.

KAVEH, A.; ZOLGHADR, A.; Meta-heuristic methods for optimization of truss structures with vibration frequency constraints. **Int. J. Acta Mechanica.** v. 229, p. 3971-3992, sept 2018.

KENNEDY J.; EBERHSRT R. Particle swarm optimization. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS. 1995. Perth. **Proceedings [...]** IEEE, 1995, p. 1942-1948.

LEE, K, S; GEEM, Z, W. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. **Computer methods in applied mechanics and engineering.** v. 194, p. 3902-3933, sept 2005.

LEE, K. S.; GEEM, Z. W. A new structural optimization method based on the Harmony Search Algorithm. **Computers and Structures.** v. 82, p. 781-798, jan 2004.

LOUZAZNI, M.; KHOUYA, A.; AMECHNOUE, K.; GANDELLI, A.; MUSSETTA, M; CRACIUNESCU, A. Metaheuristic Algorithm for Photovoltaic Parameters: Comparative Study and Prediction with a Firefly Algorithm. **Applied Sciences.** v. 8, p. 1-22, feb. 2018.

MARÇON, L. P. N. **Otimização de uma rede de trocadores de calor associados à produção de óleo vegetal.** 2018. Monografia (Graduação em Engenharia Química) – Faculdade de Engenharia Química, Universidade Federal de Uberlândia, Uberlândia, 2018.

MIGUEL, L. F. F.; FADEL MIGUEL, L. F. Assessment of modern metaheuristic algorithms – HS, ABC and FA – in shape and size optimization of structures with different types of constraints. **Int. J. Metaheuristics,** v. 2, n. 1, p. 256-293, sept. 2013.

MIRJALILI, S.; LEWIS A. The Whale Optimization Algorithm. **Advances in Engineering Software,** v. 95, p. 51-67, may. 2016.

MIRJALILI, S.; MIRJALILI, S. B.; LEWIS, A. Grey Wolf Optimizer. **Advances in Engineering Software,** v. 69, p. 46-61, mar. 2014.

RAO, S. S. **Engineering Optimization:** Theory and Practice. 4ª ed. Hoboken: John Wiley & Sons, 2009.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. GSA: A gravitational search algorithm. **Information Sciences,** v. 179, n. 13, p. 2232-2248, july. 2009.

SERAPIÃO, A, B, S. Fundamentos de otimização por inteligência de enxames: uma visão geral. **Revista controle e automação.** v. 20, n. 3, p. 271-304, jul. 2009.

SOUZA, R. P. **Otimização de treliças com restrições de falha combinando técnicas de programação de algoritmos contínuos e discretos.** 2009. Dissertação (Mestrado em Engenharia Mecânica) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2009.

STORN, R.; PRICE, K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. **Journal of Global Optimization.** v. 11, n. 4, p. 341-359, dec. 1997.

YANG, X-S. A new metaheuristic bat-inspired algorithm. *In:* WORKSHOP ON NATURE INSPIRED COOPERATIVE STRATEGIES FOR OPTIMIZATION, 2010a, Granada. **Proceedings […]** Berlin: Springer, 2010a, p. 65-74.

YANG, X-S. Firefly algorithms for multimodal optimization. *In*: INTERNATIONAL SYMPOSIUM ON STOCHASTIC ALGORITHMS, 5., 2009a, Sapporo. **Proceedings […]** Berlin: Springer, 2009a, p. 169-178.

YANG, X-S. Firefly Algorithm, Stochastic Test Functions and Design Optimization. **Int. J. Bio-Inspired Computation**, v. 2, n. 2, p. 78-84, mar. 2010b.

YANG, X-S. Flower Pollination Algorithm for Global Optimization. *In:* UNCONVENTIONAL COMPUTATION AND NATURAL COMPUTATION, 2012, Orléan. **Proceedings […]** Berlin: Springer, 2012, p. 240-249.

YANG, X-S. Harmony Search as a Metaheuristic Algorithm. *In*: GEEM, Z. W. **Music-Inspired Harmony Search Algorithm**: Theory and Applications. In: Studies in Computational Intelligence. Rockville: Springer, 2009b. v. 191, cap. 1, p. 1-14.

YANG, X-S. **Nature-inspired Metaheuristic Algorithms**. 2ª e d. Frome: Luniver Press, 2010c.

YANG, X-S.; DEB, S. Cuckoo Search via Lévy Flights *In*: WORLD CONGRESS ON NATURE AND BIOLOGICALLY INSPIRED COMPUTING, 2009, Coimbatore. **Proceedings […]** IEEE, 2009, p. 210-214.