UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR VALLS DA COSTA SILVA

ACOPLAMENTO DE ALGORITMOS GENÉTICOS E DE INTELIGÊNCIA DE ENXAME PARA A SOLUÇÃO DE PROBLEMAS CONTÍNUOS E COMBINATÓRIOS

RIO DE JANEIRO
2025

ARTHUR VALLS DA COSTA SILVA

ACOPLAMENTO DE ALGORITMOS GENÉTICOS E DE INTELIGÊNCIA DE ENXAME PARA A SOLUÇÃO DE PROBLEMAS CONTÍNUOS E COMBINATÓRIOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Carolina Gil Marcelino

RIO DE JANEIRO

2025

CIP - Catalogação na Publicação

S586a

Silva, Arthur Valls da Costa
Acoplamento de algoritmos genéticos e de inteligência de enxame para a solução de problemas contínuos e combinatórios / Arthur Valls da Costa Silva. -- Rio de Janeiro, 2025.
51 f.

Orientadora: Carolina Gil Marcelino.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2025.

1. Otimização. 2. Meta-heurísticas. 3. Algoritmos híbridos. 4. Telecomunicações. 5. Cobertura de torres. I. Marcelino, Carolina Gil, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

ARTHUR VALLS DA COSTA SILVA

ACOPLAMENTO DE ALGORITMOS GENÉTICOS E DE INTELIGÊNCIA DE ENXAME PARA A SOLUÇÃO DE PROBLEMAS CONTÍNUOS E COMBINATÓRIOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 06 de fevereiro de 2025

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
CAROLINA GIL MARCELINO
Data: 11/02/2025 09:29:34-0300
Verifique em https://validar.iti.gov.br

Profa. Carolina Gil Marcelino
D.Sc. (IC/UFRJ)

gov.br Documento assinado digitalmente
SILVANA ROSSETTO
Data: 11/02/2025 11:04:25-0300
Verifique em https://validar.iti.gov.br

Profa. Silvana Rossetto
D.Sc. (IC/UFRJ)

gov.br Documento assinado digitalmente
ALAN COSTA DE SOUZA
Data: 11/02/2025 12:16:07-0300
Verifique em https://validar.iti.gov.br

Dr. Alan Costa de Souza
D.Sc. (IC/NCE/PPGI/UFRJ)

Dedico este trabalho aos meus avós: Vilma, Yolanda, Carlindo e Ricardo.

AGRADECIMENTOS

Quero expressar minha profunda gratidão à minha parceira, Julia, que desde o início da faculdade esteve ao meu lado, literalmente, pois nosso encontro aconteceu em uma aula de Cálculo 1 aqui na universidade. Desde então, ela nunca se afastou e foi, sem dúvida, uma das maiores forças que me impulsionaram e motivaram a concluir meu percurso acadêmico. Sempre que fiquei nervoso com provas ou apresentações, ela estava lá ao meu lado. Quando a motivação vacilava, ela era quem me erguia. Em praticamente todos os trabalhos que realizamos durante a faculdade, Julia foi minha companheira constante. Sem ela, certamente, não teria chegado tão longe. Meu agradecimento mais sincero por tudo.

Agradeço também aos meus pais, Venicio e Luciana, por sempre me incentivarem e me permitirem sonhar sem limites, garantindo que eu pudesse me dedicar exclusivamente aos meus objetivos, sem me preocupar com outras questões. Eles nunca me deixaram desistir do sonho de me tornar um cientista, neste caso, da computação. Sou profundamente grato por todo o amor, disciplina e aprendizado que me transmitiram.

Minha gratidão se estende à minha tia Cida, a quem considero uma segunda mãe, por sua constante preocupação, carinho e disponibilidade nos momentos mais difíceis. Agradeço ainda ao meu irmão Rafael, cujo apoio foi muito além de meras conversas. Ele contribuiu significativamente em apresentações e trabalhos, nos quais seu talento em design foi indispensável.

Quero também agradecer aos meus amigos de infância Rafael, Lucca, Pedro e Lucas, que me acompanham há 14 anos. Estudamos juntos desde o colégio até a faculdade, e todas as risadas, conselhos e ajudas tornaram essa caminhada muito mais leve e prazerosa.

Agradeço de coração à *Tropa de Choque*, o grupo de colegas de Ciência da Computação formado quando ingressamos na UFRJ e que se mantém inabalável até hoje. Sem as noites sem dormir estudando juntos, sem as rodas de conversa, as dicas e os ensinamentos compartilhados, provavelmente eu não estaria escrevendo este trabalho. Obrigado por tudo.

Por fim, minha gratidão à minha orientadora, Professora Carolina, que me apresentou à área das meta-heurísticas e aprofundou ainda mais meu gosto pelo estudo. Agradeço pela paciência em responder às minhas dúvidas, pelos ensinamentos, pelas correções necessárias e pela presença constante ao longo desse processo de pesquisa acadêmica.

"Do you see how infinite you are?"

Sekishūsai Yagyū

RESUMO

Meta-heurísticas são ferramentas essenciais para a resolução de problemas de otimização complexos, especialmente quando métodos exatos se tornam inviáveis devido ao alto custo computacional. Entre essas técnicas, os Algoritmos Genéticos (GA) e a Otimização por Enxame de Partículas (PSO) destacam-se por sua capacidade de explorar o espaço de busca e evitar a convergência prematura em mínimos locais. Neste trabalho, é proposto o GA-BBPSO, uma abordagem acoplada que integra operadores genéticos ao PSO Simplificado (BBPSO). Essa combinação tem como objetivo aprimorar a diversidade da população e acelerar a convergência, permitindo uma exploração mais eficaz das possíveis soluções. A metodologia desenvolvida foi avaliada por meio de um conjunto abrangente de funções de benchmark, evidenciando a robustez do GA-BBPSO em relação aos métodos tradicionais, tanto na qualidade das soluções quanto na redução do esforço computacional. Além disso, sua aplicação ao problema do posicionamento de torres de sinal, um desafio clássico de otimização combinatória, reforçou sua eficácia em cenários reais. Os resultados demonstram que o GA-BBPSO apresenta um potencial promissor para a resolução de problemas complexos.

Palavras-chave: otimização; meta-heurísticas; algoritmos híbridos; telecomunicações; cobertura de torres.

ABSTRACT

Metaheuristics are essential tools for solving complex optimization problems, especially when exact methods become impractical due to high computational costs. Among these techniques, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) stand out for their ability to explore the search space and avoid premature convergence to local minima. This work proposes GA-BBPSO, a hybrid approach that integrates genetic operators into Simplified PSO (BBPSO). This combination aims to enhance population diversity and accelerate convergence, allowing for a more effective exploration of possible solutions. The developed methodology was evaluated using a comprehensive set of benchmark functions, highlighting the robustness of GA-BBPSO compared to traditional methods, both in solution quality and computational effort reduction. Additionally, its application to the signal tower placement problem, a classic combinatorial optimization challenge, reinforced its effectiveness in real-world scenarios. The results demonstrate that GA-BBPSO holds promising potential for solving complex problems.

Keywords: optimization; metaheuristics; hybrid algorithms; telecommunications; tower coverage.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

PSO	Particle Swarm Optimization
GA	Genetic Algorithm
BBPSO	BareBones Particle Swarm Optimization
GA-BBPSO	Genetically Altered BareBones Particle Swarm Optimization
CR	Crossover Rate
NP	Non-deterministic Polynomial-time
DBBePSO	Dynamic Barebones Particle Swarm Optimization
ITU	International Telecommunication Union

LISTA DE SÍMBOLOS

σ	Letra grega minúscula sigma
μ	Letra grega mu
$U(0,1)$	Distribuição Uniforme no intervalo [0,1].

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

Resolver desafios de otimização de maneira exata pode ser inviável para muitas questões de grande relevância industrial e científica. Na prática, busca-se soluções razoáveis, que podem ser obtidas de maneira eficiente, por meio de algoritmos heurísticos ou meta-heurísticas. As meta-heurísticas, em particular, formam uma classe de técnicas de otimização aproximada que ganhou ampla popularidade nas últimas duas décadas. Elas estão entre as abordagens mais promissoras e eficazes, pois oferecem soluções aceitáveis em tempos viáveis para problemas complexos em áreas como telecomunicações, aprendizado de máquina, finanças, processamento de sinais, logística e transporte (TALBI, 2009).

Muitas meta-heurísticas são inspiradas em processos naturais. Exemplos incluem algoritmos evolutivos e sistemas imunológicos artificiais, oriundos da biologia; otimização por colônias de formigas, abelhas e enxames de partículas, baseados na inteligência coletiva (ciências sociais); e o recozimento simulado, fundamentado em conceitos da física (SÖRENSEN; GLOVER, 2013).

No entanto, apesar das diversas vantagens dessas técnicas, elas enfrentam uma limitação fundamental: ao serem avaliadas em diferentes tipos de problemas, sua performance média tende a ser equivalente. Essa observação, formalizada pelo **Teorema do Almoço Grátis** (WOLPERT; MACREADY, 2005), revela que enquanto um algoritmo pode apresentar resultados superiores em tarefas unimodais, outro pode se destacar em problemas multimodais. Assim, não existe um algoritmo generalista capaz de resolver de forma eficiente toda a diversidade de problemas. Como resposta a essa limitação, é comum o desenvolvimento de versões acopladas, que integram características vantajosas de diferentes métodos existentes, buscando explorar o melhor de cada abordagem.

Diante desse cenário, o presente trabalho propõe uma nova meta-heurística que faz o acoplamento de duas técnicas já conhecidas, para abordar um problema específico de otimização. A abordagem integra elementos do Algoritmo Genético (GA) (HOLLAND, 1992) e do PSO Simplificado (BBPSO) (KENNEDY, 2003).

1.2 MOTIVAÇÃO

O tráfego global de dados móveis tem aumentado substancialmente nos últimos anos. De acordo com a União Internacional de Telecomunicações (ITU), a quantidade de tráfego de dados por assinante deve alcançar aproximadamente 257 exabytes (257×10^{18} bytes) em 2030 (ITU-R, 2015). Além disso, estima-se que quase 60% da população mundial estará utilizando a Internet móvel até 2025 (ITU-R, 2015).

Esse tipo de comunicação depende de torres de sinal, que desempenham um papel fundamental na distribuição do serviço. No entanto, essas estruturas possuem custo elevado, podendo chegar a aproximadamente R$ 3.000.000 (Anatel and Teleco, 2025). Nesse sentido, expandir o acesso à Internet para o maior número de pessoas demanda um aumento da quantidade de torres. Todavia, devido ao alto investimento necessário, torna-se essencial alocar esses equipamentos de forma estratégica, minimizando gastos e maximizando o número de dispositivos dentro de suas áreas de cobertura (KASHYAP et al., 2014).

O desafio de posicionar torres de sinal pertence à classe de problemas NP-difíceis da otimização combinatória. A resolução exata desse tipo de desafio frequentemente envolve um custo computacional muito alto e, em muitos casos, pode ser inviável em tempo hábil. Por essa razão, a adoção de técnicas heurísticas ou meta-heurísticas é uma abordagem amplamente utilizada para encontrar soluções factíveis dentro de prazos aceitáveis (FERREIRA et al., 2024).

1.3 OBJETIVOS

O objetivo geral deste trabalho é propor e desenvolver uma nova meta-heurística de otimização que combine características de duas heurísticas já conhecidas, avaliando seu desempenho no problema de posicionamento de torres de sinal. Essa nova técnica, denominada GA-BBPSO, baseia-se em uma variação simplificada do conhecido algoritmo de Otimização por Enxame de Partículas (KENNEDY; EBERHART, 1995) (PSO, na sigla em inglês), chamada de Otimização por Enxame de Partículas Simplificada (KENNEDY, 2003) (BBPSO), à qual serão incorporados operadores genéticos inspirados no Algoritmo Genético (HOLLAND, 1992). A adição desses operadores visa aprimorar a eficiência da busca do BBPSO.

Após a implementação, o novo método será testado em diferentes tipos de funções de *benchmark* (SURJANOVIC; BINGHAM, 2024) (PLEVRIS; SOLORZANO, 2022) (GAVANA, 2024), de modo a avaliar seu comportamento em cenários variados. Por fim, o GA-BBPSO será aplicado ao problema de posicionamento de torres de sinal, e seus resultados serão comparados com os de outras heurísticas predecessoras.

1.3.1 Objetivos específicos

- Estudar e compreender o funcionamento das meta-heurísticas e suas aplicações em problemas de otimização.
- Analisar métodos acoplados já propostos e avaliar como eles aprimoram algoritmos originais.
- Projetar e implementar o novo método acoplado GA-BBPSO.

- Comparar o desempenho do GA-BBPSO em diferentes funções de *benchmark* com os algoritmos originais (PSO e BBPSO).
- Aplicar o GA-BBPSO ao problema de posicionamento de torres de sinal e avaliar sua eficiência em comparação com outros algoritmos.

1.4 ESTRUTURA DO TRABALHO

O trabalho foi dividido em seis capítulos. Inicialmente, o capítulo 1 apresentou o contexto, a motivação e os objetivos gerais e específicos da pesquisa. Em seguida, o capítulo 2 aborda os conceitos-chave de otimização e meta-heurísticas, detalhando os algoritmos base do estudo, como o Algoritmo Genético e o BBPSO. O capítulo 3 analisa estudos prévios relevantes, destacando suas contribuições e limitações. No capítulo 4 são apresentados o desenvolvimento, os operadores utilizados e o pseudocódigo da nova meta-heurística GA-BBPSO. Já o capítulo 5 detalha a configuração experimental, os testes realizados e a análise dos resultados comparativos. Por fim, o capítulo 6 resume as contribuições, limitações e possíveis direções para estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 OTIMIZAÇÃO

A otimização matemática busca determinar valores que minimizam ou maximizam uma função, geralmente sujeita a restrições que limitam as variáveis (NOCEDAL; WRIGHT, 1999). Na ausência de restrições, o problema pode ser formalmente definido como:

$$x^* = \arg\min_x f(x), \tag{2.1}$$

onde x^* representa o vetor que minimiza a função objetivo $f(x)$. No entanto, em aplicações reais, há restrições físicas, tecnológicas ou subjetivas, que podem ser expressas de forma geral como:

$$\begin{cases} g_i(x) \leq 0, & i = 1, \ldots, r, \\ h_j(x) = 0, & j = 1, \ldots, p. \end{cases} \tag{2.2}$$

Essa formulação representa um caso geral de problemas de otimização com restrições, onde as funções $g_i(x)$ correspondem a desigualdades que limitam o espaço de busca e as funções $h_j(x)$ representam restrições de igualdade. Em situações práticas, a escolha específica de $g_i(x)$ e $h_j(x)$ depende do contexto do problema, podendo abranger desde restrições físicas, como capacidade máxima de um recurso, até condições operacionais e normativas (TAKAHASHI, 2007).

Nesse contexto, o problema de determinar valores que satisfaçam tais condições é denominado **problema de factibilidade** (TAKAHASHI, 2007). Quando se deseja encontrar a melhor solução dentro dessas restrições, obtém-se a formulação geral da otimização prática:

$$x^* = \arg\min_x f(x), \quad \text{sujeito a: } \begin{cases} g_i(x) \leq 0, & i = 1, \ldots, r, \\ h_j(x) = 0, & j = 1, \ldots, p. \end{cases} \tag{2.3}$$

A otimização é amplamente utilizada em áreas quantitativas, como ciência da computação, engenharia, pesquisa operacional e economia. Em sua forma mais simples, consiste em maximizar ou minimizar uma função real, escolhendo os valores de entrada dentro de um conjunto permitido (Engati Team, 2025). Essa função, denominada *função objetivo* ou *função de custo*, orienta o processo de tomada de decisão.

A área de otimização pode ser dividida em diversas subáreas, dentre elas:

a) **Otimização Linear**, na qual todos os componentes, desde a função de custo até as restrições, são lineares (LUENBERGER; YE et al., 1984).

b) **Otimização Não Linear**, empregada quando há componentes não lineares na função objetivo ou nas restrições (LUENBERGER; YE et al., 1984).

c) **Otimização Combinatória**, em que se busca identificar a melhor solução em um conjunto finito ou discretizado de objetos (SCHRIJVER et al., 2003).

2.2 POSICIONAMENTO DE TORRES DE SINAL

O estudo de (FERREIRA et al., 2024) aborda o problema de localização de instalações, amplamente aplicado em diferentes contextos, como o posicionamento estratégico de ambulâncias para maximizar o atendimento de pacientes em situações de emergência (ADENSO-DIAZ; RODRIGUEZ, 1997), a localização de escolas que atendam ao maior número possível de residências (PIZZOLATO; BARCELOS; LORENA, 2004) e, no caso deste trabalho, a instalação de antenas de telecomunicação (torres de sinal) que otimizem o acesso à Internet para o maior número de usuários (LORENA; PEREIRA, 2002). Em todos esses cenários, o objetivo central é posicionar as instalações de forma eficiente, garantindo que os usuários estejam dentro do alcance das mesmas, enquanto se busca minimizar a sobreposição, ou seja, evitar que um mesmo usuário seja atendido por mais de uma instalação (FERREIRA et al., 2024).

Originalmente, esse problema é formulado como um problema de otimização combinatória (FERREIRA et al., 2024), no qual se deve selecionar um subconjunto de instalações em uma malha bidimensional, a fim de maximizar o número de pessoas (ou células da malha) que estejam dentro do alcance delas. Caso a malha seja discretizada — ou seja, as torres só possam ser posicionadas em coordenadas inteiras —, por exemplo, em uma grade $n \times n$ com $n = 5$, então só poderiam ser usadas as seguintes coordenadas:

x \y	**0**	**1**	**2**	**3**	**4**
0	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
1	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
2	(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
3	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
4	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)

Quadro 1 – Possíveis posições de uma torre em uma malha 5×5

Temos, portanto, 25 possíveis posições para acomodar *uma* torre na malha de 5×5.

Contudo, as meta-heurísticas e o algoritmo proposto neste trabalho geram vetores de solução em um espaço contínuo $\mathbb{R}^d$, onde $d \in \mathbb{Z}$ e $d = 2T$ (com T representando o número de torres, sendo que cada torre possui uma coordenada x e uma coordenada y). Assim, o problema foi adaptado para permitir que as torres sejam posicionadas em qualquer ponto dentro da malha, desde que respeitem as restrições de tamanho dessa malha. A formulação matemática completa do problema é apresentada no Capítulo 5.

2.3 META-HEURÍSTICAS

Uma meta-heurística é uma técnica de alto nível projetada para identificar ou ajustar uma heurística (também conhecida como algoritmo de busca parcial) capaz de encontrar uma solução razoavelmente boa para um problema de otimização (BIANCHI et al., 2009).

As meta-heurísticas são suficientemente flexíveis para serem utilizadas em questões onde não dispomos de informações completas sobre o problema ou quando o problema é de tamanha complexidade que não dispomos de recursos computacionais suficientes (BIANCHI et al., 2009).

Ao contrário de algoritmos de otimização tradicionais ou métodos iterativos, essas técnicas não garantem que alcançaremos a solução ótima para o problema em questão. Grande parte das meta-heurísticas possui componentes aleatórios em seus algoritmos, tornando-os métodos altamente estocásticos (BLUM; ROLI, 2003).

Em problemas de otimização combinatória, muitos dos quais pertencem à classe NP-completa, torna-se inviável resolver exatamente em tempo viável a partir de níveis moderados de complexidade (PAPADIMITRIOU; STEIGLITZ, 1998). Por exemplo, como ilustrado no exemplo da malha 5×5 na seção anterior, o que aconteceria se aumentássemos um pouco o número de torres, considerando-as **indistinguíveis** e que pudessem ser posicionadas no mesmo local? Teríamos então uma combinação com repetição (GUICHARD, 2017), que pode ser definida como:

$$C(n+k-1,k) = \binom{n+k-1}{k}$$

onde $C(n+k-1,k)$ representa o número de combinações com repetição possíveis para posicionar k torres em n locais distintos.

O termo conhecido como explosão combinatória (EDWARDS; GLASS, 2000) se aplica a esses tipos de problemas, quando uma pequena perturbação em uma variável aumenta exponencialmente a quantidade de possibilidades (ou possíveis passos) que seriam necessários para explorarmos o problema completamente.

Número de torres	**Possibilidades**
$k = 1$	25
$k = 2$	325
$k = 5$	118,755
$k = 10$	131,128,020

Quadro 2 – Exemplo da explosão combinatória ao posicionar k torres em uma malha 5×5.

Nesse cenário, as meta-heurísticas frequentemente oferecem soluções de boa qualidade com menos esforço computacional do que métodos iterativos, heurísticas simples ou aproximações tradicionais (SÖRENSEN; GLOVER, 2013). Essa eficiência também se aplica a

problemas de otimização contínua ou com números inteiros mistos (GENDREAU; POTVIN et al., 2010).

Perto do fim do século XX, tornou-se comum utilizar heurísticas inspiradas nos mecanismos de adaptação dos seres vivos para resolver problemas de alta complexidade. Esses mecanismos, observados na natureza, baseiam-se na interação e coordenação de agentes simples que, em conjunto, solucionam problemas de adaptação ao ambiente. Exemplos naturais incluem colônias de formigas, bandos de pássaros e respostas imunológicas de mamíferos, que lidam com problemas de busca e adaptação contínua em ambientes dinâmicos (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012).

2.3.1 Algoritmo Genético

Algoritmos genéticos são meta-heurísticas inspiradas nos mecanismos de evolução de populações de seres vivos. Desenvolvidos originalmente por (HOLLAND, 1992) e posteriormente difundidos por (GOLDBERG, 1989), esses tipos de algoritmos são guiados pela seleção natural de Darwin, onde os indivíduos mais *aptos* possuem a maior chance de sobreviver e gerar descendentes (LACERDA; CARVALHO, 1999).

O algoritmo começa gerando uma população inicial aleatória de possíveis soluções para o problema a ser resolvido (LACERDA; CARVALHO, 1999). Durante o processo de evolução, toda a população é avaliada com base na função objetivo, e cada indivíduo passa a possuir uma aptidão que indica a qualidade da solução em relação ao problema (para minimização, a melhor aptidão será a mais baixa, enquanto para maximização ocorre o contrário) (GOLDBERG, 1989).

Seguindo o processo de seleção natural, no próximo passo (**seleção**), os indivíduos mais aptos são escolhidos para gerar descendentes. Essa escolha pode ser feita por comparação, onde pares de indivíduos são comparados e o com melhor aptidão é selecionado (Torneio), ou de forma aleatória ponderada com pesos baseados na aptidão, onde indivíduos com maior aptidão têm maior chance de serem selecionados (Roleta) (LACERDA; CARVALHO, 1999). A quantidade de indivíduos selecionados para recombinação é controlada por um hiperparâmetro denominado taxa de recombinação (T_r). Esse hiperparâmetro assume valores reais no intervalo [0, 1], onde valores próximos a 1 indicam que quase toda a população será submetida à recombinação, enquanto valores mais baixos sugerem uma recombinação mínima ou inexistente. Estudos na literatura recomendam configurar esse parâmetro em valores entre 0.6 e 0.8 (LACERDA; CARVALHO, 1999).

Os indivíduos selecionados são então recombinados (**cruzamento**) e geram filhos que possuem as características dos pais (LACERDA; CARVALHO, 1999). Um exemplo deste operador com dois indivíduos que representam soluções para um problema com 4 dimensões:

$$I_1 = [0,\ 2,\ 4,\ 3] \tag{2.4}$$

$$I_2 = [1,\ 3,\ 4,\ 2] \tag{2.5}$$

O indivíduo filho pode ser gerado a partir de diferentes métodos. Um deles, que foi utilizado como inspiração para o GA-BBPSO mais à frente, é realizar a média dos pais e, com isso, obter o filho:

$$I_{1,2} = [0.5,\ 2.5,\ 4,\ 2.5] \tag{2.6}$$

Após realizar o cruzamento, a população inteira (pais e proles) pode ou não sofrer um processo de **mutação**. Esse processo é necessário caso as soluções tenham estagnado em um mínimo local, diversificando e aumentando a exploração da população para que ela saia desse ponto. Similar ao cruzamento, existem diversas formas de realizar uma mutação. Um exemplo simples é a multiplicação de alguns genes do indivíduo por um fator aleatório de mutação. Por exemplo:

$$I_{1,2} = [0.5,\ 2.5,\ 4,\ 2.5] \quad \text{(indivíduo inicial)}$$

Aplicando o fator de mutação ($M = 0.1$) em um gene selecionado aleatoriamente:

$$I'_{1,2} = [0.5 \cdot M,\ 2.5,\ 4,\ 2.5] = [0.05,\ 2.5,\ 4,\ 2.5]$$

Nesse caso, apenas o primeiro gene foi alterado, exemplificando como a mutação promove variação na população.

A quantidade de indivíduos selecionados para mutação é determinada por outro hiperparâmetro, denominado taxa de mutação (T_m), que também varia no intervalo [0, 1]. A literatura recomenda valores baixos para esse hiperparâmetro, visando refletir a realidade biológica, em que mutações são eventos raros. Além disso, uma taxa de mutação elevada, onde quase toda a população fosse alterada, transformaria a meta-heurística em uma busca aleatória pelo mínimo global, prejudicando sua eficiência (GOLDBERG, 1989).

O algoritmo inteiro é então executado na nova população novamente, até que um critério de parada seja satisfeito. A evolução de uma população em uma função objetivo relativamente simples pode ser observada na Figura 1 e a rotina completa de um GA genérico é representada no Algoritmo 1.

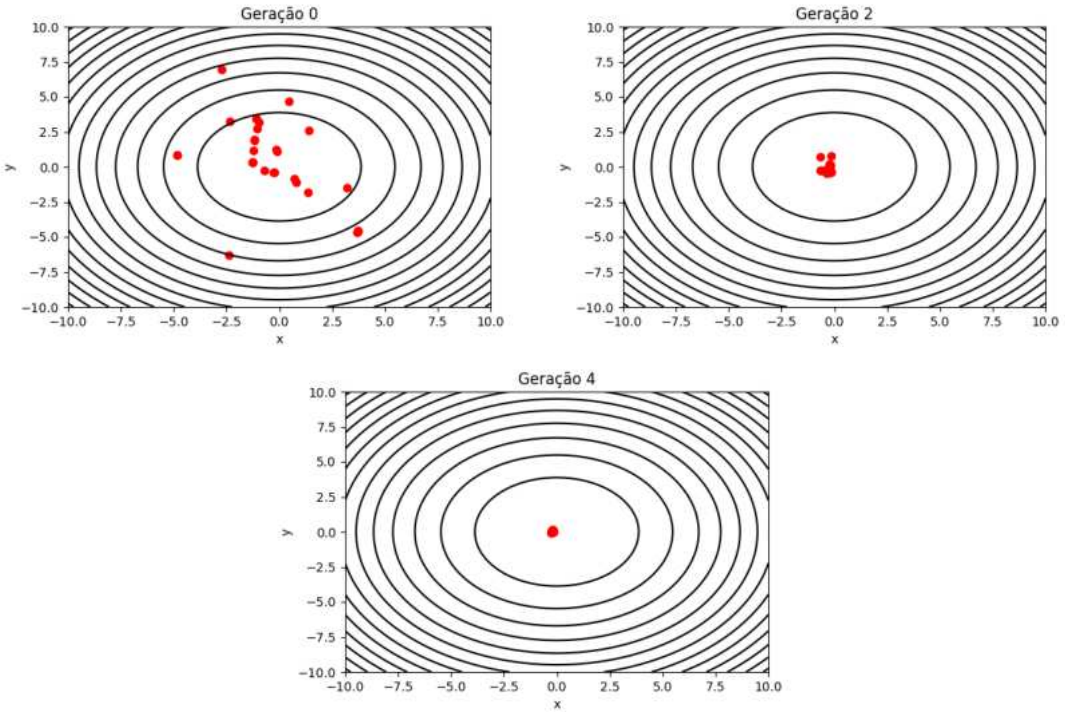


Figura 1 – Evolução do Algoritmo Genético com 30 indivíduos convergindo para o mínimo da função da Esfera ($f(x) = x^2 + y^2$).

Algoritmo 1: Algoritmo Genético Genérico

Input: N: Tamanho da População, T_r: Taxa de Recombinação, T_m: Taxa de Mutação
Output: A melhor solução encontrada

```
Inicializar a população P com N soluções aleatórias;
while critério de parada não atingido do
    Avaliar a aptidão de cada indivíduo em P;
    Selecionar indivíduos para reprodução com base em T_r;
    Realizar cruzamento para gerar nova geração;
    Aplicar mutação aos indivíduos com probabilidade T_m;
    Atualizar a população P;
end
return a melhor solução encontrada;
```

2.3.2 Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas (PSO) foi proposta por (KENNEDY; EBERHART, 1995) e é fortemente inspirada no comportamento coletivo de animais, como bandos de pássaros ou cardumes de peixes. A ideia central desse algoritmo é gerar uma população de soluções que percorrem o espaço de busca, com seus trajetos influenciados tanto por suas próprias experiências quanto pelo comportamento do enxame como um todo. Assim, estabelece-se uma metáfora em que todas as soluções podem explorar seus arredores, relembrar soluções passadas, observar a posição do líder do grupo (a melhor solução até o

momento) e serem influenciadas a explorar caminhos próximos a ele.

Cada indivíduo, aqui denominado partícula, representa uma possível solução, cuja posição atual é descrita por um vetor d-dimensional. A posição da partícula é atualizada iterativamente com base em uma fórmula de velocidade (v_i), composta por três componentes principais:

- **Inércia:** Representa a tendência de a partícula manter sua direção e velocidade anteriores, permitindo continuidade no movimento. Esse componente é modulado pelo hiperparâmetro w, geralmente selecionado no intervalo $[0.5, 1.0]$ (URLI, 2014).

$$v_i^{\text{inércia}} = w \cdot v_i$$

- **Fator Cognitivo:** Representa a influência da experiência individual da partícula, calculada pela diferença entre a melhor posição já encontrada por ela (p^i_{best}) e sua posição atual (x_i). Esse componente incentiva a partícula a se mover em direção à sua melhor posição conhecida e é controlado pelo hiperparâmetro c_1, comumente definido como $0.5 + \ln 2$ (ZAMBRANO-BIGIARINI; CLERC; ROJAS, 2013).

$$v_i^{\text{cognitivo}} = c_1 \cdot r_1 \cdot (p^i_{\text{best}} - x_i)$$

- **Fator Social:** Representa a influência do enxame como um todo, calculada pela diferença entre a melhor solução global encontrada pelo grupo (g_{best}) e a posição atual da partícula (x_i). Esse componente incentiva a partícula a se mover em direção à melhor posição global conhecida e é controlado pelo hiperparâmetro c_2, também definido como $0.5 + \ln 2$ (ZAMBRANO-BIGIARINI; CLERC; ROJAS, 2013).

$$v_i^{\text{social}} = c_2 \cdot r_2 \cdot (g_{\text{best}} - x_i)$$

Os fatores cognitivo e social também incorporam os valores aleatórios r_1 e r_2, que são números extraídos de uma distribuição uniforme no intervalo $[0, 1]$ (KENNEDY; EBERHART, 1995). Esses valores são atualizados a cada iteração.

A velocidade final da partícula é obtida pela soma dessas três componentes:

$$v_i = v_i^{\text{inércia}} + v_i^{\text{cognitivo}} + v_i^{\text{social}}$$

Subsequentemente, a posição da partícula é atualizada com base na nova velocidade, conforme a equação:

$$x_i = x_i + v_i$$

A cada iteração do algoritmo, a posição atual de cada partícula é avaliada e comparada com os valores de p^i_{best} e g_{best}. Caso o valor da partícula seja superior a qualquer um desses dois, eles são atualizados para refletir a nova melhor solução encontrada.

A evolução de uma população de partículas, com o vetor de velocidade representado como a seta azul, na função da Esfera se encontra na Figura 2 e a rotina completa do PSO é demonstrada no Algoritmo 2.

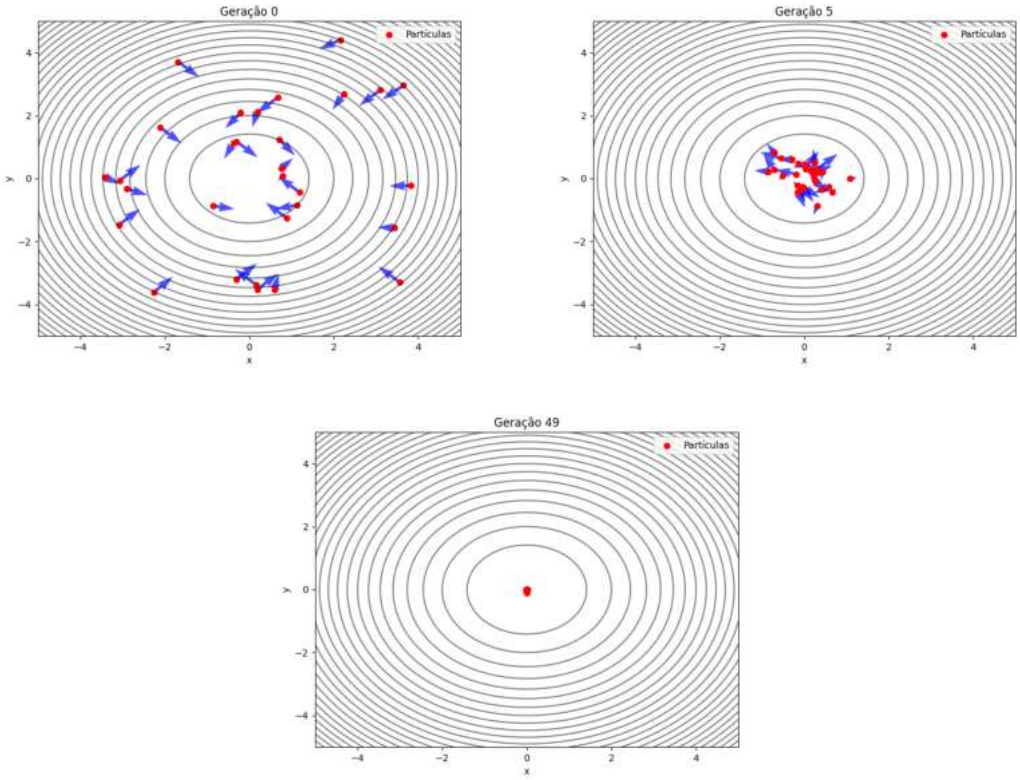


Figura 2 – Evolução do PSO com 30 indivíduos convergindo para o mínimo da função da Esfera.

Algoritmo 2: Otimização por Enxame de Partículas (PSO)

Input: N: Tamanho da População,
w: Fator de Inércia,
c_1: Fator Cognitivo,
c_2: Fator Social

Output: A melhor solução encontrada g_{best}

Inicializar uma população de partículas de tamanho N com posições x_i e velocidades v_i aleatórias;
Avaliar a função de aptidão para cada partícula;
Para cada partícula i, definir $p^i_{\text{best}} = x_i$;
Definir g_{best} como o melhor p^i_{best};
while *critério de parada não atingido* **do**
  for *cada partícula* $i \in \{1, \ldots, N\}$ **do**
    Gerar dois números aleatórios $r_1, r_2 \sim U(0, 1)$;
    Atualizar a velocidade:

$$v_i \leftarrow w \cdot v_i + c_1 \cdot r_1 \cdot (p^i_{\text{best}} - x_i) + c_2 \cdot r_2 \cdot (g_{\text{best}} - x_i)$$

    Atualizar a posição:

$$x_i \leftarrow x_i + v_i$$

    Avaliar a função de aptidão em x_i;
    if $f(x_i) < f(p^i_{best})$ **then**
      $p^i_{\text{best}} \leftarrow x_i$;
    end
    if $f(p^i_{best}) < f(g_{best})$ **then**
      $g_{best} \leftarrow p^i_{\text{best}}$;
    end
  end
end
return g_{best};

2.3.3 Otimização por Enxame de Partículas Simplificado

Kennedy identificou algumas limitações no Particle Swarm Optimization (PSO) original. O PSO é extremamente sensível aos parâmetros, como w, c_1 e c_2, o que pode influenciar significativamente seu desempenho (JORDEHI; JASNI, 2013). Além disso, há o risco de convergência prematura, onde o algoritmo pode ficar preso em mínimos locais, especialmente em funções multimodais (KENNEDY; EBERHART, 1995).

Para abordar essas questões, o algoritmo BBPSO foi introduzido por (KENNEDY, 2003). O BBPSO modifica o PSO tradicional ao eliminar a fórmula de atualização de

velocidade. Em vez disso, no BBPSO, a nova posição das partículas é determinada com base em uma distribuição normal centrada entre a melhor posição individual (p^i_{best}) e a melhor posição global (g_{best}):

$$x_i \sim \mathcal{N}\left(\frac{p^i_{best} + g_{best}}{2}, |p^i_{best} - g_{best}|\right). \tag{2.7}$$

As modificações implementadas no BBPSO concentram-se na simplificação e redução do número de parâmetros. Diferentemente do PSO tradicional, que requer ajustes finos de ω, c_1 e c_2, o novo método utiliza diretamente a média e o desvio padrão para orientar a explotação (intensificação da busca local) e a exploração do espaço de busca. A proposta do PSO simplificado, conforme sugerido pelo nome, visa reduzir a complexidade inerente ao PSO tradicional e, ao mesmo tempo, melhorar a diversidade das partículas para lidar com funções multimodais. Os resultados apresentados por (KENNEDY, 2003) demonstraram que o BBPSO é competitivo em relação ao PSO original. O Algoritmo 3 ilustra o novo método.

Algoritmo 3: Otimização por Enxame de Partículas Simplificado (BBPSO)

Input: N: Tamanho da População
Output: A melhor solução encontrada g_{best}

Inicializar uma população de partículas de tamanho N com posições x_i;
Avaliar a função de aptidão para cada partícula;
Para cada partícula i, definir $p^i_{\text{best}} = x_i$;
Definir g_{best} como o melhor p^i_{best};
while *critério de parada não atingido* **do**
    for *cada partícula* $i \in \{1, \ldots, N\}$ **do**
        Atualizar a posição utilizando uma distribuição normal:

$$x_i \sim \mathcal{N}\left(\frac{p^i_{\text{best}} + g_{best}}{2}, |p^i_{\text{best}} - g_{best}|\right)$$

        Avaliar a função de aptidão em x_i;
        if $f(x_i) < f(p^i_{best})$ **then**
            $p^i_{\text{best}} \leftarrow x_i$;
        end
        if $f(p^i_{best}) < f(g_{best})$ **then**
            $g_{best} \leftarrow p^i_{\text{best}}$;
        end
    end
end
return g_{best};

Apesar da simplicidade e eficiência do BBPSO, o algoritmo ainda enfrenta problemas de convergência prematura. Isso ocorre quando o valor da função objetivo p_{best}^{i} de uma partícula se iguala ao g_{best}, fazendo com que o desvio padrão seja zero e impedindo a atualização dessa partícula. Uma solução possível é atribuir um valor fixo pequeno ao desvio padrão, porém isso limita a capacidade de exploração de forma constante (ZHANG et al., 2011).

3 TRABALHOS RELACIONADOS

Em (KENNEDY, 2003), o próprio criador do método observa que, como p^i_{best} representa a melhor posição já encontrada para x_i e provavelmente não é uma posição ruim, reter alguns elementos do vetor associado ao melhor desempenho anterior pode acelerar a convergência.

Com base nessa ideia, surge a nova versão do método, denominada Exploiting Barebones PSO (BBePSO). Nesse modelo, a influência de p^i_{best} é incorporada de forma seletiva, sendo aceita apenas se passar por um teste probabilístico. A atualização da posição de uma partícula segue a equação:

$$x_i = \begin{cases} \mathcal{N}\left(\frac{p^i_{\text{best}}+g_{best}}{2}, |p^i_{\text{best}} - g_{best}|\right), & \text{se } U(0,1) < 0.5, \\ p^i_{best}, & \text{caso contrário.} \end{cases}$$

Com uma probabilidade de 0.5, a posição de uma partícula pode ser ajustada para seu melhor histórico pessoal, aumentando, assim, a exploração direcionada (explotação) e mostrou ser, de acordo com (KENNEDY, 2003), uma das versões mais confiáveis do algoritmo, com os melhores resultados nos testes realizados.

Com base no método anterior, (DURÁN-ROSAL et al., 2019) propôs o algoritmo Dynamic BBePSO (DBBePSO) que ajusta dinamicamente a importância das componentes cognitivas e sociais ao longo das gerações. A atualização das posições segue:

$$x_i = \begin{cases} N\left(\frac{p^i_{\text{best}}+g_{best}}{2}, \lambda|p^i_{\text{best}} - g_{best}|\right), & \text{se } U(0,1) < 0.5, \\ p^i_{best}, & \text{caso contrário.} \end{cases}$$

A principal novidade é o parâmetro λ, que regula o desvio padrão da distribuição gaussiana e decresce ao longo das gerações de acordo com:

$$\lambda = 0.9\frac{(L-l)}{L} + 0.1,$$

onde L é o número máximo de avaliações de função (métrica de tempo em uma metaheurística) e l o número atual de avaliações. Assim, λ começa em 1.0 e decresce para 0.1.

Em (WANG et al., 2013), é apresentado o Gaussian Bare-Bones Differential Evolution (GBDE), uma variante simplificada da Evolução Diferencial (DE) (STORN; PRICE, 1997), que elimina a necessidade de ajuste do fator de escala F, substituindo a mutação tradicional por uma abordagem baseada em uma distribuição gaussiana. No GBDE, os novos vetores de mutação são gerados a partir da fórmula:

$$V_{i,G} = \mathcal{N}(\mu, \sigma),$$

onde a média $\mu = \frac{g_{best}+x_i}{2}$ representa o ponto médio entre a melhor solução encontrada e a posição atual, enquanto o desvio padrão $\sigma = |g_{best} - x_i|$ expressa a distância entre esses dois pontos. O GBDE também preserva as etapas de cruzamento e seleção da DE clássica, gerando um vetor de teste $U_{i,G}$ por meio da combinação entre $V_{i,G}$ e x_i. Entretanto, esse método introduz o hiperparâmetro CR (Taxa de Recombinação) da DE, que, neste caso específico, é ajustado de forma adaptativa com base na comparação entre a aptidão do vetor de teste e a posição atual. Essa estratégia de adaptação, embora eficiente, exige uma avaliação adicional da função objetivo como contrapartida.

Por fim, (GUO et al., 2023) investigaram o impacto da seleção de descendentes em algoritmos evolutivos, propondo o BBPSO com memória cruzada (BPSO-CM), que introduz dois novos mecanismos: o mecanismo de armazenamento de múltiplas memórias (MSM) e a estratégia de seleção de descendentes de elite (EOSS). O MSM amplia a capacidade de memória das partículas, permitindo que cada partícula armazene não apenas a melhor posição histórica, mas também outras melhores posições, aumentando as chances de escapar de ótimos locais. Já o EOSS é responsável por selecionar as melhores posições candidatas para serem armazenadas, garantindo diversidade sem sacrificar a precisão.

Embora a eficácia do BPSO-CM seja competitiva com outros métodos, o algoritmo adiciona complexidade ao incluir os novos mecanismos, pois aumenta a carga computacional ao exigir que cada partícula mantenha e atualize múltiplas memórias, além de demandar avaliações adicionais para selecionar os melhores candidatos, especialmente em configurações com muitos indivíduos e altas dimensões. Isso faz com que o custo computacional do algoritmo cresça proporcionalmente ao número de partículas, à profundidade da memória e à dimensão do problema. Além disso, o método requer o ajuste adequado de novos parâmetros, como a profundidade da memória das partículas (PD) e a profundidade da memória global (GD), que desempenham um papel crítico no equilíbrio entre exploração e explotação.

Enquanto as abordagens de (KENNEDY, 2003) e (DURÁN-ROSAL et al., 2019) focam em mitigar o problema da coincidência entre p_{best} e g_{best} (ZHANG et al., 2011) com baixo custo computacional – enfatizando a exploração pessoal –, outras propostas, como as de (WANG et al., 2013) e (GUO et al., 2023), implementam técnicas mais sofisticadas que priorizam a exploração global e promovem maior diversidade na população. No entanto, tais estratégias demandam avaliações de função adicionais e aumentam o uso de memória do algoritmo. Essa constatação abre a possibilidade de combinar o melhor desses dois mundos, buscando desenvolver uma nova heurística que amplie a diversidade da população sem comprometer a eficiência computacional.

4 ALGORITMO PROPOSTO

Levando em consideração as vantagens dos estudos apresentados, este trabalho propõe uma variação híbrida do BBPSO original, combinada com operadores inspirados nos mecanismos de cruzamento e mutação do GA. A ideia por trás dessa hibridização é intensificar a exploração, evitando a falta de atualização das partículas e prevenindo a convergência prematura, ao mesmo tempo em que se preserva o baixo custo computacional, com menos avaliações de função. Além disso, dois novos parâmetros (inspirados no GA) foram introduzidos. Mesmo sem ajustes finos, esses parâmetros mostraram resultados competitivos em comparação com os métodos originais.

O algoritmo segue, em essência, a mesma configuração do BBPSO original, com modificações na média (μ) e no desvio padrão (σ), utilizados para gerar novas partículas a partir da distribuição gaussiana.

No operador de cruzamento, com uma taxa T_r predefinida, a média do BBPSO passa a ser calculada com base na posição atual da partícula e na posição de outra partícula aleatória do enxame (x_{idx}). O valor de T_r foi fixado em 0.8, conforme a literatura (LACERDA; CARVALHO, 1999). Assim, em 80% dos casos, a partícula segue o novo operador, enquanto nos outros 20%, utiliza o método anterior. O objetivo desse operador é evitar que a partícula se guie exclusivamente pelo ótimo global e o melhor pessoal, que podem, em certos casos, estarem presos em um mínimo local. A fórmula desse operador é dada por:

$$\mu = \begin{cases} \frac{x_i + x_{idx}}{2}, & \text{se } U(0,1) < T_r, \\ \frac{p^i_{best} + g_{best}}{2}, & \text{caso contrário.} \end{cases}$$

O operador de mutação é guiado pelo hiperparâmetro T_m, cujo valor foi fixado em 0.1 com base na literatura (GOLDBERG, 1989). Com uma chance de 10%, o desvio padrão será calculado a partir da diferença entre a partícula atual (x_i) e o melhor global (g_{best}), com o vetor g_{best} tendo seus genes alterados por um valor extraído de uma distribuição uniforme entre 0 e 1. Esse mecanismo evita que o desvio padrão seja zerado caso p^i_{best} e g_{best} sejam iguais, ou quando x_i é igual ou próximo de g_{best}. Além disso, promove maior exploração ao ignorar, em alguns casos, o melhor já encontrado. A fórmula é expressa como:

$$\sigma = \begin{cases} |x_i - g_{best} \cdot U(0,1)|, & \text{se } U(0,1) < T_m, \\ |p^i_{best} - g_{best}|, & \text{caso contrário.} \end{cases}$$

A rotina do algoritmo é apresentada no Algoritmo 4.

Algoritmo 4: Algoritmo Genético com Otimização por Enxame de Partículas Simplificado (GA-BBPSO)

Input: N: Tamanho da População, T_r: Taxa de Recombinação, T_m: Taxa de Mutação

Output: A melhor solução encontrada g_{best}

Inicializar uma população de partículas de tamanho N com posições x_i;
Avaliar a função de aptidão para cada partícula;
Para cada partícula i, definir $p^i_{\text{best}} = x_i$;
Definir g_{best} como o melhor p^i_{best};
while *critério de parada não atingido* **do**
    for *cada partícula* $i \in \{1, \ldots, N\}$ **do**
        Calcular μ e σ com base nas equações apresentadas;
        Atualizar a posição da partícula usando uma distribuição normal;
        Avaliar a nova posição;
        Atualizar p^i_{best} e g_{best}, se necessário;
    end
end
return g_{best};

5 EXPERIMENTOS

Conforme citado anteriormente, não existe uma meta-heurística perfeita capaz de resolver eficientemente todos os tipos de problemas (WOLPERT; MACREADY, 2005). No entanto, antes de aplicar o algoritmo proposto a problemas reais, utilizamos funções de benchmark que simulam problemas complexos do mundo real, permitindo comparar o desempenho dos métodos em diferentes cenários (PLEVRIS; SOLORZANO, 2022). O objetivo é verificar se, em um dado conjunto de funções, o algoritmo proposto apresenta melhorias significativas em relação ao BBPSO original. Em caso positivo, aplicaremos e avaliaremos seu desempenho no problema das torres de sinal.

5.1 FUNÇÕES DE BENCHMARK

Funções de benchmark, ou funções de teste, são funções matemáticas amplamente utilizadas para avaliar o desempenho, a taxa de convergência, a precisão e a robustez de algoritmos de otimização (PLEVRIS; SOLORZANO, 2022). Essas funções são projetadas para se assemelhar ou imitar problemas reais complexos, permitindo analisar em quais contextos cada algoritmo apresenta melhor desempenho. Neste estudo, foram selecionadas 21 funções multidimensionais bem estabelecidas como testes, retiradas de (GAVANA, 2024), (PLEVRIS; SOLORZANO, 2022) e (SURJANOVIC; BINGHAM, 2024), representadas no Quadro 3.

5.2 METODOLOGIA

Devido à natureza estocástica dos algoritmos, uma única execução não é suficiente para avaliar seu desempenho. Um algoritmo pode ter resultados excepcionais por sorte ou ficar preso em mínimos locais, sem que mutações ou cruzamentos corrijam essa limitação. Por isso, é necessário realizar várias execuções e aplicar inferência estatística aos resultados.

Neste trabalho, cada algoritmo foi executado 30 vezes para instâncias de 30 e 50 dimensões, conforme documentado em trabalhos anteriores (MARCELINO, 2017). Com os resultados, calculamos a média e o desvio padrão, avaliando tanto a qualidade quanto a consistência das soluções. Esse procedimento, fundamentado nas Leis dos Grandes Números (MICHEL et al., 2005), assegura que, com mais execuções, a média dos resultados se aproxima do valor esperado, permitindo uma análise mais confiável. O desvio padrão mede a variabilidade, indicando a estabilidade do algoritmo em diferentes cenários.

Além disso, como será evidenciado nas tabelas a seguir, em alguns casos os algoritmos alcançaram resultados muito próximos. Para esses casos, utilizamos o teste **t de Student** (STUDENT, 1908) para comparar as médias, e verificar, com significância estatística,

Quadro 3 – Funções de Benchmark

Nome	Definição
Sphere	$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} \left[(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right]$
Rotated Hyper-Ellipsoid	$f_3(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
Bent Cigar	$f_4(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$
Discus	$f_5(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$
Bohachevsky	$f_6(\mathbf{x}) = \sum_{i=1}^{n-1} \left(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7\right)$
ModRidge	$f_7(\mathbf{x}) = \lvert x_1\rvert + 2\left(\sum_{i=2}^{n} x_i^2\right)^{0.1}$
Alpine	$f_8(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i\rvert$
Rastrigin	$f_9(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$
Griewank	$f_{10}(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$
Ackley	$f_{11}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
Schwefel	$f_{12}(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$
Levy	$f_{13}(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1}\left[(w_i - 1)^2\left(1 + 10\sin^2(\pi w_i + 1)\right)\right] + (w_n - 1)^2\left(1 + \sin^2(2\pi w_n)\right)$
Dixon-Price	$f_{14}(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n}\left[i\left(2x_i^2 - x_{i-1}\right)^2\right]$
Schaffer F7	$f_{15}(\mathbf{x}) = \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\left[\sqrt{x_i^2 + x_{i+1}^2} + \sqrt{x_i^2 + x_{i+1}^2}\sin^2\left(50(\sqrt{x_i^2 + x_{i+1}^2})^{0.1}\right)\right]\right)^2$
Drop-Wave	$f_{16}(\mathbf{x}) = 1 - \frac{1+\cos\left(12\sqrt{\sum_{i=1}^{n} x_i^2}\right)}{0.5\sum_{i=1}^{n} x_i^2 + 2}$
Salomon	$f_{17}(\mathbf{x}) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$
Happy Cat	$f_{18}(\mathbf{x}) = \left\lvert\left(\sum_{i=1}^{n} x_i^2\right) - n\right\rvert^{\alpha} + \frac{0.5\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i}{n} + 0.5,\ \alpha = \frac{1}{4}$
HG Bat	$f_{19}(\mathbf{x}) = \left\lvert\left(\frac{\sum_{i=1}^{n} x_i^2}{n}\right)^2 - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2\right\rvert^{0.5} + \frac{0.5\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i}{n} + 0.5$
Schwefel 2.21	$f_{20}(\mathbf{x}) = \max_{1 \le i \le n} \lvert x_i\rvert$
Schwefel 2.22	$f_{21}(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$

se um método é realmente mais eficiente em relação ao outro, naquele caso específico. Também avaliamos as curvas de convergência média, que mostram o quão rápido cada algoritmo conseguiu chegar ao mínimo em uma determinada função.

5.3 IMPLEMENTAÇÃO

Para facilitar os testes e experimentos realizados, foi desenvolvida uma estrutura de projeto modular e extensível. Nesse sentido, o projeto conta com um único ponto de entrada, o arquivo `main.py`, que orquestra a execução do sistema por meio do consumo de arquivos JSON contendo as configurações específicas de cada método. Assim, a parametrização flexível permite a experimentação com diferentes algoritmos e configurações sem a necessidade de alterações profundas no código.

Adicionalmente, todas as funções de benchmark foram concebidas de forma escalável, possibilitando sua adaptação para o número desejado de dimensões. Além disso, foi criada uma única classe capaz de englobar e generalizar todos os métodos avaliados. Essa abordagem simplifica tanto a integração quanto a substituição dos métodos, facilitando a comparação de desempenho e a seleção dos algoritmos que compõem este estudo.

Para realizar a manipulação dos dados obtidos durante os experimentos, o projeto dispõe de classes dedicadas ao salvamento e à análise dos resultados. Para esse fim, a biblioteca `pandas` (MCKINNEY, 2010) é utilizada no armazenamento e processamento dos dados, permitindo o cálculo de métricas estatísticas, como médias e desvios padrão. Além disso, as ferramentas estatísticas do `scipy` (VIRTANEN et al., 2020) foram empregadas na realização dos testes de hipótese. Para consumir os dados salvos e gerar gráficos, outra classe foi desenvolvida, com o auxílio da biblioteca `matplotlib` (HUNTER, 2007).

É importante ressaltar que todas as operações matemáticas, funções de teste e cálculos do projeto foram implementados utilizando a biblioteca `numpy` (HARRIS et al., 2020). A adoção do `numpy` foi determinante para melhorar o desempenho e reduzir o tempo de execução dos experimentos, pois possibilita a realização de operações vetorizadas e a eliminação de loops desnecessários. Um exemplo dessa otimização é a função de cobertura de torres, na qual o uso de operações vetorizadas e a pré-computação dos deslocamentos eliminaram a necessidade de iterações aninhadas para cada torre, reduzindo drasticamente o custo computacional.

Por fim, destaca-se que o projeto é de código aberto e está disponível para acesso e colaboração na plataforma GitHub (GITHUB, 2020), podendo ser acessado em *TCC*.

5.4 AMBIENTE

O desempenho do GA-BBPSO será comparada com seu predecessor, o BBPSO, e com o método que originou ambos, o PSO. Com exceção do BBPSO tradicional, que não possui parâmetros ajustáveis, a Tabela 4 apresenta os parâmetros utilizados para o GA-BBPSO e o PSO. Os parâmetros do PSO foram empiricamente ajustados ao longo dos anos, conforme descrito em (ZAMBRANO-BIGIARINI; CLERC; ROJAS, 2013).

Quadro 4 – Parâmetros utilizados pelo GA-BBPSO e PSO em todos os experimentos realizados.

Método	**Parâmetro**	**Valor**
GA-BBPSO	T_r	0.8
GA-BBPSO	T_m	0.1
PSO	w	$0.5 \cdot \log 2$
PSO	c_1	$0.5 + \log 2$
PSO	c_2	$0.5 + \log 2$

Cada execução dos algoritmos foi realizada com critério de parada definido pelo número máximo de avaliações da função objetivo, fixado em $1 \cdot 10^5$. A população foi configurada de acordo com o número de dimensões do problema (número de indivíduos = número de dimensões) (MARCELINO et al., 2018).

Os experimentos foram realizados em um computador com sistema operacional Ubuntu 22.04, processador Intel i5-1035G1 (8) @ 3.600GHz, 16 GB de RAM, utilizando Python 3.11.

5.5 RESULTADOS DOS TESTES

Tabela 1 – Desempenho de cada método em todas as funções

Função	**Dim**	**BBPSO**		**GA-BBPSO**		**PSO**	
		Média	**Desvio**	**Média**	**Desvio**	**Média**	**Desvio**
f_1	30	**2.490e-57**	**7.131e-57**	3.792e-28	6.418e-28	0.003	0.006
	50	2.621	7.864	**4.468e-10**	**3.514e-10**	0.0366	0.010
f_2	30	37.263	49.746	**23.241**	1.210	28.565	**1.159**
	50	151.923	229.702	**45.299**	**1.038**	52.898	1.761
f_3	30	4180.610	2921.827	9.157	5.094	**0.530**	**1.124**
	50	17194.872	6203.977	1425.564	209.561	**12.530**	**21.158**
f_4	30	7333.333	4422.166	**2.225e-19**	**6.070e-19**	3245.804	1043.435
	50	333341666.666	1795055245.207	**0.104**	**0.086**	42936.804	13224.878
f_5	30	13333.333	9775.252	**1.996e-25**	**3.915e-25**	8.489	4.565
	50	23666.666	18882.678	**2.395e-7**	**2.281e-7**	41.563	15.409
f_6	30	1.468	1.143	**0.688**	**0.819**	4.966	1.802
	50	25.446	88.380	**1.503**	**1.430**	18.973	3.065
f_7	30	3.4033	2.598	**0.004**	**0.0005**	1.320	0.065
	50	5.311	1.428	**0.412**	**0.033**	1.836	0.061
f_8	30	1.628	2.684	**2.507e-9**	**8.240e-9**	0.180	0.128
	50	7.867	5.877	**0.006**	**0.025**	0.741	0.412
f_9	30	103.104	32.766	38.703	11.336	**36.126**	**9.775**
	50	259.595	47.221	83.408	20.747	**67.212**	**14.116**
f_{10}	30	**0.008**	0.013	0.009	**0.010**	0.012	0.015
	50	3.009	16.172	**0.004**	**0.005**	0.016	0.011
f_{11}	30	0.321	0.690	**7.102e-14**	**4.231e-14**	2.483	0.499
	50	2.095	5.147	**8.894e-4**	**3.822e-4**	3.830	0.580
f_{12}	30	3787.828	661.101	**3736.929**	**576.028**	8301.336	1512.610
	50	6572.651	856.573	**6359.124**	**841.095**	14792.376	2917.585
f_{13}	30	6.556	4.289	**0.402**	**0.748**	1.614	1.699
	50	17.852	7.518	**0.687**	**0.955**	2.446	1.173
f_{14}	30	4453.1	23740.626	**5.709e-24**	**1.069e-23**	0.093	0.047
	50	21441.900	55591.076	**6.249e-6**	**1.237e-4**	1.365	0.442
f_{15}	30	11.0244	6.842	**0.037**	**0.066**	10.652	2.820
	50	20.981	7.524	**0.515**	**0.506**	9.642	1.705
f_{16}	30	0.421	0.116	**0.212**	**0.011**	0.316	0.088
	50	0.656	0.098	**0.389**	**0.053**	0.600	0.058
f_{17}	30	0.353	0.071	**0.209**	**0.029**	0.383	0.077
	50	0.783	0.502	**0.436**	**0.060**	0.673	0.115
f_{18}	30	0.805	1.798	0.557	0.109	**0.536**	**0.105**
	50	0.900	1.331	**0.694**	**0.103**	0.724	0.113
f_{19}	30	**0.036**	**0.011**	0.061	0.012	0.042	**0.011**
	50	0.305	1.195	0.120	0.016	**0.081**	**0.013**
f_{20}	30	1.566	1.469	**0.046**	**0.030**	0.221	0.100
	50	90.637	2.646	6.831	2.009	**2.339**	**0.654**
f_{21}	30	**346.666**	**180.246**	534.953	226.940	7162.332	33755.762
	50	1060.000	307.245	**1047.092**	**240.492**	6.617e+12	3.549e+16

No geral, considerando a média e o desvio padrão, o GA-BBPSO mostrou-se mais eficaz em 31 dos 42 casos analisados. Contudo, em algumas situações, as médias obtidas foram semelhantes. Para avaliar se essas diferenças são estatisticamente significativas, foi realizado um teste de hipótese (**t de Student**) com nível de significância $\alpha = 0.05$. Caso a hipótese nula seja rejeitada, pode-se afirmar, com significância estatística, que o GA-BBPSO é a melhor escolha naquele caso específico. Os resultados desses testes estão apresentados na Tabela 2.

No total, considerando as 42 configurações das funções benchmark com dimensões 30 e 50, o GA-BBPSO mostrou significância estatística rejeitando H_0 em 25 casos. Em

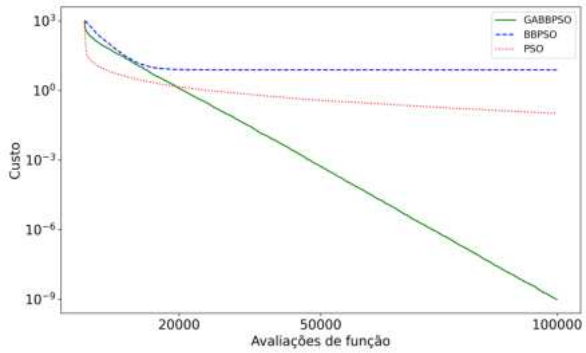
Tabela 2 – Resultados do teste de hipótese para funções onde as médias foram semelhantes.

Função	**Dimensão**	**Comparação**	**p-valor**	**Resultado**	**Resposta**
f_2	30	GA-BBPSO vs PSO	2.660e-24	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs PSO	1.228e-24	Rejeitar H_0	GA-BBPSO
f_8	50	GA-BBPSO vs PSO	1.611e-10	Rejeitar H_0	GA-BBPSO
f_9	30	GA-BBPSO vs PSO	0.357	Aceitar H_0	Ambos
	50	GA-BBPSO vs PSO	0.001	Rejeitar H_0	PSO
f_{10}	30	GA-BBPSO vs PSO	0.421	Aceitar H_0	Ambos
	50	GA-BBPSO vs PSO	2.687e-06	Rejeitar H_0	GA-BBPSO
	30	GA-BBPSO vs BBPSO	0.772	Aceitar H_0	Ambos
	50	GA-BBPSO vs BBPSO	0.325	Aceitar H_0	Ambos
f_{12}	30	GA-BBPSO vs BBPSO	0.755	Aceitar H_0	Ambos
	50	GA-BBPSO vs BBPSO	0.342	Aceitar H_0	Ambos
f_{16}	30	GA-BBPSO vs PSO	6.299e-07	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs PSO	1.179e-20	Rejeitar H_0	GA-BBPSO
	30	GA-BBPSO vs BBPSO	1.415e-10	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs BBPSO	1.093e-16	Rejeitar H_0	GA-BBPSO
f_{17}	30	GA-BBPSO vs PSO	1.669e-13	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs PSO	1.349e-12	Rejeitar H_0	GA-BBPSO
	30	GA-BBPSO vs BBPSO	3.384e-12	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs BBPSO	0.0009	Rejeitar H_0	GA-BBPSO
f_{18}	30	GA-BBPSO vs PSO	0.477	Aceitar H_0	Ambos
	50	GA-BBPSO vs PSO	0.303	Aceitar H_0	Ambos
	30	GA-BBPSO vs BBPSO	0.463	Aceitar H_0	Ambos
	50	GA-BBPSO vs BBPSO	0.412	Aceitar H_0	Ambos
f_{19}	30	GA-BBPSO vs PSO	8.734e-08	Rejeitar H_0	PSO
	50	GA-BBPSO vs PSO	4.643e-14	Rejeitar H_0	PSO
	30	GA-BBPSO vs BBPSO	7.301e-11	Rejeitar H_0	BBPSO
	50	GA-BBPSO vs BBPSO	0.411	Aceitar H_0	Ambos
f_{20}	30	GA-BBPSO vs BBPSO	5.177e-06	Rejeitar H_0	GA-BBPSO
	30	GA-BBPSO vs PSO	1.549e-10	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs PSO	2.132e-13	Rejeitar H_0	PSO
f_{21}	50	GA-BBPSO vs BBPSO	0.859	Aceitar H_0	Ambos

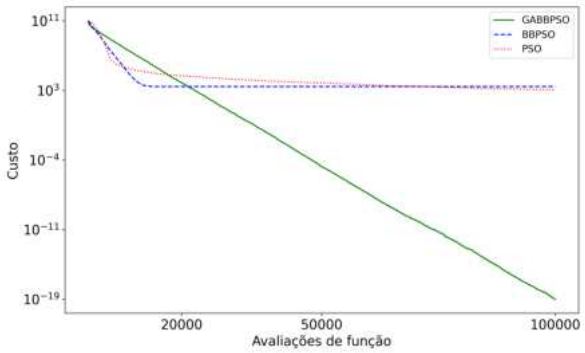
9 funções aceitou H_0, e mostrou que em 5 funções, quando comparado aos demais, a hipótese alternativa H_1 beneficiou a outra técnica. É importante observar que na função f_{19} com 30 dimensões, o GA-BBPSO mostrou dificuldade em encontrar uma solução média comparável aos demais algoritmos, salvo com 50 dimensões em relação ao BBPSO.

Também é possível observar que, em determinados casos, o GA-BBPSO apresentou resultado estatístico robusto, especialmente nas funções f_1, f_4, f_5, f_{11} e f_{14}. Esse comportamento é evidenciado pelas curvas de convergência apresentadas na Figura 3. Nessas curvas, também percebe-se que o GA-BBPSO não só superou o BBPSO, mas também foi significativamente mais robusto que o método original (PSO). Isso provavelmente se deve à diversidade amplificada pelos novos operadores adicionados. Caso o método fique preso em um mínimo local, ele dispõe de mais recursos para escapar, evitando depender exclusivamente da memória do melhor indivíduo já encontrado. Além disso, se o melhor global estiver preso, a mutação pode alterar sua posição, permitindo a superação desse obstáculo.

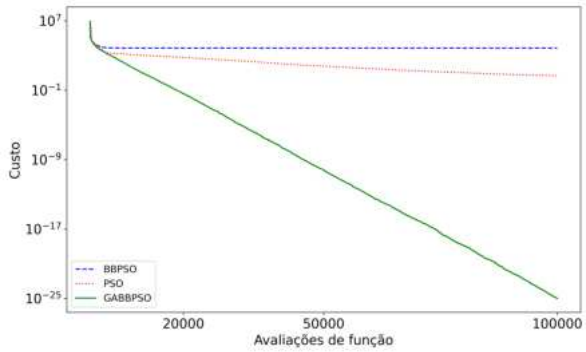
Figura 3 – Curvas de convergência para as funções f_1, f_4, f_5, f_{11} e f_{14}, onde o GA-BBPSO demonstrou desempenho estatisticamente robusto.



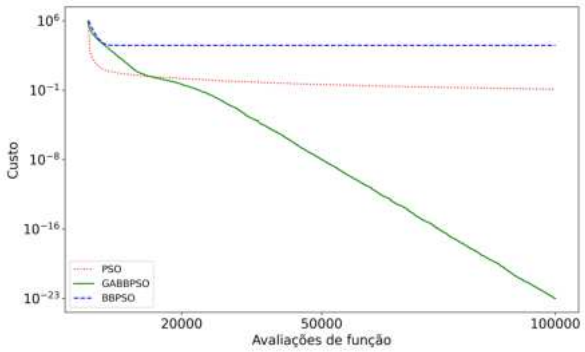
(a) Convergência em f_1



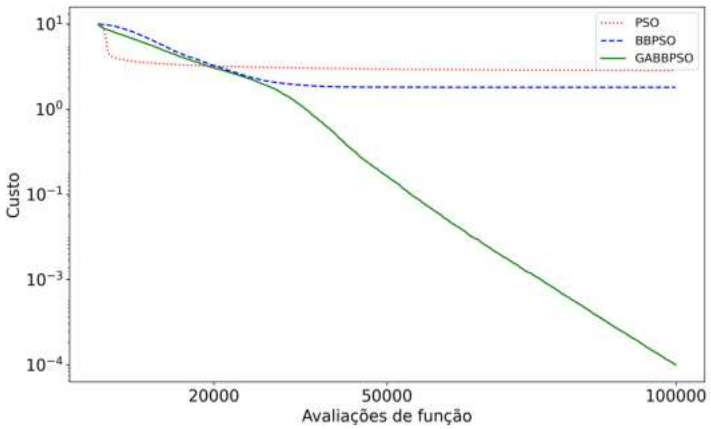
(b) Convergência em f_4



(c) Convergência em f_5



(d) Convergência em f_{11}



(e) Convergência em f_{14}

Na função f_2, todos os algoritmos seguem uma convergência gradual em direção ao mínimo. O PSO inicia com vantagem, mas logo estagna, enquanto o GA-BBPSO continua avançando em direção ao mínimo global. O BBPSO, por sua vez, estagna e não consegue sair do mínimo.

Na função f_3, todos os métodos enfrentam dificuldades. Embora o GA-BBPSO comece ligeiramente melhor, perde eficiência ao longo das gerações, enquanto o PSO continua avançando e atinge um mínimo local.

Para a função f_6, o PSO inicialmente mostra-se mais eficiente, convergindo rapida-

mente nas primeiras iterações. No entanto, ele atinge um mínimo local e permanece estagnado. O GA-BBPSO e o BBPSO enfrentam dificuldades semelhantes, atingindo o mesmo mínimo local. Contudo, devido aos novos operadores, o GA-BBPSO consegue escapar desse mínimo e continuar a convergir, consolidando-se como o método com melhor desempenho nessa função. O BBPSO, por outro lado, permanece preso no mínimo local.

Na função f_7, os métodos começam com alguma dificuldade, mas logo após as primeiras iterações, o GA-BBPSO consegue convergir de forma contínua em direção ao mínimo global. O BBPSO estagna e não consegue convergir, enquanto o PSO tenta, mas acaba obtendo um valor inferior.

Para a função f_8, cada método apresenta um comportamento distinto. Nas primeiras iterações, o GA-BBPSO parece ser o pior, com convergência lenta e estagnação. Contudo, à medida que as avaliações de função aumentam, ele retoma a convergência, demonstrando eficiência ao superar mínimos locais graças aos novos operadores. O PSO começa a convergir rapidamente, mas sua melhoria por iteração desacelera o processo. O BBPSO apresenta uma convergência mais lenta.

Na função f_9, o GA-BBPSO inicialmente é o método mais lento, possivelmente devido à grande quantidade de mínimos locais presentes. No entanto, ele rapidamente ultrapassa o BBPSO e, em certo momento, até o PSO. Eventualmente, o PSO mostra-se mais eficiente até o final, seguido do GA-BBPSO.

Na função f_{10}, o GA-BBPSO inicialmente se destaca, mas estagna por volta de 4×10^4 avaliações de função, momento em que o PSO o ultrapassa. Contudo, o próprio PSO também estagna em um mínimo local, permitindo que o GA-BBPSO se liberte e continue a busca pelo mínimo global, alcançando o melhor valor.

A função f_{16} é complexa de otimizar devido à sua superfície extremamente irregular com vários mínimos locais circulares. As curvas de convergência indicam que os valores mínimos de todos os algoritmos descem em degraus, parando temporariamente em diferentes mínimos locais até encontrarem posições melhores. O PSO converge mais rapidamente, porém estagna, enquanto o GA-BBPSO explora a superfície de maneira mais eficiente, caminhando rapidamente até o mínimo global.

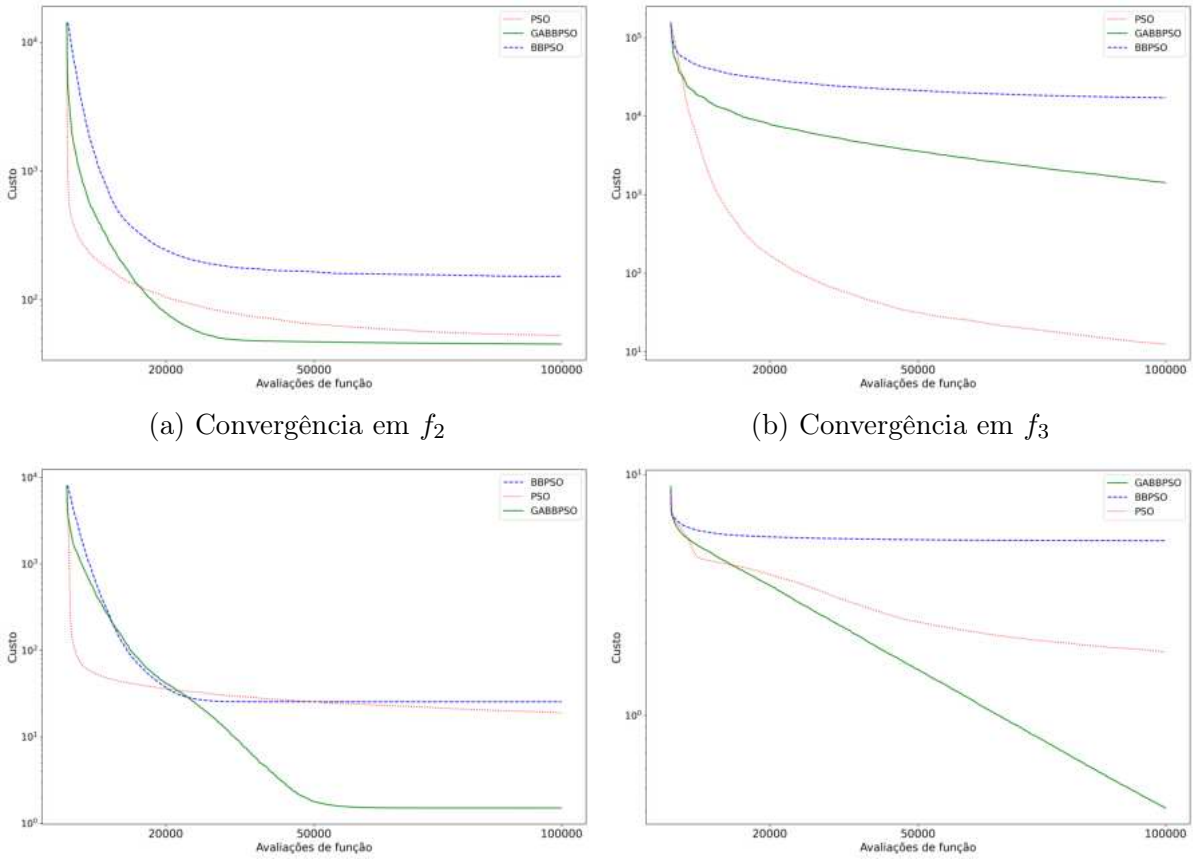
Na função f_{18}, o PSO converge rapidamente para o mínimo, enquanto o BBPSO e o GA-BBPSO alternam entre os menores valores até que o BBPSO estagne, provavelmente em um mínimo local. Devido à exploração intensificada, o GA-BBPSO consegue encontrar lentamente o mínimo global, aparentando superar o PSO, embora ocorra um empate estatístico.

Na função f_{19}, o PSO se mostra mais eficiente até o final, seguido do GA-BBPSO. Todos os métodos convergem rapidamente no início até atingirem um ponto de parada, provavelmente uma bacia de atração local, formando gráficos em formato de cotovelo. O PSO é o único que consegue escapar efetivamente desse mínimo local, provavelmente por ter encontrado o mínimo primeiro e dispender mais avaliações de função para se

libertar. O GA-BBPSO também consegue escapar, mas com menos avaliações disponíveis, não atinge a mesma proximidade do mínimo que o PSO. Em relação ao BBPSO, o GA-BBPSO, novamente devido à sua capacidade de exploração, consegue se libertar, enquanto o BBPSO fica preso no ponto de cotovelo do gráfico, sem atingir posições melhores.

Na função f_{20}, embora à primeira vista seja simples de minimizar (a função retorna o valor absoluto do maior elemento em um vetor de números reais, bastando que algum valor se aproxime de zero para atingir o mínimo), apresentou um desafio para o GA-BBPSO. Inicialmente, o GA-BBPSO supera os outros métodos, mas estagna após poucas iterações, permitindo que o PSO aproxime-se do mínimo mais rapidamente e vença no final. Contudo, é importante notar que o GA-BBPSO mostrou-se mais eficiente quando comparado ao BBPSO, que acabou preso em um mínimo local. A rotina de gerar novas posições a partir de uma distribuição gaussiana do BBPSO mostrou-se pouco eficaz para escapar desse mínimo. Provavelmente, no início, p^i_{best} e g_{best} se igualaram, zerando o desvio padrão e impedindo a atualização correta das partículas (ZHANG et al., 2011). As curvas de convergência das funções comentadas podem ser observadas nas Figuras 4, 5 e 6.

Figura 4 – Curvas de convergência para f_2, f_3, f_6 e f_7.



(a) Convergência em f_2

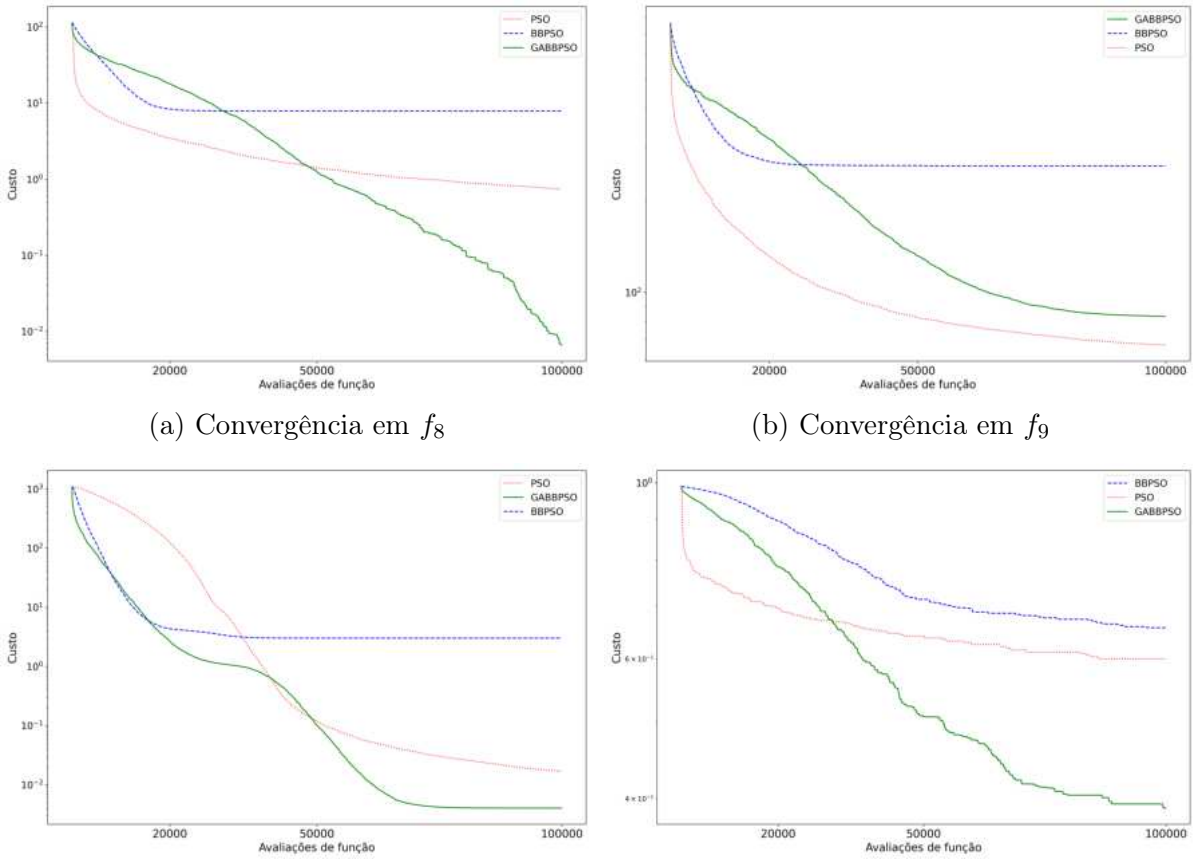
(b) Convergência em f_3

(c) Convergência em f_6

(d) Convergência em f_7

Ao comparar exclusivamente o GA-BBPSO com o BBPSO, observa-se que o primeiro foi estatisticamente mais robusto em 39 dos 42 casos, evidenciando que aumentar a ex-

Figura 5 – Curvas de convergência para f_8, f_9, f_{10} e f_{16}.



(a) Convergência em f_8

(b) Convergência em f_9

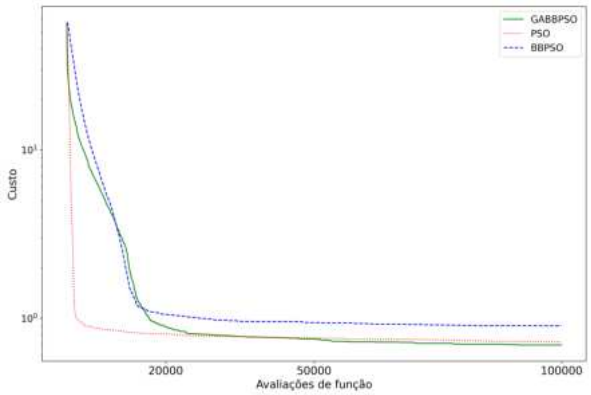
(c) Convergência em f_{10}

(d) Convergência em f_{16}

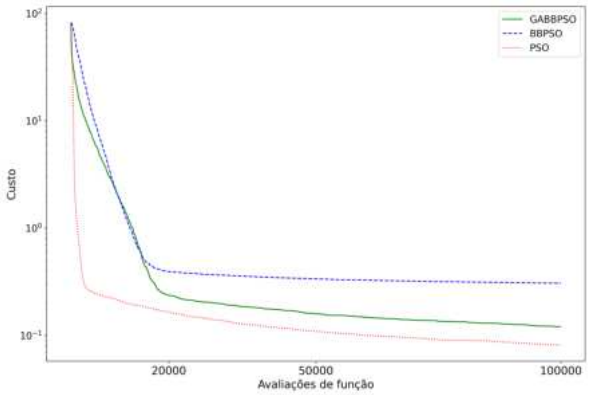
ploração e reduzir a dependência da memória pessoal foi benéfico para o algoritmo.

Com base nesses resultados, pode-se concluir, com significância estatística, que o método proposto é competitivo em relação aos algoritmos PSO e BBPSO originais, demonstrando grande potencial para aplicação em problemas reais.

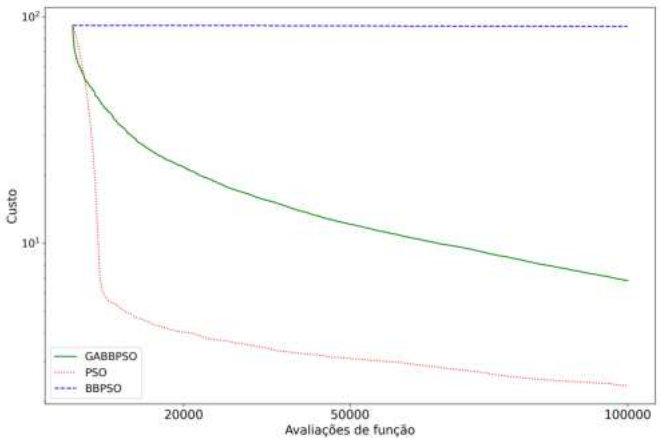
Figura 6 – Curvas de convergência para f_{18}, f_{19} e f_{20}



(a) Convergência em f_{18}



(b) Convergência em f_{19}



(c) Convergência em f_{20}

5.6 TORRES DE SINAL

O problema do posicionamento de torres de sinal, derivado do problema de localização de instalações, pode ser abordado com diferentes objetivos (FERREIRA et al., 2024). As instalações podem variar em custos, raios de alcance, velocidade de transmissão (no caso das torres) ou de atuação (no caso de ambulâncias), o que exige a definição de qual instalação será posicionada e onde, para minimizar custos, maximizar a velocidade de atendimento ou ampliar o raio de cobertura, considerando diferentes configurações de instalação.

Todas as heurísticas e funções apresentadas neste trabalho possuem um propósito mono-objetivo. Assim, o problema de posicionamento de torres de sinal abordado neste estudo foca exclusivamente em maximizar o número de usuários (células) atendidos dentro do alcance das torres.

5.6.1 Definição Formal

Vamos considerar uma malha de tamanho $n \times n$. Cada coordenada inteira desta malha representa uma célula, totalizando n^2 células. Nosso objetivo é posicionar T torres dentro desta malha, todas com o mesmo raio R de alcance, de maneira que o maior número possível de células seja coberto, minimizando a sobreposição entre as coberturas das torres.

Como nosso algoritmo foi projetado para resolver problemas de minimização, invertemos o sinal da função objetivo para transformar o problema de maximização em minimização. Além disso, adicionamos uma penalidade para soluções que possuam células com cobertura sobreposta por múltiplas torres, com o objetivo de que as heurísticas busquem soluções com menos sobreposições.

5.6.2 Variáveis

- $M = \{(x, y) \mid x, y \in \{0, 1, 2, \ldots, n-1\}\}$: Conjunto de todas as células na malha $n \times n$.
- $T = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_T\}$: Conjunto de torres, onde cada torre $\mathbf{t}_i = (x_i, y_i)$ está posicionada em coordenadas contínuas dentro da malha.
- C_i: Conjunto de células cobertas pela torre $\mathbf{t}_i$.
- C: Conjunto total de células cobertas por todas as torres.
- $o(c)$: Número de torres que cobrem a célula c.
- R: Raio (em células) que representa o alcance de cada torre.

5.6.3 Cobertura das Torres

A cobertura de uma torre $\mathbf{t}_i$ é definida como a distância Euclidiana entre cada célula e a respectiva torre (DEZA; DEZA, 2009):

$$C_i = \left\{ (x, y) \in M \;\middle|\; \sqrt{(x - x_i)^2 + (y - y_i)^2} \leq R \right\}$$

O conjunto total de células cobertas é:

$$C = \bigcup_{i=1}^{T} C_i$$

O número de torres que cobrem uma célula $c \in C$ é dado por:

$$o(c) = \left| \left\{ i \in \{1, 2, \ldots, T\} \;\middle|\; c \in C_i \right\} \right|$$

A função de custo que desejamos minimizar é composta por dois termos: a maximização do número de células cobertas e a penalização pelas sobreposições de cobertura. Formalmente, a função de custo é definida como:

$$f(T) = -|C| + \lambda \sum_{c \in C} \max(0, o(c) - 1)$$

Onde:

- $|C|$ representa o número total de células únicas cobertas pelas torres.
- λ é um fator de penalização ($\lambda = 0.1$ neste caso) que controla a importância da redução das sobreposições.
- $\sum_{c \in C} \max(0, o(c) - 1)$ soma as penalidades para cada célula que é coberta por mais de uma torre.

Além da função objetivo, deve se considerar que as coordenadas das torres devem estar dentro dos limites da malha:

$$0 \leq x_i \leq n - 1, \quad 0 \leq y_i \leq n - 1, \quad \forall i = 1, 2, \ldots, T$$

5.7 AVALIAÇÃO DOS ALGORITMOS NO PROBLEMA DAS TORRES DE SINAL

Conforme mencionado anteriormente, à medida que as instâncias do problema aumentam, a complexidade cresce exponencialmente devido à explosão combinatória. A função objetivo utilizada neste trabalho é particularmente custosa em termos computacionais, pois exige o cálculo da distância de cada célula para cada torre, resultando em uma complexidade de $O(n^2 \cdot T)$ (ARORA; BARAK, 2009). Além disso, cada método foi executado 30 vezes para possibilitar inferências estatísticas, com cada execução realizando 10^5 avaliações da função objetivo. Consequentemente, o tempo total de execução foi significativamente alto.

Devido a essas restrições, optou-se por realizar os testes em uma única configuração, com uma malha de dimensão $n \times n$, onde $n = 100$. Nessa configuração, foram consideradas 25 torres com raio de cobertura $R = 10$. Todos os métodos foram avaliados 30 vezes, seguindo o mesmo critério de parada de 10^5 avaliações da função objetivo. Os hiperparâmetros utilizados foram os mesmos dos testes anteriores. Como temos 25 torres, cada uma com coordenadas x e y, o problema apresenta 50 dimensões, o que implica no uso de 50 indivíduos na população. As métricas inferidas estão resumidas na Tabela 3.

Vale ressaltar que, no contexto deste trabalho, quanto menor o custo, melhor a configuração encontrada, uma vez que a função objetivo foi transformada de maximização para minimização.

Tabela 3 – Cobertura total por método

Função	**Dim**	**BBPSO**		**GA-BBPSO**		**PSO**	
		Média	**Desvio**	**Média**	**Desvio**	**Média**	**Desvio**
Cobertura	50	-7524.520	178.558	**-7731.693**	**115.607**	-7281.646	360.545

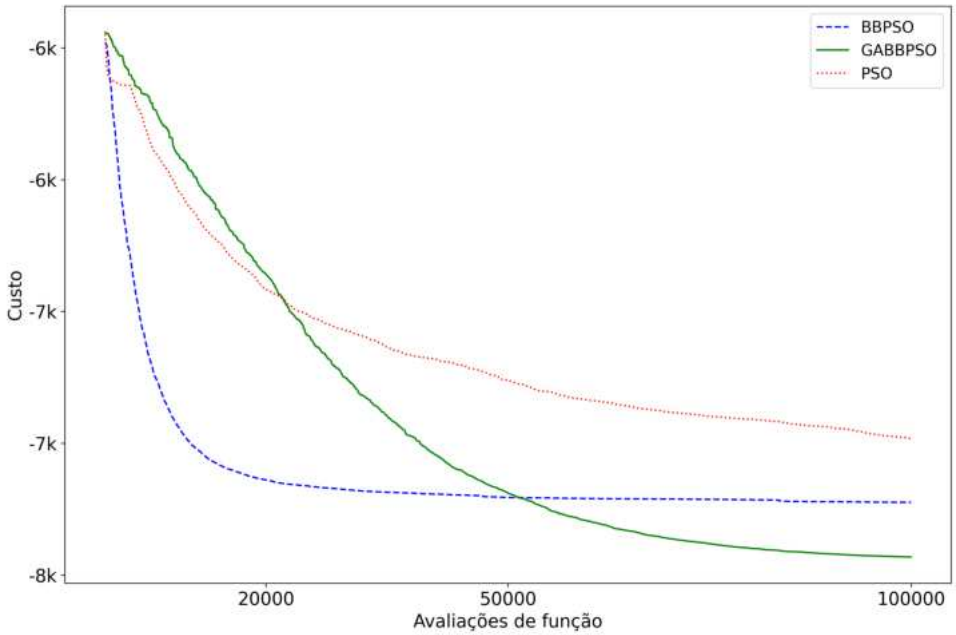
Dentro das escalas, as médias foram parecidas, para termos certeza de qual algoritmo é preferível, vamos realizar o teste de hipótese em relação a igualdade das médias novamente, com os resultados na Tabela 4.

Tabela 4 – Resultados do teste de hipótese para a função de cobertura

Função	**Dimensão**	**Comparação**	**p-valor**	**Resultado**	**Resposta**
Cobertura	50	GA-BBPSO vs PSO	2.325e-07	Rejeitar H_0	GA-BBPSO
	50	GA-BBPSO vs BBPSO	3.221e-06	Rejeitar H_0	GA-BBPSO

Assim, observamos, e temos evidência estatística para afirmarmos, que o método GA-BBPSO destacou-se como o mais eficiente no problema das torres de sinal.

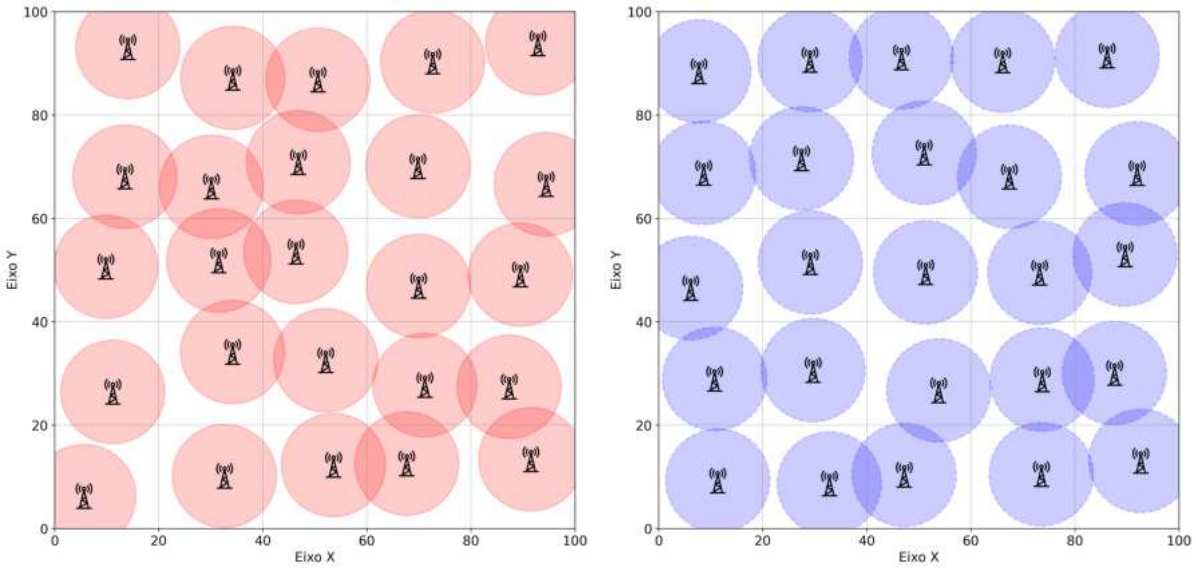
Figura 7 – Curva de convergência para a função de cobertura das torres



Analisando a convergência dos métodos na função de cobertura (Figura 7), podemos perceber que o BBPSO é o método que inicialmente converge mais rapidamente, seguido pelo PSO. No entanto, mesmo sendo o que começa a convergir mais lentamente, o GA-BBPSO evita ficar preso em mínimos locais, como ocorre com o BBPSO em aproximadamente $2 \cdot 10^4$ avaliações de função, nesta instância do problema das torres. Além disso, o GA-BBPSO mantém uma convergência eficiente até o final, ao contrário do PSO, que começa a perder eficiência por volta de $2.5 \cdot 10^4$ avaliações. Na Figura 8, é possível observar

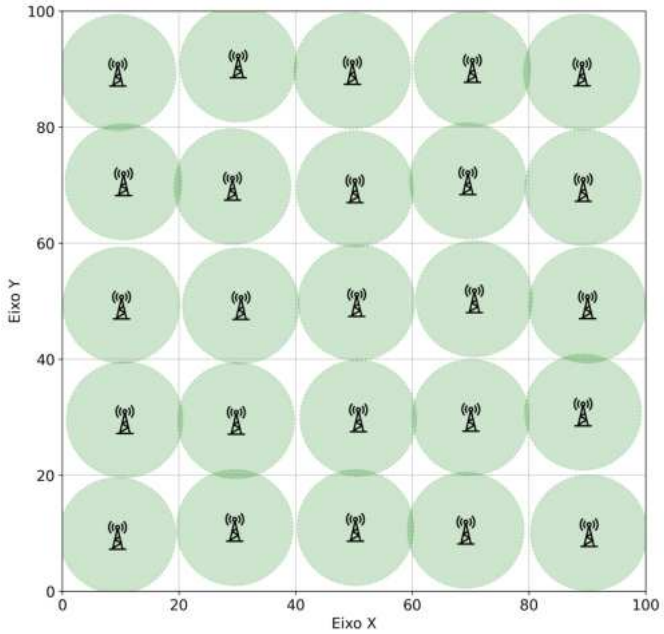
graficamente a comparação das configurações de torres. Nota-se que os outros métodos apresentaram muitas sobreposições, enquanto o GA-BBPSO produziu uma configuração com torres quase uniformemente espaçadas, minimizando as sobreposições.

Figura 8 – Configuração das torres encontradas pelos três métodos.



(a) PSO

(b) BBPSO



(c) GA-BBPSO

6 CONCLUSÃO

Este estudo explorou o acoplamento de duas heurísticas conhecidas, o Algoritmo Genético (GA) e a Otimização por Enxame de Partículas Simplificada (BBPSO), propondo o método GA-BBPSO. A combinação das capacidades de exploração aprimorada do GA com a eficiência do BBPSO resultou em um algoritmo robusto, capaz de equilibrar exploração e explotação. A aplicação do GA-BBPSO ao problema de posicionamento eficiente de torres de sinal, uma questão relevante na área de telecomunicações, demonstrou sua eficácia na minimização da sobreposição e maximização da cobertura.

Os experimentos realizados validaram a superioridade do GA-BBPSO em relação aos métodos PSO e BBPSO, como evidenciado pelas funções de benchmark e pelos testes estatísticos. Em 80% dos casos, o GA-BBPSO mostrou-se igual ou superior aos métodos comparados, sendo notavelmente mais eficiente em algumas instâncias, onde os outros métodos obtiveram resultados abaixo do esperado. Além disso, o desempenho satisfatório do GA-BBPSO, mesmo sem ajustes finos dos hiperparâmetros, destaca seu potencial para otimizações ainda mais precisas, caso sejam realizadas tunagens específicas.

A capacidade do GA-BBPSO de adaptar-se a diferentes cenários, comprovada por sua eficácia em funções unimodais e multimodais de diferentes graus de complexidade, pode indicar sua aplicabilidade prática em cenários reais. No problema das torres de sinal, o método demonstrou, com significância estatística, ser a abordagem mais eficiente, mesmo o problema sendo uma instância com dimensionalidade moderada.

Como trabalhos futuros, propõem-se as seguintes direções:

a) Realizar tunagens detalhadas dos novos hiper-parâmetros (taxas de mutação e recombinação), para maximizar a eficiência do algoritmo em diferentes cenários.

b) Aplicar o GA-BBPSO em variações do problema de posicionamento de torres, como diferentes configurações de malha ou ambientes urbanos complexos.

c) Explorar sua utilização em problemas de otimização multiobjetivo, como o balanceamento entre custo de instalação, cobertura de usuários e impacto ambiental.

d) Estender o método para outras áreas de aplicação, como planejamento logístico ou sistemas de energia.

REFERÊNCIAS

ADENSO-DIAZ, B.; RODRIGUEZ, F. A simple search heuristic for the mclp: Application to the location of ambulance bases in a rural region. **Omega**, Elsevier, v. 25, n. 2, p. 181–187, 1997.

Anatel and Teleco. **Cell Sites**. 2025. Acessado em 2025/01/01. Disponível em: https://www.teleco.com.br/en/en_erb.asp.

ARORA, S.; BARAK, B. **Computational complexity: a modern approach**. [S.l.]: Cambridge University Press, 2009.

BIANCHI, L. et al. A survey on metaheuristics for stochastic combinatorial optimization. **Natural Computing**, Springer, v. 8, p. 239–287, 2009.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

DEZA, M. M.; DEZA, E. **Encyclopedia of Distances**. [S.l.]: Springer, 2009.

DURÁN-ROSAL, A. M. et al. A hybrid dynamic exploitation barebones particle swarm optimisation algorithm for time series segmentation. **Neurocomputing**, Elsevier, v. 353, p. 45–55, 2019.

EDWARDS, R.; GLASS, L. Combinatorial explosion in model gene networks. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 10, n. 3, p. 691–704, 2000.

Engati Team. **Mathematical Optimization - Definition, Examples, and Applications**. 2025. Accessed: 2025-01-05. Disponível em: https://www.engati.com/glossary/mathematical-optimization.

FERREIRA, M. M. de O. et al. Algoritmo genético aplicado a problemas de alocação de torres de radiotransmissão. **InterSciencePlace**, v. 19, 2024.

GASPAR-CUNHA, A.; TAKAHASHI, R.; ANTUNES, C. H. **Manual de computação evolutiva e metaheurística**. [S.l.]: Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

GAVANA, A. **Global Optimization Test Functions**. 2024. Accessed: 2024-11-28. Disponível em: http://infinity77.net/global_optimization/test_functions_nd_B.html.

GENDREAU, M.; POTVIN, J.-Y. et al. **Handbook of metaheuristics**. [S.l.]: Springer, 2010. v. 2.

GITHUB. **GitHub**. 2020. Disponível em: https://github.com/.

GOLDBERG, D. E. Genetic algorithms in search, optimization, and machine learning. **Addion wesley**, v. 1989, n. 102, p. 36, 1989.

GUICHARD, D. An introduction to combinatorics and graph theory. **Whitman College-Creative Commons**, 2017.

GUO, J. et al. A bare-bones particle swarm optimization with crossed memory for global optimization. **IEEE Access**, IEEE, v. 11, p. 31549–31568, 2023.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: https://doi.org/10.1038/s41586-020-2649-2.

HOLLAND, J. H. Genetic algorithms. **Scientific american**, JSTOR, v. 267, n. 1, p. 66–73, 1992.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

ITU-R, M. **IMT Traffic Estimates for the Years 2020 to 2030**. [S.l.]: M, 2015.

JORDEHI, A. R.; JASNI, J. Parameter selection in particle swarm optimisation: a survey. **Journal of Experimental & Theoretical Artificial Intelligence**, Taylor & Francis, v. 25, n. 4, p. 527–542, 2013.

KASHYAP, R. et al. Algorithmic approach for strategic cell tower placement. In: IEEE. **2014 5th International Conference on Intelligent Systems, Modelling and Simulation**. [S.l.], 2014. p. 619–624.

KENNEDY, J. Bare bones particle swarms. In: IEEE. **Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)**. [S.l.], 2003. p. 80–87.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-international conference on neural networks**. [S.l.], 1995. v. 4, p. 1942–1948.

LACERDA, E. G. de; CARVALHO, A. D. Introdução aos algoritmos genéticos. **Sistemas inteligentes: aplicaçoes a recursos hıdricos e ciências ambientais**, v. 1, p. 99–148, 1999.

LORENA, L. A.; PEREIRA, M. A. A lagrangean/surrogate heuristic for the maximal covering location problem using hillman's edition. **International Journal of Industrial Engineering**, Citeseer, v. 9, p. 57–67, 2002.

LUENBERGER, D. G.; YE, Y. et al. **Linear and nonlinear programming**. [S.l.]: Springer, 1984. v. 2.

MARCELINO, C. G. **Uma Abordagem Evolutiva e Híbrida Para a Solução de Problemas de Fluxo de Potência Ótimo**. 140 p. Tese (Tese (Doutorado)) — CEFET-MG, Belo Horizonte, 2017.

MARCELINO, C. G. et al. Solving security constrained optimal power flow problems: a hybrid evolutionary approach. **Applied Intelligence**, Springer, v. 48, p. 3672–3690, 2018.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MICHEL, D. et al. **A Modern Introduction to Probability and Statistics**. [S.l.]: New York: Springer, 2005.

NOCEDAL, J.; WRIGHT, S. J. **Numerical optimization**. [S.l.]: Springer, 1999.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. **Combinatorial optimization: algorithms and complexity**. [S.l.]: Courier Corporation, 1998.

PIZZOLATO, N. D.; BARCELOS, F. B.; LORENA, L. A. N. School location methodology in urban areas of developing countries. **International Transactions in Operational Research**, Wiley Online Library, v. 11, n. 6, p. 667–681, 2004.

PLEVRIS, V.; SOLORZANO, G. A collection of 30 multidimensional functions for global optimization benchmarking. **Data**, MDPI, v. 7, n. 4, p. 46, 2022.

SCHRIJVER, A. et al. **Combinatorial optimization: polyhedra and efficiency**. [S.l.]: Springer, 2003. v. 24.

SÖRENSEN, K.; GLOVER, F. Metaheuristics. **Encyclopedia of operations research and management science**, Springer Boston, MA, USA, v. 62, p. 960–970, 2013.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, Springer, v. 11, p. 341–359, 1997.

STUDENT. The probable error of a mean. **Biometrika**, JSTOR, p. 1–25, 1908.

SURJANOVIC, S.; BINGHAM, D. **Virtual Library of Simulation Experiments: Optimization Test Functions and Datasets**. 2024. Accessed: 2024-11-28. Disponível em: https://www.sfu.ca/~ssurjano/optimization.html.

TAKAHASHI, R. H. Otimização escalar e vetorial. **Notas de aula**, 2007.

TALBI, E. Metaheuristics: From design to implementation. **John Wiley & Sons google schola**, v. 2, p. 268–308, 2009.

URLI, T. Hybrid meta-heuristics for combinatorial optimization. **Università degli Studi di Udine**, 2014.

VIRTANEN, P. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. **Nature Methods**, v. 17, p. 261–272, 2020.

WANG, H. et al. Gaussian bare-bones differential evolution. **IEEE Transactions on Cybernetics**, IEEE, v. 43, n. 2, p. 634–647, 2013.

WOLPERT, D. H.; MACREADY, W. G. Coevolutionary free lunches. **IEEE Transactions on evolutionary computation**, IEEE, v. 9, n. 6, p. 721–735, 2005.

ZAMBRANO-BIGIARINI, M.; CLERC, M.; ROJAS, R. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: IEEE. **2013 IEEE congress on evolutionary computation**. [S.l.], 2013. p. 2337–2344.

ZHANG, H. et al. Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data. **Fluid Phase Equilibria**, Elsevier, v. 301, n. 1, p. 33–45, 2011.