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

PATRICK PAIVA MEDEIROS DE ALBUQUERQUE

UMA ABORDAGEM HÍBRIDA PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM LARGA ESCALA

RIO DE JANEIRO
2024

PATRICK PAIVA MEDEIROS DE ALBUQUERQUE

UMA ABORDAGEM HÍBRIDA PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM LARGA ESCALA

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

2024

CIP - Catalogação na Publicação

A345a

Albuquerque, Patrick Paiva Medeiros de
Uma abordagem híbrida para solução de problemas de otimização em larga escala / Patrick Paiva Medeiros de Albuquerque. -- Rio de Janeiro, 2024.
57 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, 2024.

1. Inteligência Computacional. 2. Algoritmos Evolucionários. 3. Otimização com Metaheurísticas. 4. Otimização em Larga Escala. 5. Método de Powell. 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.

PATRICK PAIVA MEDEIROS DE ALBUQUERQUE

UMA ABORDAGEM HÍBRIDA PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM LARGA ESCALA

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 13 de dezembro de 2024

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
CAROLINA GIL MARCELINO
Data: 04/01/2025 09:18:05-0300
Verifique em https://validar.iti.gov.br

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

gov.br Documento assinado digitalmente
JOAO CARLOS PEREIRA DA SILVA
Data: 05/01/2025 12:33:31-0300
Verifique em https://validar.iti.gov.br

Prof. João Carlos Pereira da Silva
D.Sc. (IC/UFRJ)

gov.br Documento assinado digitalmente
MARIA HELENA CAUTIERO HORTA JARDIM
Data: 11/01/2025 10:51:03-0300
Verifique em https://validar.iti.gov.br

Profa. Maria Helena Cautiero Horta Jardim
D.Sc. (IC/UFRJ)

AGRADECIMENTOS

Quando eu decidi recomeçar minha vida profissional mudando para a carreira que eu sempre quis e na universidade que eu sempre sonhei, fiquei com medo de ser desencorajado, afinal, eu estaria me arriscando a tentar algo quando meu tempo já tinha passado. No entanto, minha família me apoiou em cada momento, e é graças a ela que essa jornada foi possível. Eu os amo muito.

Preciso agradecer a minha mulher, Vanessa, por ter embarcado nesse desafio comigo. Seu apoio desde o dia em que eu comecei a estudar para o vestibular, até a escrita desse documento me deram forças para continuar. Obrigado pela paciência e pelo carinho nesses anos todos.

Agradeço a minha mãe, Ana Maria, por ter me incentivado a seguir meus sonhos e a me ensinar valores que foram essenciais nesse caminho. Nunca esquecerei as nossas conversas motivadoras e os bons conselhos que saíram delas. Agradeço também ao Antiquariato e toda a sua equipe pelo suporte financeiro e por esses anos trabalhando juntos. Sem você, mãe, isso não teria sido possível.

Quero agradecer ao meu pai Jayme pelos fortes valores ensinados e todo o amor e carinho que recebi tanto dele quanto da Renata, Roberta e Leandro. Aos amigos que fiz na UFRJ, Hugo, Pedro, Diego e tantos outros eu externo minha gratidão por terem me acolhido mesmo com a diferença de idade. Nós formamos uma verdadeira tropa e sem ela, minha trilha no curso teria sido muito mais custosa. Levarei todos para a vida.

Gostaria de agradecer também a minha orientadora Profa. Carolina Gil Marcelino por todo apoio e paciência que teve comigo. Agradeço também aos colegas do Laboratório LC3 do IC/NCE/UFRJ pelo acolhimento e ao CNPq e a FAPERJ pelo investimento que possibilitou esse experimento.

RESUMO

Existem diversos problemas de otimização em larga escala que não possuem algoritmos eficientes capazes de encontrar sua solução ótima. Áreas como a de finanças, logística, biomedicina e energia possuem problemas complexos que se beneficiariam de boas soluções aproximadas. Nesse contexto, os algoritmos de otimização metaheurísticos não fornecem garantia de encontrar a solução ótima para esses problemas, mas podem obter soluções aproximadas de qualidade. Por isso, o estudo e a proposição de novos algoritmos tem ganhado muita força nos últimos anos. Dessa forma, é proposto e apresentado um novo algoritmo híbrido C-DEEPSO-Powell, mesclando o C-DEEPSO (*Canonical Differential Evolutionary Particle Swarm Optimization*) com o Método de Powell. Essa mescla se dá pelo fato do C-DEEPSO possuir características que o permitem realizar uma busca global eficiente, enquanto que o método de Powell apresenta boa performance como busca local. Para avaliação, foram realizados experimentos utilizando a *suite* de testes de *benchmark* do CEC'2013 em um computador de alto desempenho, comparando o C-DEEPSO com o C-DEEPSO-Powell. A análise desses experimentos foi feita utilizando técnicas de inferência estatística, e os resultados mostraram que o novo híbrido se mostrou competitivo, principalmente em problemas de otimização em que a função objetivo possui características de formato fortemente quadrático.

Palavras-chave: inteligência computacional; algoritmos evolucionários; hibridização; otimização com metaheurísticas; otimização em larga escala; C-DEEPSO; método de Powell.

ABSTRACT

There are several large-scale optimization problems that do not have efficient algorithms capable of finding their optimal solution. Areas such as finance, logistics, biomedicine, and energy have complex problems that would benefit from good approximate solutions. In this context, metaheuristics optimization algorithms do not provide a guarantee of finding the optimal solution for these problems, but they can obtain good approximate solutions. Therefore, the study and proposal of new algorithms has gained attention in recent years. Thus, a new hybrid algorithm C-DEEPSO-Powell is proposed and presented, combining C-DEEPSO (Canonical Differential Evolutionary Particle Swarm Optimization) with Powell's method. This combination is due to the fact that C-DEEPSO has characteristics that allow it to perform an efficient global search, while Powell's method presents good performance as a local search. For evaluation, experiments were performed using the CEC'2013 benchmark test suite on a high-performance computer, comparing C-DEEPSO with C-DEEPSO-Powell. The analysis of these experiments was performed using statistical inference techniques, and the results showed that the new hybrid was competitive, especially in optimization problems that the objective function has quadratic characteristics.

Keywords: computational intelligence; evolutionary algorithms; hybridization; optimization with metaheuristics; large-scale optimization; C-DEEPSO; Powell's method.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AE	Algorítimos Evolucionários
AG	Algoritmos Genéticos
C-DEEPSO	Canonical Differential Evolutionary Particle Swarm Optimization
CEC	Congress on Evolutionary Computation
DE	Differencial Evolution
DEEPSO	Differential Evolutionary Particle Swarm Optimization
EPSO	Evolutionary Particle Swarm Optimization
GALSP	Genetic Algorithms with Local Search Procedure
IA	Inteligência Artificial
IC	Inteligência Computacional
LSGO	Large-scale Global Optimization
LSGOP	Large-scale Global Optimization Problems
PSO	Particle Swarm Optimization

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

A Inteligência Computacional (IC) é uma subárea da Inteligência Artificial (IA) que estuda a criação de sistemas inteligentes com inspiração na natureza e no comportamento humano (CARVALHO, BRAGA e LUDERMIR, 2003 apud MARCELINO, 2017). Os Algoritmos Evolucionários (AE) são exemplos de técnicas de IC que implementam metaheurísticas inspiradas em processos biológicos como a adaptação natural dos seres vivos junto ao meio em que vivem. A ideia é que esses processos naturais, quando convertidos em algoritmos, sejam capazes de resolver problemas de alta complexidade.

Os algoritmos que seguem o padrão evolutivo geralmente possuem um formato semelhante. Eles possuem uma população de indivíduos que sofrem operações de seleção, recombinação e mutação. Essas operações alteram suas características de forma aleatória, ao longo do tempo, com o objetivo de alcançar suas versões mais adaptadas ao meio. Tais versões adaptadas, nada mais são que possíveis soluções para problemas de otimização, em que temos um conjunto de variáveis de decisão, restrições e uma função objetivo que queremos otimizar (MARCELINO, 2017).

Do ponto de vista prático do mundo real, existem diversos problemas de otimização em larga escala que se beneficiam das estratégias evolutivas. Alguns desses problemas envolvem as áreas de finanças, logística, biomedicina, energia, dentre outras. Isso ocorre pois muitos desses problemas enfrentam dificuldade em obter soluções ótimas devido a sua alta dimensionalidade e espaço de busca aumentado. Para muitos deles, não se conhecem algoritmos de otimização que encontrem soluções ótimas em um tempo razoável, por esse motivo, os algoritmos de otimização com metaheurísticas tem ganhado força como alternativa para obter boas soluções aproximadas em um tempo aceitável (BEYER; SCHWEFEL; WEGENER, 2002).

Não existe uma heurística perfeita capaz de resolver qualquer problema de otimização (PRESS, 2007). Algumas são mais adequadas para determinados tipos de problemas que outras. Nesse sentido, é comum mesclar as melhores características de cada heurística com o intuito de criar um algoritmo híbrido capaz de resolver uma gama maior de problemas. Este trabalho propõe a união de duas heurísticas já existentes em um novo algoritmo de otimização híbrido que será exposto nas próximas seções.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é estudar e propor a criação de um novo algoritmo de otimização híbrido com metaheurísticas. Ele será composto pela união de dois outros algoritmos conhecidos que são o C-DEEPSO (*Canonical Differential Evolutionary Particle Swarm Optimization* (MARCELINO et al., 2016) e o Método de Powell (POWELL, 1964). Será realizado um experimento para avaliar a sua performance utilizando uma *suite* de testes de *benchmark* consolidada. Com essa avaliação, seremos capazes de verificar se o híbrido proposto é eficiente e para quais cenários ele é útil.

1.2.2 Objetivos Específicos

- Entender os conceitos inerentes a otimização com Algoritmos Evolutivos;
- Entender o funcionamento do Método de Powell e suas limitações;
- Propor o algoritmo C-DEEPSO-Powell;
- Realizar um teste de performance utilizando a *suite* de *benchmark* do CEC'2013 (LI et al., 2013);
- Avaliar os resultados do teste e identificar o cenário em que o C-DEEPSO-Powell pode ser útil;

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em 6 capítulos. O capítulo 2 traz a fundamentação teórica necessária para a construção do novo híbrido proposto. Nele estão expostos alguns algoritmos de otimização importantes e os conceitos por traz deles. O terceiro capítulo aborda trabalhos que têm alguma relação com o proposto aqui. São tentativas anteriores de utilizar o Método de Powell neste contexto. O quarto capítulo traz a proposta do C-DEEPSO-Powell com sua estratégia e pseudocódigo. O quinto capítulo explica o experimento na *suite* de *benchmark* do CEC'2013 e avalia seus resultados. Por fim, o capítulo 6 aborda a conclusão e as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 OTIMIZAÇÃO

A otimização é um ramo da matemática e da ciência da computação que tem por objetivo encontrar os máximos ou mínimos de uma função. Para isso, ela busca encontrar valores para as variáveis independentes que maximizam ou minimizam a função (PRESS, 2007). Além disso, é desejável encontrar estes valores de forma rápida e com pouco gasto computacional, no entanto, dependendo do formato da função, este pode ser um bom desafio. Esta função, também chamada por função de custo, geralmente representa matematicamente o custo de algo no mundo real. Devemos ter clareza sobre o significado do máximo ou mínimo da função que queremos otimizar, pois o extremo a ser encontrado deve guiar, por exemplo, a construção de um sistema com máxima eficiência energética ou menor gasto financeiro (TAKAHASHI; CUNHA, 2012). A falta dessa clareza pode nos levar a uma otimização subótima ou até mesmo a resultados contrários ao nosso objetivo.

A modelagem de um problema real por vezes é um dos maiores desafios no processo de otimização. Em geral, queremos formular um problema no seguinte formato:

$$\begin{aligned} \mathbf{x}^* = \quad & \arg\min_{\mathbf{x}} f(\mathbf{x}) \\ \text{sujeito a:} \quad & \begin{cases} \mathbf{g_i}(\mathbf{x}) \leq 0, & i = 1, \ldots, r, \\ \mathbf{h_j}(\mathbf{x}) = 0, & j = 1, \ldots, p, \end{cases} \end{aligned} \tag{2.1}$$

A solução ótima é representada por $\mathbf{x}^*$, que é um vetor que quando aplicado a $f(\mathbf{x})$ encontra seu valor de mínimo. Além disso, nosso problema pode possuir restrições. Por exemplo, o máximo de capital que é possível ser investido, ou que o diâmetro da roda de um carro deve ser positivo. Esses dois exemplos são restrições de desigualdade e estão representados em (2.1) por $\mathbf{g_i}(\mathbf{x})$. Já as restrições de igualdade são aquelas em que precisamos fixar valores exatos. Essas restrições estão representadas por $\mathbf{h_j}(\mathbf{x})$ em (2.1). Com esses dados, temos então a região factível, que é o conjunto de pontos que satisfaz a todas as restrições simultaneamente (TAKAHASHI, 2007).

Para encontrar a solução ótima existem diversas técnicas e algoritmos para os mais diversos cenários. Algumas técnicas garantem que ao final retornarão a solução ótima, ou seja, são algoritmos determinísticos. Por exemplo, quando a função objetivo a ser otimizada e suas restrições são lineares é possível utilizar o *algoritmo simplex* e este sempre encontrará a solução ótima do problema (PRESS, 2007). Esta classe de problemas define um ramo da otimização que é a Programação Linear(TAKAHASHI; CUNHA, 2012). Para outros cenários, outros algoritmos são mais apropriados, dessa forma não existe um algoritmo de otimização perfeito para todos os casos.

Em problemas em que a função objetivo possui duas ou mais variáveis, existem métodos que se baseiam no cálculo do vetor gradiente para caminharem em direção ao mínimo global (PRESS, 2007). É o caso do *método de Newton* que se baseia no cálculo da matriz hessiana. No entanto, para cenários de alta dimensionalidade, este cálculo pode ser muito custoso computacionalmente.

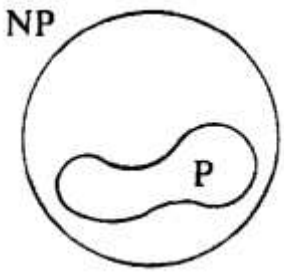
Algumas outras abordagens não utilizam derivadas por conta dessas limitações. É o caso dos chamados métodos *downhill* (ACTON, 1990), também o *método de Powell* (PRESS, 2007) que é o foco deste trabalho e as abordagens com metaheurísticas. O *método de Powell* é um algoritmo de otimização para busca do mínimo de uma função, que funciona através de sucessivas minimizações em linha em um conjunto de direções conjugadas (POWELL, 1964). Essas técnicas, no entanto, geralmente não fornecem garantia de que retornarão a solução ótima (PRESS, 2007).

2.2 METAHEURÍSTICAS

Ao longo dos últimos anos foram desenvolvidas diversas técnicas para encontrar a solução ótima de problemas com características de certa forma padronizadas, como por exemplo é o caso dos problemas que o *simplex* resolve. No entanto, no mundo real, lidamos com problemas com formatos muito diversos (TAKAHASHI; CUNHA, 2012). Alguns problemas não possuem um algoritmo determinístico até o momento conhecido e são considerados intratáveis.

Existe um ramo da ciência da computação que estuda a NP-completude dos problemas. Problemas do tipo P são aqueles em que conhecemos um algoritmo determinístico capaz de resolvê-lo em tempo polinomial. Existem também os problemas do tipo NP em que não conhecemos um algoritmo determinístico capaz de resolvê-lo em tempo polinomial, mas dada uma possível solução, somos capazes, em tempo polinomial, de verificar se a solução atende ao problema ou não (TAKAHASHI; CUNHA, 2012). Pelas características de cada tipo, podemos afirmar que $P \subseteq NP$ como no diagrama em 1 (GAREY; JOHNSON, 1979).

Figura 1 – Possível representação de P e NP



Fonte: Garey, Johnson (1979, p. 34)

Alguns problemas de classe NP são ditos NP-completos pois outros problemas NP,

através de transformações polinomiais da entrada do problema, conseguem ser convertidos neles. Muitos cientistas alegam que $P = NP$, e portanto, existem algoritmos para resolver problemas do tipo NP em tempo polinomial, nós é que ainda não os descobrimos. Caso alguém encontre um algoritmo que resolva um problema NP-completo em tempo polinomial, ele poderia ter sua entrada convertida para todos os outros problemas de classe NP (GAREY; JOHNSON, 1979).

Embora exista um enorme esforço da comunidade científica para atingir esse feito, muitos acreditam que talvez seja impossível encontrar tal algoritmo e portanto, vamos precisar conviver com os problemas NP-completos. Nesse sentido, como é intratável encontrar uma solução ótima para um problema NP, ou a tempo de ainda estarmos vivos para ver o resultado final, aplicamos as chamadas heurísticas (TAKAHASHI; CUNHA, 2012).

Algoritmos heurísticos são procedimentos que são aplicados a um problema de otimização e que buscam retornar uma boa solução, mas sem garantia de que ela seja a solução ótima, nem mesmo próximo dela. Eles são desenvolvidos para retornar resultados em um tempo menor que um algoritmo guloso ou exponencial faria com um problema NP. E como não temos garantia de que no futuro teremos uma solução determinística em tempo polinomial para problemas NP-completo, é relevante o estudo das heurísticas (TAKAHASHI; CUNHA, 2012).

Nesse sentido, a comunidade científica começou a criar heurísticas inspiradas na natureza, como é o caso do *algoritmo da colônia de formigas* e do *recozimento simulado* que possuem bases biológicas e físicas respectivamente. Normalmente, o objetivo das heurísticas é resolver classes de problemas particulares. Nos últimos anos, surgiram as metaheurísticas, que são técnicas estocásticas que utilizam heurísticas inspiradas em mecanismos de adaptação e seleção natural (MARCELINO, 2017), mas que possuem uma proposta mais geral no que tange a resolução de problemas.

2.2.1 Algoritmo Genético

Com o crescimento dos estudos em metaheurísticas baseadas em mecanismos evolutivos, surgiu uma ramificação da área de inteligência computacional chamada de Algoritmos Evolucionários (AE) (BEYER; SCHWEFEL; WEGENER, 2002). Nesse contexto, temos o surgimento dos Algoritmos Genéticos (AG).

Os Algoritmos Genéticos são metaheurísticas inspiradas na teoria da evolução apresentada por Darwin. Eles possuem dois processos básicos, seleção e a reprodução com variação. A ideia principal seria gerar indivíduos em uma população que ao longo do tempo evoluiriam e eventualmente teríamos um bom indivíduo (PEREIRA, 2012).

Uma primeira versão de um AG foi desenvolvida por John Holland nos anos 70 (HOLLAND, 1992). Em 1989, Goldberg propôs uma versão do AG com procedimentos de seleção, recombinação e mutação, responsáveis por criar novas gerações (GOLDBERG,

1989). A ideia é selecionar os melhores indivíduos conforme sua aptidão, depois recombinar esses indivíduos gerando filhos com características dos seus pais e por fim realizar mutação em algumas dessas características através de um operador de mutação. No nosso contexto de otimização, a aptidão ou *fitness* é o valor que aquele indivíduo (vetor x) possui quando aplicado a função objetivo do problema (GOLDBERG, 1989). Cada vetor (indivíduo) é formado por uma quantidade D (dimensão do vetor) de parâmetros, que em um paralelo com a biologia, compõem suas informações genéticas.

Algoritmo 1 - Pseudocódigo do algoritmo genético

```
Inicializar uma população P de N indivíduos aleatoriamente.
while condição de parada não satisfeita do
    Avaliar a aptidão de cada indivíduo em P.
    Selecionar indivíduos para a reprodução com base na aptidão, utilizando o método
  da roleta ou um torneio.
    Gerar nova população P':
    for cada par de indivíduos selecionados do
        Aplicar operador de crossover para produzir descendentes.
        Aplicar operador de mutação aos descendentes.
        Adicionar descendentes à nova população P'.
    end for
    Atualizar população P ← P'.
end while
Retornar o melhor indivíduo encontrado.
```

2.2.2 Evolução Diferencial

O Algoritmo *Differential Evolution* (DE) foi proposto por (STORN; PRICE, 1995) em 1995 e tem como base uma espécie de operador de mutação diferencial. Trata-se de um algoritmo evolutivo, embora não tenha sido inspirado em nenhum processo natural, pois apresenta o mesmo princípio de otimização com uma população que através dessas mutações diferenciais, melhora ao longo do tempo (GUIMARÃES, 2012).

A ideia principal do algoritmo é extrair dois vetores da população corrente de forma aleatória e calcular a diferença entre eles. É extraído outro vetor também de forma aleatória e essa diferença é aplicada a ele. Então, teremos uma nova solução mutante que nada mais é do que uma perturbação em algum indivíduo da população (STORN; PRICE, 1995).

Uma população de soluções pode ser expressa por $X_t = \{\mathbf{X}_{t,i};\, i = 1, \ldots, N\}$, com $\mathbf{X}_{t,i} \in \mathbb{R}^n$ em que t é o índice da geração corrente e i é o índice do indivíduo na população. Cada indivíduo pode ser expresso conforme o vetor coluna abaixo, sendo o segundo índice uma entre as n variáveis do problema:

$$\mathbf{X}_{t,i} = \begin{bmatrix} X_{t,i,1} \\ X_{t,i,2} \\ \vdots \\ X_{t,i,n} \end{bmatrix} \tag{2.2}$$

O *vetor-mutante* é expresso pela seguinte fórmula:

$$\mathbf{V}_{t,i} = \mathbf{X}_{t,r_1} + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right), \quad r_1, r_2, r_3 \in \{1, \ldots, N\} \tag{2.3}$$

Assim, $\mathbf{V}_{t,i}$ representa o *vetor-mutante* da i-ésima solução mutante. Temos $\mathbf{X}_{t,r_1}$ que é nosso *vetor base* e F sendo é um fator entre 0 e 1, iniciado pelo usuário, que é aplicado à diferença para controle do tamanho de $\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}$ que será aplicado ao *vetor base*. Aplicando a toda a população, teremos então uma população mutante $\mathbf{V}_{t,i}$ e a nossa população de teste $\mathbf{U}_{t,i}$ será calculada conforme a seguinte regra:

$$\mathbf{U}_{t,i,j} = \begin{cases} \mathbf{V}_{t,i,j}, & \text{se } \mathcal{U}_{[0,1]} \leq C \vee j = \delta_i \\ \mathbf{X}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.4}$$

sendo que $\mathcal{U}_{[0,1]}$ é um valor gerado aleatoriamente segundo uma distribuição uniforme entre 0 e 1, $\delta_i \in \{1, \ldots, n\}$ é valor sorteado aleatoriamente para que vez ou outra a igualdade $j = \delta_i$ ocorra e garanta que ao menos um dos indivíduos mutantes será aproveitado. Além disso C é um parâmetro definido pelo usuário para recombinação discreta com $C \in [0, 1]$ (GUIMARÃES, 2012).

Por fim, os vetores mutantes da população de teste $\mathbf{U}_{t,i,j}$ são avaliados na função objetivo. Os que resultarem em valores melhores que os vetores da população original, vão substituir estes, os demais não são aproveitados. Um pseudocódigo para o *Differential Evolution* de (STORN; PRICE, 1995) encontra-se no Algoritmo 2.

O algoritmo mostrado é o clássico que também é representado por **DE/rand/1/bin**. Existem outras variações desse algoritmo que são expressas no formato **DE/base/d/rec**, onde base é a forma como *vetor-base* é selecionado, d indica o número de *vetores-diferença* utilizados e *rec* é o operador de recombinação aplicado. No caso do algoritmo clássico, utilizamos o **rand** por selecionarmos o *vetor-base* de forma aleatória, $d = 1$ por utilizarmos apenas um *vetor-diferença* e o termo **bin** faz alusão a distribuição binomial que é a distribuição seguida ao aplicarmos C (GUIMARÃES, 2012). No Quadro 1 podemos ver diversas variações do algoritmo. Em algumas dessas variações foi incluído o parâmetro λ que possui a mesma lógica do fator F, mas que é iniciado pelo usuário para controlar a amplitude das diferenças em que está presente o vetor *global best* (X_{gb}).

Algoritmo 2 - Pseudocódigo do algoritmo de evolução diferencial

```
1: t ← 1
2: Inicializar população X_t = {X_{t,i}; i = 1, ..., N}
3: while algum critério de parada não for satisfeito do
4:     for i = 1 até N do
5:         Selecionar aleatoriamente r_1, r_2, r_3 ∈ {1, ..., N}
6:         Selecionar aleatoriamente δ_i ∈ {1, ..., n}
7:         for j = 1 até n do
8:             if U_[0,1] ≤ C ∨ j = δ_i then
9:                 U_{t,i,j} ← X_{t,r_1,j} + F · (X_{t,r_2,j} − X_{t,r_3,j})
10:            else
11:                U_{t,i,j} ← X_{t,i,j}
12:            end if
13:        end for
14:    end for
15:    for i = 1 até N do
16:        if f(U_{t,i}) ≤ f(X_{t,i}) then
17:            X_{t+1,i} ← U_{t,i}
18:        else
19:            X_{t+1,i} ← X_{t,i}
20:        end if
21:    end for
22:    t ← t + 1
23: end while
```

Quadro 1 – Variações de esquemas de mutação diferencial no algoritmo DE

Notação	**Mutação diferencial**
DE/rand/1/bin	$\mathbf{V}_{t,i} = \mathbf{X}_{t,r_1} + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right)$
DE/best/1/bin	$\mathbf{V}_{t,i} = \mathbf{X}_{t,\mathrm{gb}} + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right)$
DE/mean/1/bin	$\mathbf{V}_{t,i} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{X}_{t,k} + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right)$
DE/rand-to-best/1/bin	$\mathbf{V}_{t,i} = \mathbf{X}_{t,r_1} + \lambda\left(\mathbf{X}_{t,\mathrm{gb}} - \mathbf{X}_{t,r_1}\right) + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right)$
DE/current-to-best/1/bin	$\mathbf{V}_{t,i} = \mathbf{X}_{t,i} + \lambda\left(\mathbf{X}_{t,\mathrm{gb}} - \mathbf{X}_{t,i}\right) + F\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right)$
DE/rand/2/bin	$\mathbf{V}_{t,i} = \mathbf{X}_{t,r_1} + F_1\left(\mathbf{X}_{t,r_2} - \mathbf{X}_{t,r_3}\right) + F_2\left(\mathbf{X}_{t,r_4} - \mathbf{X}_{t,r_5}\right)$

Fonte: Guimarães (2012, p. 157).

2.2.3 Otimização por Enxame de Partículas

O *Particle Swarm Optimization* (PSO) é um algoritmo populacional inspirado no movimento dos pássaros ou peixes (KENNEDY; EBERHART, 1995). Através do estudo da dinâmica social desses coletivos, os autores formularam um algoritmo que basicamente gera uma população aleatória que caminha pelo espaço de busca com uma determinada velocidade. Essa velocidade é calculada a cada iteração com base em conhecimentos da própria partícula e também de seus vizinhos. Sendo assim, cada partícula aprende com suas experiências passadas e as experiências de seus vizinhos. Ao longo do tempo, todas

tendem a chegar no melhor local, que analogamente seria o ponto com mais quantidade de comida para o bando de pássaros por exemplo.

A posição de cada partícula é representada por um vetor $\mathbf{X}_{t,i} \in \mathbb{R}^n$ em que t é o índice da geração corrente, i é o índice da partícula na geração e o movimento de cada partícula outro vetor $\mathbf{V}_{t,i} \in \mathbb{R}^n$. Assim, a atualização da posição das partículas se dá pela expressão abaixo (VALLE et al., 2008).

$$\mathbf{X}_{t,i} = \mathbf{X}_{t-1,i} + \mathbf{V}_{t,i}. \tag{2.5}$$

A velocidade é composta por componentes que trazem o resultado das experiências da própria partícula e a de seus vizinhos. Sendo assim, podemos escrever a velocidade da seguinte forma:

$$\mathbf{V}_{t,i} = \varphi_{ic} \cdot \mathbf{V}_{t-1,i} + \varphi_1 \cdot \text{rand}_1 \cdot (\mathbf{X}_{best,i} - \mathbf{X}_{t-1,i}) + \varphi_2 \cdot \text{rand}_2 \cdot (\mathbf{X}_{gb} - \mathbf{X}_{t-1,i}) \tag{2.6}$$

com φ_{ic}, φ_1 e φ_2 sendo três parâmetros positivos e rand_1 e rand_2 sendo dois números gerados através de uma distribuição uniforme com intervalo $[0, 1]$ (VALLE et al., 2008).

A primeira parte da expressão da velocidade, $\varphi_{ic} \cdot \mathbf{V}_{t-1,i}$ é conhecida como inércia. Ela representa a tendência da partícula em se manter no mesma direção. É possível controlar a inércia através da constante de inércia $\varphi_i c$. A segunda parte da expressão, $\varphi_1 \cdot \text{rand}_1 \cdot (\mathbf{X}_{best,i} - \mathbf{X}_{t-1,i})$, representa o fator de cognição. Isso pois $\mathbf{X}_{best,i}$ é o *particle best*, ou seja, a melhor posição que aquela partícula esteve até o momento. É a parte da expressão que representa as experiências próprias dela. Já a terceira parte da expressão, $\varphi_2 \cdot \text{rand}_2 \cdot (\mathbf{X}_{gb} - \mathbf{X}_{t-1,i})$ é o chamado componente social. Essa componente trás a melhor experiência que o grupo conseguiu até agora. Isso ocorre pois $\mathbf{X}_{gb}$ representa a melhor posição que qualquer partícula conseguiu até agora (VALLE et al., 2008). No Algoritmo 3 temos o pseudocódigo do *Particle Swarm Optimization*.

2.2.4 DEEPSO

O PSO Canônico possui algumas limitações, como por exemplo, o fato dos hiperparâmetros precisarem ser escolhidos pelo usuário sendo que na maioria das vezes os valores ideais variam conforme o problema (MIRANDA; FONSECA, 2002). Além disso, existe a discussão sobre o decaimento da constante de inércia, em que é sugerido começar com um valor alto, para que a população caminhe livremente, e aos poucos diminuir esse peso, pois com o passar do tempo, é provável que a população esteja próxima ao ótimo global e portanto, devemos restringir o deslocamento provocado pela inércia (VALLE et al., 2008). Para Miranda e Fonseca, uma função fixa de decaimento não é o ideal, visto que é possível que o algoritmo fique preso em um mínimo local caso a inércia seja eliminada em um estágio prematuro (MIRANDA; FONSECA, 2002).

Algoritmo 3 - Pseudocódigo do particle swarm optimization (PSO)

Inicializar os parâmetros de entrada de inércia, cognição e social (φ_{ic}, φ_1 e φ_2)
Inicializar população de partículas com posições X_i e velocidades V_i
$t \leftarrow 1$
Para cada partícula i, inicializar melhor posição pessoal $X_{best,i} = X_i$
Definir o melhor global X_{gb} como a melhor posição de todas as partículas
while critério de parada não for satisfeito **do**
    for cada partícula i **do**
        Calcular o valor da função objetivo $f(X_{t,i})$
        if $f(X_{t,i}) < f(X_{best,i})$ **then**
            Atualizar melhor posição pessoal $X_{best,i} \leftarrow X_{t,i}$
        end if
        if $f(X_{best,i}) < f(X_{gb})$ **then**
            Atualizar melhor global $X_{gb} \leftarrow X_{best,i}$
        end if
        Atualizar a velocidade da partícula:

$$\mathbf{V}_{t,i} = \varphi_{ic} \cdot \mathbf{V}_{t-1,i} + \varphi_1 \cdot \text{rand}_1 \cdot (\mathbf{X}_{best,i} - \mathbf{X}_{t-1,i}) + \varphi_2 \cdot \text{rand}_2 \cdot (\mathbf{X}_{gb} - \mathbf{X}_{t-1,i})$$

        Atualizar a posição da partícula:

$$\mathbf{X}_{t,i} = \mathbf{X}_{t-1,i} + \mathbf{V}_{t,i}$$

    end for
    $t \leftarrow t + 1$
end while

Nesse sentido, foi proposto o *Evolutionary Particle Swarm Optimization* (EPSO) que tem por objetivo trazer aspectos da programação evolutiva para o PSO. O EPSO passou a realizar uma mutação nos hiperparâmetros e também no *global best* (MIRANDA; FONSECA, 2002). Assim, temos uma nova fórmula de velocidade, atualizada em relação à do PSO, com o seguinte formato:

$$\mathbf{V}_{t,i} = w^*_{i0}\mathbf{V}_{t,i} + w^*_{i1}(\mathbf{X}_{best,i} - \mathbf{X}_{t,i}) + \mathbf{P}[w^*_{i2}(\mathbf{X}^*_{gb} - \mathbf{X}_{t,i})] \tag{2.7}$$

com seus pesos de inércia, cognição e social (w_{i0}, w_{i1} e w_{i2}) sendo atualizados pela fórmula:

$$w^*_{ik} = w_{ik} + \tau\mathcal{N}(0,1) \tag{2.8}$$

sendo $\mathcal{N}(0,1)$ uma variável aleatória Normal com média 0 e variância 1. Já o *global best*, é mutado através da expressão:

$$\mathbf{X}^*_{gb} = \mathbf{X}_{gb}(1 + \tau'\mathcal{N}(0,1)) \tag{2.9}$$

com ambos τ, τ' parâmetros de aprendizado que podem ser fixados pelo usuário ou não, de acordo com a estratégia de mutação. Eles servem para controlar a magnitude da

perturbação aplicada aos pesos ou ao *global best* (MIRANDA; ALVES, 2013).

P é uma matriz diagonal binária, que possui 1 com probabilidade p e 0 com probabilidade $(1 - p)$. Como ela multiplica o componente social, este passa a ser um controle se a informação do *swarm* será passada para aquela partícula ou não, conferindo maior diversidade a população ao longo das iterações.

Bons resultados foram obtidos na comparação do EPSO com o PSO (MIRANDA; FONSECA, 2002). Dessa forma, Vladimiro Miranda e Rui Alves propuseram um novo híbrido, que incorporasse características do *Differential Evoluttion* (MIRANDA; ALVES, 2013).

O DEEPSO é composto por características de três algoritmos, sendo DE-EA-PSO uma forma de visualizar essa formação com o *Differential Evolution*, *Evolutionary Algorithm* e o *Particle Swarm Optimization*.

Para incluir a dinâmica do *Differential Evolution*, o DEEPSO alterou sua fórmula para o cálculo do vetor velocidade. Na parte cognitiva, era utilizado o vetor posição corrente e o vetor com a melhor posição que aquela partícula conseguiu até aquele momento (*particle past best*). No DEEPSO, diferentes estratégias podem ser usadas para escolher qual vetor assumiria o papel do *particle past best*, conforme abaixo (MIRANDA; ALVES, 2013):

1. **DEEPSO Sg** (*same generation sampling*):

$$\mathbf{V}_{t+1,i} = w^*_{i0}\mathbf{V}_{t,i} + w^*_{i1}(\mathbf{X}_{t,r_1} - \mathbf{X}_{t,i}) + \mathbf{P}[w^*_{i2}(\mathbf{X}^*_{gb} - \mathbf{X}_{t,i})] \tag{2.10}$$

onde $\mathbf{X}_{t,r_1}$ é amostrado uma vez da geração atual.

2. **DEEPSO Sg-rnd**: o mesmo que o tipo 1, mas com $\mathbf{X}_{t,r_1}$ reamostrado na geração atual para cada componente de $\mathbf{V}$.

3. **DEEPSO Pb** (*past bests sampling*):

$$\mathbf{V}_{t+1,i} = w^*_{i0}\mathbf{V}_{t,i} + w^*_{i1}(\mathbf{X}_{best,r_1} - \mathbf{X}_{t,i}) + \mathbf{P}[w^*_{i2}(\mathbf{X}^*_{gb} - \mathbf{X}_{t,i})] \tag{2.11}$$

onde $\mathbf{X}_{best,r_1}$ é amostrado uma vez dos melhores passados *particle past bests*.

4. **DEEPSO Pb-rnd**: o mesmo que o tipo 3, mas com $\mathbf{X}_{best,r_1}$ reamostrado entre os melhores passados *particle past bests* para cada componente de $\mathbf{V}$.

2.2.5 C-DEEPSO

O C-DEEPSO, *Canonical Differential Evolutionary Particle Swarm Optimization* foi proposto como uma melhoria ao DEEPSO (MARCELINO et al., 2016). Trata-se também de uma metaheurística populacional que utiliza operadores de mutação, recombinação

e seleção para gerar novos indivíduos (MARCELINO, 2017). Ele buscou aprimorar o DEEPSO trazendo maior proximidade com o *Differential Evolution*. O DEEPSO é um algoritmo que possui características do PSO com seleção e mutação. No entanto, ele possui pouca ligação com o DE tradicional. É possível verificar essa distinção quando olhamos para a equação de velocidade do DE (2.3). Nela podemos ver que três vetores são usados, enquanto que no DEEPSO apenas dois vetores são usados (2.10, 2.11). Assim, o C-DEEPSO trouxe novas versões da equação de velocidade que são:

$$V_t = w_T^* \times V_{t-1} + w_A^* \times (X_{\text{best}} + F \times (X_r - X_{t-1})) + w_C^* \times C \times (X_{gb}^* - X_{t-1}) \quad (2.12)$$

$$V_t = w_T^* \times V_{t-1} + w_A^* \times (X_{\text{best}} + F \times (X_{t-1} - X_r)) + w_C^* \times C \times (X_{gb}^* - X_{t-1}) \quad (2.13)$$

em que t é a geração atual, X é o vetor posição, X_{best} é o melhor vetor posição que aquela partícula obteve até aquela geração, X_{gb} é o vetor o melhor vetor posição que qualquer uma das partículas conseguiu até o momento, V_t é a velocidade do indivíduo, C é a matriz diagonal binária nos moldes do proposto no EPSO (MIRANDA; FONSECA, 2002) e por fim w_T, w_A e w_C são os pesos de inércia, assimilação (cognição) e comunicação (social) respectivamente. Os elementos marcados com (*) indicam que são sujeitos a mutação durante a execução do algoritmo (MARCELINO et al., 2016).

As equações (2.12) e (2.13) implementam a estratégia **DE/best/1/bin** sendo que a Eq. (2.12) é usada quando $X_r > X_{t-1}$ e a Eq. (2.13) quando $X_{t-1} > X_r$.

O C-DEEPSO também incorpora um novo método de amostragem do X_r, além dos quatro propostos pelo DEEPSO. O $S_g P_b - rnd$ proposto é uma combinação de $S_g rnd$ e $P_b rnd$, sendo feita uma recombinação uniforme de diferentes soluções para então obter um novo X_r (MARCELINO, 2017).

Em um trabalho posterior, foi proposta a utilização da estratégia evolutiva *current-to-best* (HUANG; QIN; SUGANTHAN, 2006) e uma reformulada equação de velocidade passou a ser (MARCELINO, 2017):

$$V_t = w_T^* V_{t-1} + w_A^*(X_{\text{st}} - X_{t-1}) + w_C^* C(X_{gb}^* - X_{t-1}). \quad (2.14)$$

em que X_{st} é gerado conforme a estratégia evolutiva **DE/current-to-best/1/bin** e é expresso por:

$$X_{\text{st}} = X_r + F(X_{\text{best}} - X_r) + F(X_{r_1} - X_{r_2}). \quad (2.15)$$

Uma vez que X_{st} é obtido, seu fitness é comparado ao de X_r. Caso, o fitness de X_{st} seja pior que o de X_r, ele então recebe o valor de X_r para que seja feito o cálculo da velocidade. O pseudocódigo do C-DEEPSO é mostrado no Algoritmo 4.

Algoritmo 4 - Pseudocódigo do algoritmo C-DEEPSO

Início
Passo 1: Iniciar os parâmetros de controle do C-DEEPSO - Tamanho da população N_P, Taxa de Mutação τ, Taxa de Comunicação P, Tamanho da Memória M_B, N_{gl}, dada uma taxa γ de ocorrência e Dimensão do problema (D).
Passo 2: Iniciar o contador de gerações $t = 1$ e iniciar aleatoriamente a população de N_P partículas, conforme $P_t = \{X_{1,t}, \ldots, X_{N_P,t}\}$ com $X_{i,t} = [x_{i,t,1}, x_{i,t,2}, \ldots, x_{i,t,D}]$, em que cada partícula está distribuída uniformemente no intervalo $[X_{\min}, X_{\max}]$, no qual $X_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$ e $X_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$ com $i = 1, 2, \ldots, N_P$.
Passo 3: Avaliar a população corrente, P_t.
Passo 4: Atualizar a melhor solução obtida até o momento, X_{gb}.
while o critério de parada não é satisfeito **do**
    for cada partícula i pertencente à população P_t **do**
        Calcular X_r usando a estratégia $S_{gPb\text{-rnd}}$;
        Mutar os parâmetros estratégicos w_T, w_A, w_C usando a Equação (2.8);
        Mutar X^*_{gb} usando a Equação (2.9);
        Aplicar regra de movimento à partícula corrente X_t via Equação (2.14);
        Avaliar a partícula corrente X_t;
        Selecionar a partícula com melhor aptidão para fazer parte da próxima população $(N_P + 1)$. **Usando por exemplo Torneio Estocástico*
    end for
    Atualizar o melhor indivíduo X_{gb} e a memória M_B;
    $t = t + 1$
end while
Fim

2.3 MÉTODO DE POWELL

O método de Powell, ou método das direções conjugadas de Powell, é um algoritmo de otimização para encontrar o mínimo de uma função sem a necessidade de conhecer suas derivadas (BRENT, 1973). Ele foi proposto por M. J. D. Powell em 1964 (POWELL, 1964). A ideia do algoritmo é que através de um conjunto de vetores direção, o algoritmo executará uma minimização em linha na primeira direção e encontrará um ponto. Desse ponto, ela partirá para a próxima direção, também minimizando em linha até o próximo ponto. A proposta de Powell é que a cada iteração, o algoritmo estaria mais próximo do mínimo global da função (PRESS, 2007).

O conjunto escolhido de vetores direção é de extrema importância para o método de Powell. As direções devem ser linearmente independentes e conjugadas para que ele funcione de forma apropriada. Dois vetores u e v quaisquer são ditos conjugados em relação a uma matriz simétrica A se

$$u^T \mathbf{A} v = 0. \tag{2.16}$$

Dessa forma, podemos dizer que u e v são direções conjugadas (BRENT, 1973). E um conjunto também será, caso todas as direções sejam conjugadas dois a dois. O movimento em um desses vetores pode ser expresso em termos do vetor gradiente pela expressão

$$\delta(\nabla f) = \mathbf{A} \cdot (\delta \mathbf{x}) \tag{2.17}$$

em que A é a matriz hessiana da função f que estamos lidando. No contexto da minimização, após nos movermos para o mínimo ao longo da direção do vetor u e em seguida mudarmos para a direção do vetor v, a única forma de não estragarmos o progresso obtido ao longo de v é que essa mudança mantenha o vetor gradiente perpendicular a u, ou seja, precisamos que

$$0 = u \cdot \delta(\nabla f) = u \cdot \mathbf{A} \cdot v \tag{2.18}$$

se mantenha, o que é o mesmo da equação (2.16), portanto, ambos devem ser conjugados.

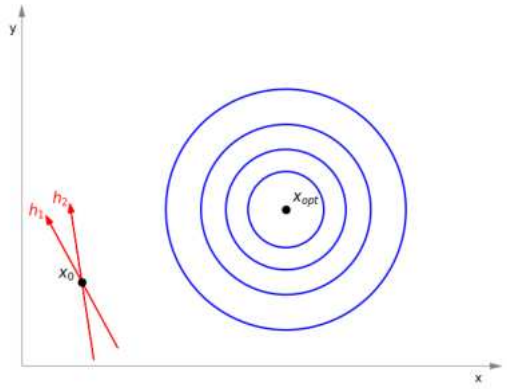
Powell formulou então o Algoritmo 5 de forma que ele fosse gerando novas direções conjugadas conforme o passar das iterações, para que em dado momento, chegássemos ao mínimo. A ideia para continuar gerando novos vetores direção é a de ao final de uma iteração, apagar o primeiro vetor e incluir ao final um novo que seria a diferença do último pelo primeiro. Assim, a cada iteração teríamos um novo conjunto de direções conjugadas (POWELL, 1964).

Algoritmo 5 - Pseudocódigo do método de Powell básico

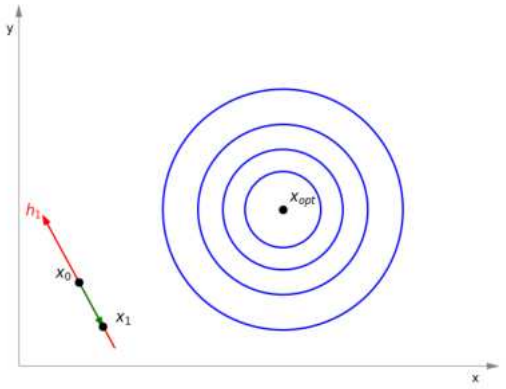
Inicialização: Definir o conjunto de direções $\mathbf{u}_i$ para os vetores base $\mathbf{e}_i$, onde $i = 0, \ldots, N-1$.
while a função objetivo continuar decrescendo **do**
    Salvar a posição inicial como $\mathbf{P}_0$.
    for $i = 0, \ldots, N-1$ **do**
        Mover $\mathbf{P}_i$ para o mínimo ao longo da direção $\mathbf{u}_i$ e chamar este ponto de $\mathbf{P}_{i+1}$.
    end for
    for $i = 0, \ldots, N-2$ **do**
        Definir $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$.
    end for
    Definir $\mathbf{u}_{N-1} \leftarrow \mathbf{P}_N - \mathbf{P}_0$.
    Mover $\mathbf{P}_N$ para o mínimo ao longo da direção $\mathbf{u}_{N-1}$ e chamar este ponto de $\mathbf{P}_0$.
end while

Na Figura 2 temos uma sequência que representa a execução do método de Powell em um problema de duas dimensões, para melhor entendimento.

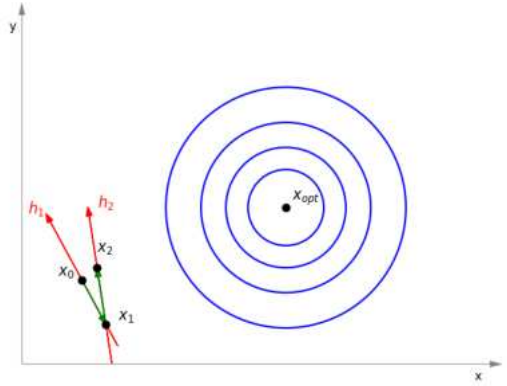
Figura 2 – Ilustração de uma execução do método de Powell



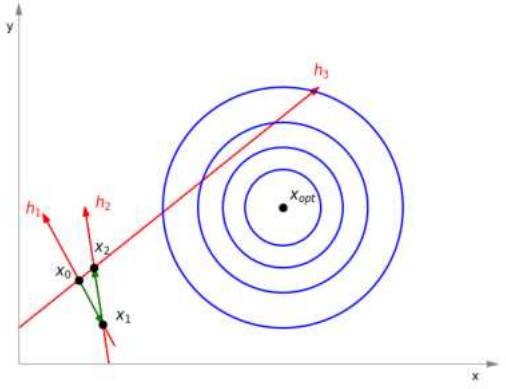
(a) Escolhemos x_0 e duas direções diferentes h_1 e h_2.



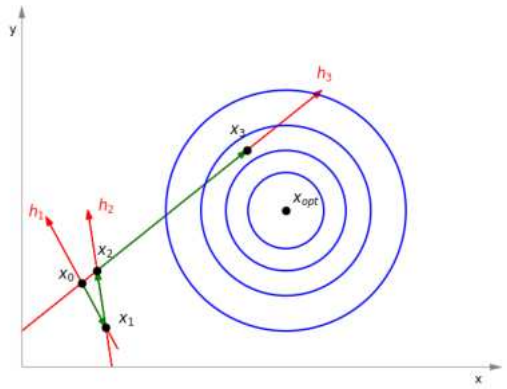
(b) A partir de x_0, executamos minimização em linha ao longo de h_1 para encontrar o extremo x_1.



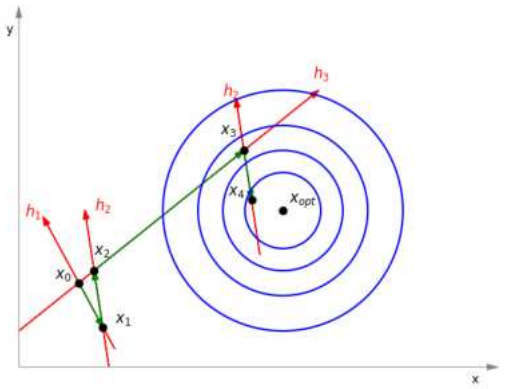
(c) A partir de x_1, executamos minimização em linha ao longo de h_2 para encontrar o extremo x_2.



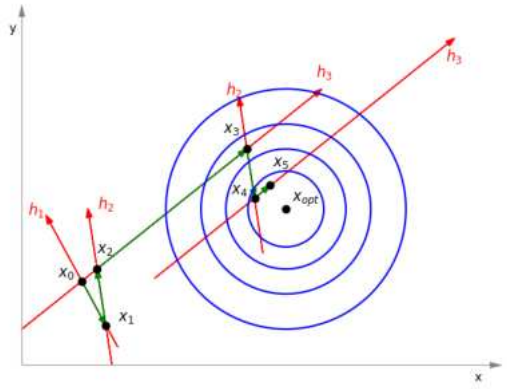
(d) Criamos uma nova direção h_3 conectando x_0 e x_2.



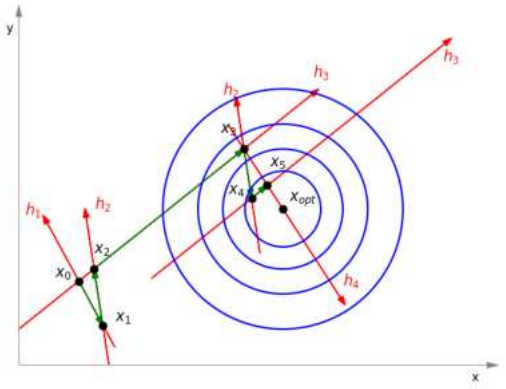
(e) A partir de x_2, executamos minimização em linha ao longo de h_3 para encontrar o extremo x_3.



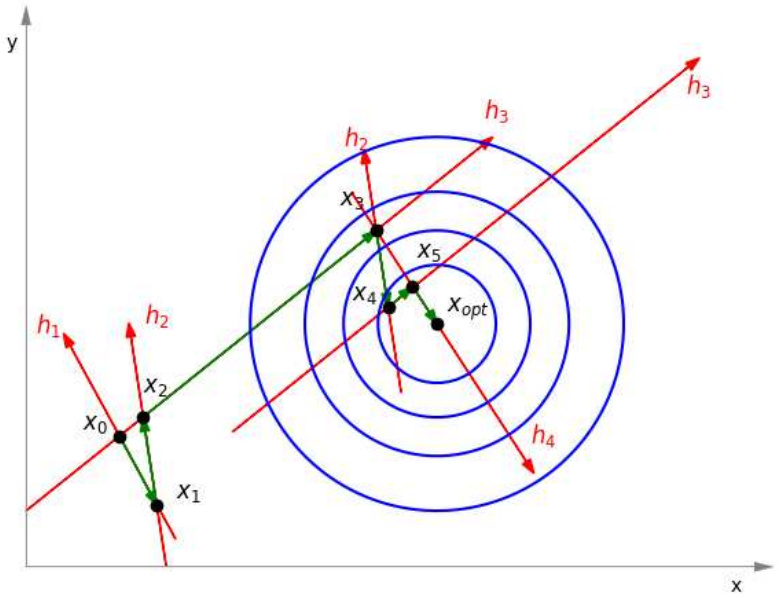
(f) A partir de x_3, executamos minimização em linha ao longo de h_2 para encontrar o extremo x_4.



(g) A partir de x_4, executamos minimização em linha ao longo de h_3 para encontrar o extremo x_5.



(h) Criamos uma nova direção h_4 conectando x_3 e x_5.



(i) A partir de x_5, executamos minimização em linha ao longo de h_4 para encontrar o ponto ótimo x_{opt}.

Para funções objetivo com formato quadrático, o método de Powell é executado em um número k de iterações, produzindo um conjunto de direções u_i em que as últimas k direções são mutuamente conjugadas. Além disso, a execução desse algoritmo garante, para funções de formato quadrático, a convergência em uma quantidade finita de iterações (BRENT, 1973), como é o caso do exemplo da Figura 2. Para as funções que não possuem formato quadrático, não existe essa garantia, embora o método consiga se aproximar bastante do mínimo da função (PRESS, 2007).

Um problema do método de Powell básico é que para dimensões maiores, eventualmente acabamos por gerar um conjunto com vetores linearmente dependentes. Isso ocorre

por conta da estratégia de eliminar o vetor mais antigo e adicionar um novo construído pela diferença do último com o primeiro mostrado nas linhas 7 a 10 do Algoritmo 5. Algumas estratégias geralmente são aplicadas para resolver esse problema. Dentre elas é possível resetar o conjunto de vetores direção a cada N iterações, por exemplo. Outra forma usada seria uma heurística capaz de gerar novas boas direções sem necessariamente serem conjugadas (PRESS, 2007). Este trabalho implementa a segunda opção.

A modificação proposta no método de Powell basicamente remove a direção em que o algoritmo teve a melhor minimização, ao invés da mais antiga (ACTON, 1990). Isso parece contraintuitivo em um primeiro momento, dado que vamos descartar a direção em que tivemos o melhor resultado, no entanto, justamente por ela apresentar uma queda maior no valor da função objetivo, é que ela provavelmente será um grande componente da nova direção a ser criada, o que aumentaria as chances desse novo vetor gerar uma dependência linear indesejada. Além disso, nem sempre é ideal descartarmos uma direção e gerar uma nova. Em alguns cenários pode ser interessante repetir a próxima iteração com o mesmo conjunto de direções. Assim, a ideia é fazermos dois testes antes de decidir o que fazer com o conjunto de direções para a próxima iteração. Para esses testes, vamos usar os fitness abaixo:

$$f_1 = f(\mathbf{p}_0) \tag{2.19}$$

$$f_2 = f(\mathbf{p}_n) \tag{2.20}$$

$$f_3 = f(2\mathbf{p}_n - \mathbf{p}_0) \tag{2.21}$$

em que os pontos $\mathbf{p}_0$ e $\mathbf{p}_n$ são os pontos da primeira minimização em linha e da última ocorridos naquela iteração, respectivamente. O terceiro ponto $(2\mathbf{p}_n - \mathbf{p}_0)$ ou de forma mais objetiva $\mathbf{p}_n + (\mathbf{p}_n - \mathbf{p}_0)$ é colinear com os outros dois e representa algo como uma extensão a partir de $\mathbf{p}_n$ com um tamanho $\mathbf{p}_n - \mathbf{p}_0$. Considerando que Δ é o valor referente ao maior decréscimo em alguma das direções daquela iteração, os testes então são:

$$f_3 \geq f_1 \tag{2.22}$$

e/ou

$$2(f_1 - 2f_2 + f_3)[(f_1 - f_2) - \Delta]^2 \geq \Delta(f_1 - f_3)^2. \tag{2.23}$$

Se alguma dessas condições for satisfeita, nós vamos manter o nosso conjunto atual de vetores direção para a próxima iteração com $\mathbf{p}_n$ assumindo como novo $\mathbf{p}_0$. A equação (2.22) nos diz que a direção $2\mathbf{p}_n - \mathbf{p}_0$ estaria de certa forma exaurida, dado que o *fitness* dessa extensão, que é como se fosse uma estimativa do que está por vir, é maior que o que temos atualmente. Portanto, não é interessante incluir essa direção no conjunto. A segunda utiliza uma aproximação da segunda derivada, junto com uma medida da

discrepância entre f_1 e f_2. Assim o lado esquerdo é um tipo de aproximação para um valor que representa a contribuição das direções atuais para o processo de otimização. O lado direito faz uma diferença ponderada por δ de f_1 e f_3 para tentar aproximar essa contribuição caso sigamos com a nova direção, portanto, se a condição for satisfeita, é forte o indício de que devemos manter as direções atuais (ACTON, 1990). O Algoritmo 6 mostra método de Powell com essas modificações.

Algoritmo 6 - Pseudocódigo do método de Powell modificado

Require: $funcao$, x_0, max_iter
1: $n \leftarrow$ tamanho de x_0
2: $direcoes \leftarrow$ matriz identidade $n \times n$
3: $x \leftarrow x_0$
4: $f_0 \leftarrow funcao(x)$
5: $f_{ret} \leftarrow f_0$
6: $k \leftarrow 0$
7: **while** $(k < max\_iter)$ **do**
8: $x_{anterior} \leftarrow x$
9: $f_{anterior} \leftarrow f_{ret}$
10: $delta \leftarrow 0$
11: $indice\_maior\_decrescimo \leftarrow 0$
12: **for** i de 1 até n **do**
13: $f_{aux} \leftarrow f_{ret}$
14: $direcao \leftarrow direcoes[i]$
15: $x, \_, f_{ret} \leftarrow$ MinimizacaoEmLinha($funcao$, x, $direcao$)
16: $decrescimo \leftarrow f_{aux} - f_{ret}$
17: **if** $decrescimo > delta$ **then**
18: $delta \leftarrow decrescimo$
19: $indice\_maior\_decrescimo \leftarrow i$
20: **end if**
21: **end for**
22: $nova\_direcao \leftarrow x - x_{anterior}$
23: **if** $nova\_direcao = 0$ **then**
24: **Break**
25: **end if**
26: $x_{extrapolado} \leftarrow x \times nova\_direcao$
27: $f_{ext} \leftarrow$ funcao($x_{extrapolado}$)
28: **if** $f_{ext} < f_{anterior}$ baseado na equação (2.22) **then**
29: $t \leftarrow$ Cálculo do teste de direção baseado na equação (2.23)
30: **if** $!t$ **then**
31: $x, \_, f_{ret} \leftarrow$ MinimizacaoEmLinha($funcao$, x, $nova\_direcao$)
32: Substituir a direção com maior redução em $direcoes$
33: Atualizar $direcoes$ com $nova\_direcao$
34: **end if**
35: **end if**
36: Incrementar k
37: **end while**

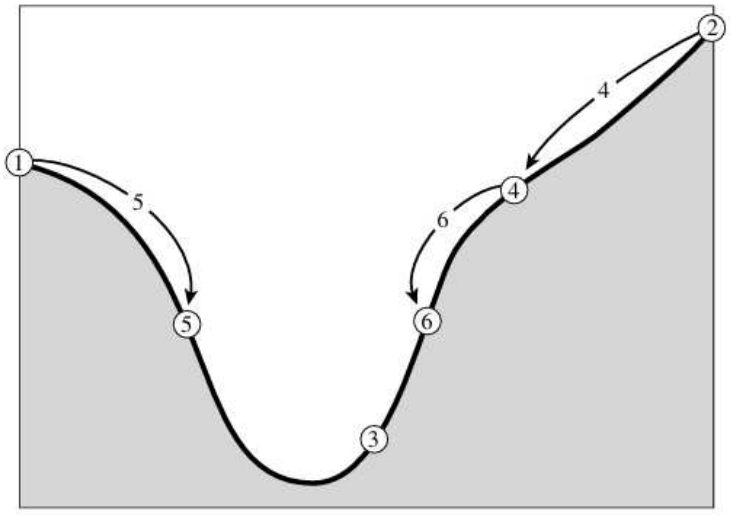
2.3.1 Minimização em Linha

O método de Powell é baseado em diversas minimizações em linha ao longo dos vetores de direção. Como essa minimização é peça chave, julgamos ser útil entender como ela é feita neste trabalho. Uma das principais formas de minimização em linha utilizadas na literatura que envolve o método de Powell é a busca por seção áurea (*golden section search*) que será explicada na próxima seção.

2.3.1.1 Seção Áurea

A busca por seção áurea foi descoberta por Kiefer em 1953 (KIEFER, 1953). Este método possui grande similaridade com o método da busca binária (*bisection method*). Para uma função unimodal escolhemos três pontos x_1, x_2 e x_3 e avaliamos o *fitness* de cada um deles. Os três então são ordenados de forma que o ponto central seja o com menor valor de $f(x)$. Escolhemos então um quarto ponto x_4 que estará contido dentro do maior dois dois intervalos. Vamos supor que x_4 estará entre x_2 e x_3. O valor de $f(x_4)$ é avaliado e caso $f(x_2) < f(x_4)$ então sabemos que o mínimo está entre x_1 e x_4, portanto alteremos os três pontos para que sejam x_1, x_2 e x_4. Caso $f(x_2) > f(x_4)$, sabemos que o mínimo está entre x_2 e x_3, e portanto o novo intervalo seria x_2, x_4 e x_3. Esse procedimento continua até que o intervalo seja muito pequeno, tão quanto uma tolerância que pode ser definida pelo usuário.

Figura 3 – Execução da seção áurea



Fonte: Press, W. H. (2007, p. 493)

Sobre a Figura 3, temos o processo da seção áurea que consiste em sucessivas delimitações do intervalo onde o mínimo está localizado (PRESS, 2007). Inicialmente, o mínimo é delimitado pelos pontos 1, 3 e 2. Em seguida, o ponto 4 é avaliado e substitui o ponto 2. Depois, o ponto 5 substitui o ponto 1 e, por fim, o ponto 6 substitui o ponto 4. A regra

adotada é manter sempre um ponto central que tenha valor menor que os dois pontos externos. Após essas etapas, o mínimo passa a ser delimitado pelos pontos 5, 3 e 6.

O que diferencia a busca por seção áurea da busca binária é a forma como escolhemos o novo ponto. Na busca binária, os intervalos vão sendo divididos pela metade em cada iteração. Para a busca por seção áurea, usamos uma estratégia diferente. Consideremos o intervalo (a, b, c). Queremos dividir este intervalo de forma a minimizar o erro na busca do ponto de mínimo. Vamos supor que b seja uma fração w do espaço entre a e c:

$$\frac{b-a}{c-a} = w \quad \frac{c-b}{c-a} = 1 - w \tag{2.24}$$

O nosso novo ponto x também seria uma fração adicional z, além de b:

$$\frac{x-b}{c-a} = z \tag{2.25}$$

dessa forma, o novo segmento de intervalo teria tamanho $w + z$ ou $1 - w$. Como queremos que os intervalos tenha tamanhos iguais, temos que:

$$z = 1 - 2w \tag{2.26}$$

Nesse sentido, considerando que x deve ficar no maior segmento, falta definir exatamente onde. Caso z seja ótimo, então w foi previamente o ótimo. Isso implica que x tenha a mesma fração de segmento que a fração de b para c, da mesma forma que foi para b a fração de segmento de a para c. Portanto, temos:

$$\frac{z}{1-w} = w \tag{2.27}$$

e, das equações (2.26) e (2.27), temos a equação quadrática abaixo que nos dá:

$$w^2 - 3w + 1 = 0 \quad \text{resultando em} \quad w = \frac{3-\sqrt{5}}{2} \approx 0.38197. \tag{2.28}$$

Assim, a divisão ótima do intervalo a, b, c possui seu ponto central em b com uma fração de distância 0.38197 do lado a e 0.61803 do lado b. Portanto, em cada iteração da busca por seção áurea, o próximo ponto a ser escolhido estará a uma fração de 0.38197 do ponto central em direção ao maior segmento (PRESS, 2007). O algoritmo 7 mostra o pseudocódigo da busca por seção áurea.

Algoritmo 7 - Busca pela Seção Áurea

Require: Função $f(x)$, intervalo inicial $[a, c]$, precisão desejada ϵ

Definir $w = \frac{3-\sqrt{5}}{2} \approx 0.38197$ ▷ Proporção áurea
Calcular os pontos internos:
$x_1 = a + (1 - w) \times (c - a)$
$x_2 = a + w \times (c - a)$
Avaliar $f_1 = f(x_1)$ e $f_2 = f(x_2)$
while $(c - a) > \epsilon$ **do**
    if $f_1 < f_2$ **then**
        $c \leftarrow x_2$
        $x_2 \leftarrow x_1$, $f_2 \leftarrow f_1$
        $x_1 = a + (1 - w) \times (c - a)$
        Avaliar $f_1 = f(x_1)$
    else
        $a \leftarrow x_1$
        $x_1 \leftarrow x_2$, $f_1 \leftarrow f_2$
        $x_2 = a + w \times (c - a)$
        Avaliar $f_2 = f(x_2)$
    end if
end while
return $(a + c)/2$ ▷ Ponto aproximado de mínimo

3 TRABALHOS RELACIONADOS

O primeiro trabalho relacionado propôs a integração do *Particle Swarm Optimization* (PSO) com uma versão modificada do método de Powell (BECK; GOMES, 2011). Nesse estudo, o PSO é responsável por fazer a busca global para encontrar a região onde estaria o ponto de mínimo global. De posse dessa região, um ponto dentro dela seria usado no método de Powell para convergir para o mínimo local, utilizando assim o melhor de cada algoritmo. O autor justifica que em determinado momento, as partículas ficam muito concentradas em torno do mínimo global e pouca melhoria ocorre em continuar executando o PSO. Assim, a transição para um método, que ele classifica como matemático, seria mais eficiente para realizar o refinamento do resultado. O método de Powell foi escolhido por não precisar do cálculo de derivadas. Sua estratégia de transição do PSO para o Powell basicamente calcula a distância das partículas para o centro de gravidade do população. Foi definida uma distância máxima a partir do centro de gravidade em que determinado percentual de partículas deve estar. Assim que esse percentual for atingido, ocorre a transição para o método e Powell. Foram feitas comparações entre este híbrido proposto contra versões do *simulated annealing* e do *ant colony* em funções de *benchmarck* entre 2 e 6 dimensões e no máximo 10 pontos de mínimos locais. Os resultados mostraram que para estes cenários, o algoritmo proposto PSO-Powell se saiu melhor em 9 das 10 comparações feitas.

O segundo trabalho que possui relação com nosso estudo realizou um experimento que envolveu a integração do método de Powell com o algoritmo genético e com o PSO (BENTO et al., 2013). Sua estratégia também era a de acionar o método de Powell ao final para um refinamento dos resultados. Diferente do trabalho anterior (BECK; GOMES, 2011), este deixou que ambos os algoritmos terminassem suas execuções, conforme seus próprios critérios de parada. A melhor solução encontrada tanto pelo AG quanto pelo PSO, seria entregue ao Powell para convergência até o mínimo. Os resultados encontrados mostraram que o algoritmo genético teve maior taxa de convergência com menos avaliações de função, em comparação com o PSO, ambos modificados para executar o método de Powell ao final.

Um último estudo relacionado propôs modificar o algoritmo genético para incorporar o método de Powell como um procedimento de busca local (GALSP) (TUNAY; ABIYEV, 2015). No texto é argumentado que os AGs possuem convergência muito lenta e por vezes é útil mesclá-los com outros métodos heurísticos com o objetivo de acelerar sua convergência. Ao contrário da ideia anterior (BENTO et al., 2013), esta aciona o método de Powell duas vezes em cada iteração do algoritmo genético. Assim, as etapas de seleção, *crossover* e mutação, se beneficiam de ter na população um indivíduo otimizado pelo método de Powell. O Algoritmo 8 mostra o pseudocódigo dessa adaptação.

Algoritmo 8 - Pseudocódigo do algoritmo genético com método de Powell (GALSP)

```
1: Início
2: population := 0
3: Inicializar população P_s
4: done := false
5: while not done do
6:     Calcular o fitness de cada indivíduo da população
7:     Aplicar o Método de Powell e salvar o melhor até o momento
8:     population := population + 1
9:     Selecionar indivíduos (população) a partir da população anterior
10:    Aplicar crossover C_p
11:    Aplicar mutação M_p
12:    Aplicar o Método de Powell com x^0
13:    done := Critério de Otimização foi atendido?
14: end while
15: Saída: melhor solução encontrada
16: Fim
```

O autor utilizou algumas funções de *benchmarck* para comparar o desempenho do AG com e sem o método de Powell. Os resultados mostraram a melhoria dos valores médio e *best* encontrados na versão com o método de Powell (GALSP).

4 PROPOSTA

4.1 ALGORITMO C-DEEPSO-POWELL

O algoritmo C-DEEPSO introduzido na seção 2.2.5 possui características do *Particle Swarm Optimization* (PSO) e do *Differential Evolution* (DE) que o conferem grande eficiência na busca global do espaço de busca. No entanto, no contexto de busca local, existe a necessidade de mesclarmos ele com outras heurísticas. Para isso, estamos propondo um novo híbrido C-DEEPSO-Powell, que incorpora o método de Powell apresentado na seção 2.3.

O método de Powell possui fortes características no sentido de realizar buscas locais, visto que com um ponto promissor, ele é capaz de se aproximar bastante do ótimo global. Sem um ponto promissor, a convergência é lenta e custosa. Dessa forma, a proposta do nosso híbrido é, em determinado ponto da execução, fornecer o *global best* para o método de Powell por algumas iterações. Ao final, devolvemos o *global best* otimizado para a população do C-DEEPSO continuar sua execução, se beneficiando agora de um indivíduo muito promissor para influenciar os demais. Essa é uma estratégia similar a utilizada no Algoritmo 8 proposto em (TUNAY; ABIYEV, 2015) em que o método de Powell ajuda a acelerar a convergência do algoritmo genético.

Diferentes estratégias podem ser utilizadas na transferência do C-DEEPSO para o método de Powell. Neste trabalho optamos por acionar o método de Powell no meio da execução do C-DEEPSO, em uma quantidade limitada de avaliações de função conforme o problema. Isso se dá pelo fato de que no meio da execução, é possível já estarmos *global best* em uma região muito promissora. O Algoritmo 9 mostra o pseudocódigo do C-DEEPSO-Powell.

Para testar a performance do novo híbrido proposto, vamos testá-lo com problemas de otimização em larga escala presentes na *suite* de testes do CEC'2013, que serão vistos no próximo capítulo.

Algoritmo 9 - Pseudocódigo do algoritmo C-DEEPSO-Powell

Início
Entrada: Iniciar os parâmetros de controle do C-DEEPSO - Tamanho da população N_P, Taxa de Mutação τ, Taxa de Comunicação P, Tamanho da Memória M_B, N_{gl}, dada uma taxa γ de ocorrência, Dimensão do problema (D) e numéro máximo de iterações k_{max}.
Passo 1: Iniciar o contador de gerações $t = 1$ e iniciar aleatoriamente a população de N_P partículas, conforme $P_t = \{X_{1,t}, \ldots, X_{N_P,t}\}$ com $X_{i,t} = [x_{i,t,1}, x_{i,t,2}, \ldots, x_{i,t,D}]$, em que cada partícula está distribuída uniformemente no intervalo $[X_{\min}, X_{\max}]$, no qual $X_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$ e $X_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$ com $i = 1, 2, \ldots, N_P$.
Passo 2: Avaliar a população corrente, P_t.
Passo 3: Atualizar a melhor solução obtida até o momento, X_{gb}.
while $k < k_{max}$ **do**
    for cada partícula i pertencente à população P_t **do**
        Calcular X_r usando a estratégia $S_{gPb\text{-rnd}}$;
        Mutar os parâmetros estratégicos w_T, w_A, w_C usando a Equação (2.8);
        Mutar X^*_{gb} usando a Equação (2.9);
        Aplicar regra de movimento à partícula corrente X_t via Equação (2.14);
        Avaliar a partícula corrente X_t;
        Selecionar a partícula com melhor aptidão para fazer parte da próxima população $(N_P + 1)$. **Usando por exemplo Torneio Estocástico*
    end for
    if $k == k_{max}/2$ **then**
        Atualiza X^*_{gb} com método de Powell conforme Algoritmo 6
    end if
    Atualizar o melhor indivíduo X_{gb} e a memória M_B;
    $k = k + 1$
end while
Fim

5 EXPERIMENTOS E RESULTADOS

5.1 FUNÇÕES DE BENCHMARK

Nesta seção, descrevemos as funções de benchmark básicas utilizadas no LSGO CEC'2013. Elas são a base das 15 funções utilizadas na competição (LI et al., 2013).

5.1.1 Função da Esfera

A função da esfera é uma função unimodal e totalmente separável, definida no intervalo $x_i \in [-100, 100]$. Uma função é dita totalmente separável se todos os seus subcomponentes dependem de apenas uma variável. Ela possui um mínimo global em $x = (0, \ldots, 0)$ com $f(x) = 0$. Sendo D a dimensão proposta, sua expressão é dada por:

$$f(x) = \sum_{i=1}^{D} x_i^2$$

5.1.2 Função Elíptica

A função elíptica utilizada na competição é uma função unimodal e não separável, caracterizada por um alto condicionamento, dado que é uma variação da função da esfera. Uma função é dita não separável quando não é possível decompô-la em subcomponentes independentes. Ela é definida no intervalo $x_i \in [-100, 100]$ e possui um mínimo global em $x = (0, \ldots, 0)$ com $f(x) = 0$. Sendo D a dimensão proposta, sua expressão é:

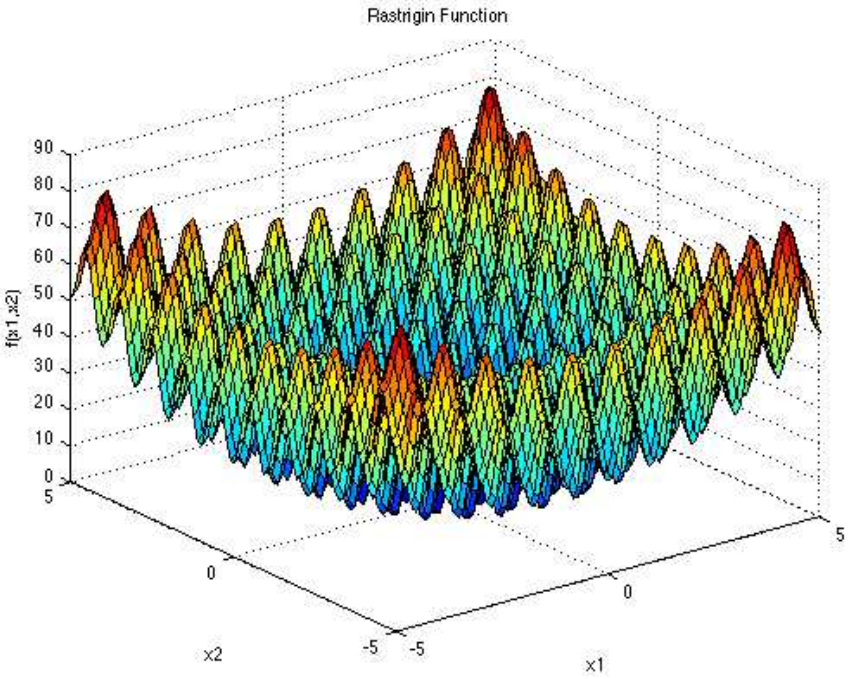
$$f(x) = \sum_{i=1}^{D} 10^{6^{\frac{i-1}{D-1}}} x_i^2$$

5.1.3 Função de Rastrigin

A função de Rastrigin é uma função multimodal e separável, definida no intervalo $x_i \in [-5.12, 5.12]$. Ela possui muitos mínimos locais e um mínimo global em $x = (0, \ldots, 0)$, onde $f(x) = 0$. Seu gráfico pode ser visualizado na Figura 4. Sendo D a dimensão proposta, sua expressão é:

$$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

Figura 4 – Gráfico da função de Rastrigin



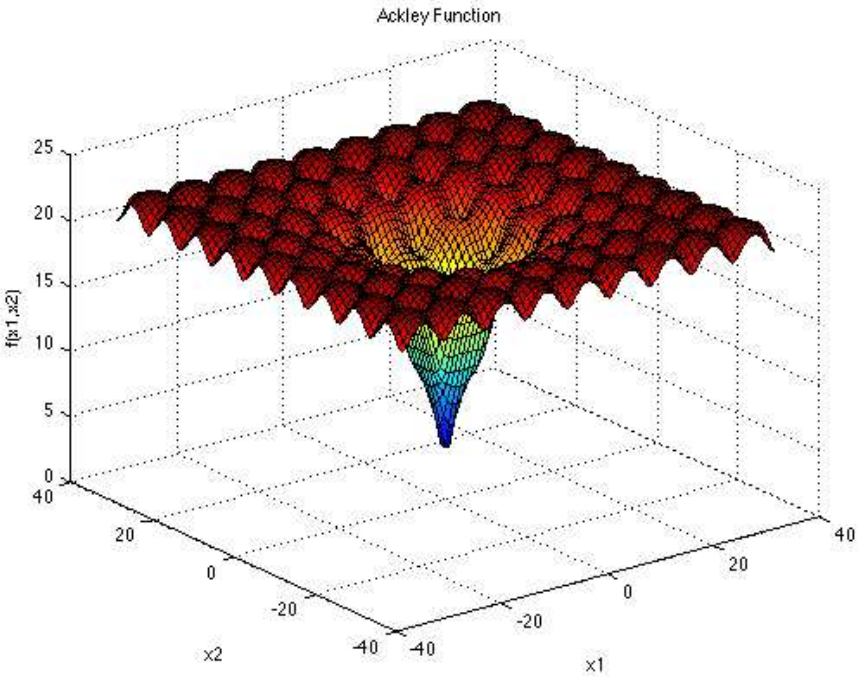
Fonte: Surjanovich, Sonja e Bingham, Derek. Disponível em: https://www.sfu.ca/~ssurjano/optimization.html. Acessado em: 16 nov. 2024.

5.1.4 Função de Ackley

A função de Ackley é uma função multimodal e não separável, definida no intervalo $x_i \in [-32.768, 32.768]$. Ela possui um mínimo global em $x = (0, \ldots, 0)$, com $f(x) = 0$. Seu gráfico pode ser visualizado na Figura 5. Sendo D a dimensão proposta, sua expressão é:

$$f(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$$

Figura 5 – Gráfico da função de Ackley



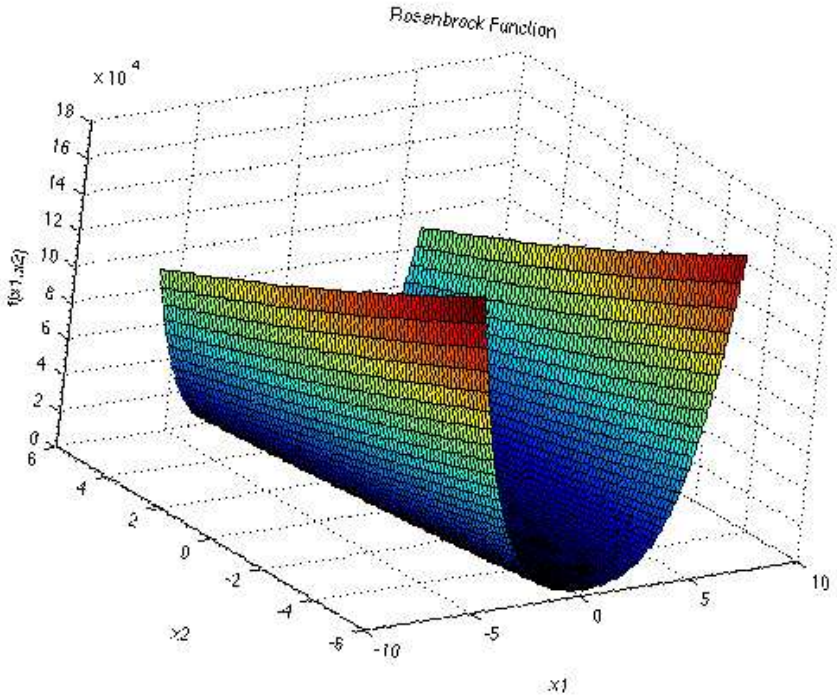
Fonte: Surjanovich, Sonja e Bingham, Derek. Disponível em: https://www.sfu.ca/~ssurjano/optimization.html. Acessado em: 16 nov. 2024.

5.1.5 Função de Rosenbrock

A função de Rosenbrock, também conhecida como função *Valley* ou *Banana*, é unimodal e não separável, geralmente avaliada no intervalo $x_i \in [-2.048, 2.048]$. Ela possui um vale estreito e curvado com um mínimo global em $x = (1, \ldots, 1)$, onde $f(x) = 0$. Seu gráfico pode ser visualizado na Figura 6. Sendo D a dimensão proposta, sua expressão é:

$$f(x) = \sum_{i=1}^{D-1} \left[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right]$$

Figura 6 – Gráfico da função de Rosenbrock



Fonte: Surjanovich, Sonja e Bingham, Derek. Disponível em: https://www.sfu.ca/~ssurjano/optimization.html. Acessado em: 16 nov. 2024.

5.1.6 Problema 1.2 de Schwefel

O problema 1.2 de Schwefel é uma função multimodal e não separável, definida no intervalo $x_i \in [-100, 100]$. Ela possui muitos mínimos locais, com um mínimo global em $x = (0, \ldots, 0)$, onde $f(x) = 0$. Sendo D a dimensão proposta, sua expressão é:

$$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i \right)^2$$

5.2 AJUSTE DE HIPERPARÂMETROS

Nosso experimento começa com a otimização dos hiperparâmetros de entrada do nosso C-DEEPSO. Na seção 2.2.4 foi feita uma crítica ao PSO pelo fato dos hiperparâmetros serem fixos e muito dependentes da função objetivo que está sendo otimizada. Hiperparâmetros de uma função objetivo, não necessariamente nos darão bons resultados em outra. O estudo proposto por Avancini (AVANCINI, 2022) mostrou a diferença entre usar hiperparâmetros otimizados no C-DEEPSO e não usar. Os resultados obtidos com a versão otimizada foram muito superiores. Por esse motivo, neste trabalho nós vamos otimizar os hiperparâmetros do C-DEEPSO com o objetivo de obter melhores execuções.

Optamos por utilizar um algoritmo genético (AG) nos moldes do discutido na seção 2.2.1. O AG foi executado com a seguinte configuração:

- **Seleção**: o operador de seleção utilizado foi o torneio com 3 indivíduos. Ou seja, são selecionados três indivíduos aleatórios e o de melhor *fitness* sobrevive.
- **Crossover**: o operador de *crossover* foi configurado com $\alpha = 0.5$.
- **Mutação**: o operador de mutação foi parametrizado com uma probabilidade $p = 0.2$ de ocorrer. Além disso, ao ocorrer, ela segue uma distribuição normal ao redor do gen do indivíduo com média 0.2.

Os hiperparâmetros otimizados do C-DEEPSO foram os pesos de inércia (w_T), cognição (w_A) e social (w_C), além das taxas de comunicação (P) e mutação (τ). Foram executadas otimizações para as funções de *benchmark* citadas acima em 100 dimensões, por 30 gerações e população fixada em 50 indivíduos. O critério de parada de cada execução do C-DEEPSO foi o limite de 100.000 avaliações de função. Os resultados com os hiperparâmetros encontrados estão na Tabela 1.

Tabela 1 – Hiperparâmetros otimizados por função de benchmark

Função	W_T	W_A	W_C	P	τ
Elíptica	0.437	0.769	0.809	0.901	0.803
Rastrigin	0.522	0.770	0.565	0.797	0.827
Ackley	0.513	0.670	0.205	0.696	0.368
Rosenbrock	0.402	0.379	0.754	0.582	0.304
Schwefel	0.503	0.455	0.093	0.582	0.339

5.3 PROBLEMAS DE OTIMIZAÇÃO EM LARGA ESCALA (LSGO)

Problemas que envolvem um grande número de variáveis de decisão são comumente chamados de *large-scale global optimization problems* (LSGOPs). Existem diversos cenários no mundo real em que este tipo de problema ocorre, como por exemplo no design de *power systems*, em redes de distribuição de água e problemas de roteamento de veículos em larga escala (LIU et al., 2024). Os muitos algoritmos de otimização que conhecemos são aplicados na resolução desse tipo de problema, no entanto, muitos deles caem no que chamamos de "maldição da dimensionalidade", que é quando a performance de um algoritmo começa a se deteriorar conforme as dimensões do problema vão crescendo.

A *IEEE CEC'2013 benchmark suite* foi desenvolvida com o objetivo de prover uma forma de testar e comparar algoritmos que se propoêm a resolver problemas de otimização em larga escala (LI et al., 2013). São 15 funções que partem das funções base evidenciadas

na seção 5.1. O objetivo dessas funções é simular os problemas LSGO do mundo real para ajudar pesquisadores a testarem e compararem a eficiência e resposta de seus algoritmos.

5.3.1 Funções de teste do IEEE CEC'2013

As 15 funções da *suite* de testes do CEC'2013 utilizam as funções base da seção 5.1 promovendo transformações nelas com o objetivo de aumentar sua irregularidade com distorções ou rotações por exemplo. O Quadro 2 mostra essas funções.

Quadro 2 – Funções de Benchmark do CEC'2013 para LSGO

Função	**Fórmula**	**Tipo**
Fully-separable Functions		
Shifted Elliptic Function	$f_1(z) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2$	Unimodal, Separável
Shifted Rastrigin's Function	$f_2(z) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right)$	Multimodal, Separável
Shifted Ackley's Function	$f_3(z) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + e$	Multimodal, Separável
Partially Additive Separable Functions I		
7-nonseparable, 1-separable Shifted and Rotated Elliptic Function	$f_4(z) = \sum_{i=1}^{\lvert S\rvert-1} w_i f_{\text{elliptic}}(z_i) + f_{\text{elliptic}}(z_{\lvert S\rvert})$	Unimodal, Parcialmente Separável
7-nonseparable, 1-separable Shifted and Rotated Rastrigin's Function	$f_5(z) = \sum_{i=1}^{\lvert S\rvert-1} w_i f_{\text{rastrigin}}(z_i) + f_{\text{rastrigin}}(z_{\lvert S\rvert})$	Multimodal, Parcialmente Separável
7-nonseparable, 1-separable Shifted and Rotated Ackley's Function	$f_6(z) = \sum_{i=1}^{\lvert S\rvert-1} w_i f_{\text{ackley}}(z_i) + f_{\text{ackley}}(z_{\lvert S\rvert})$	Multimodal, Parcialmente Separável
7-nonseparable, 1-separable Shifted Schwefel's Function	$f_7(z) = \sum_{i=1}^{\lvert S\rvert-1} w_i f_{\text{schwefel}}(z_i) + f_{\text{sphere}}(z_{\lvert S\rvert})$	Multimodal, Parcialmente Separável

Função	Fórmula	Tipo
Partially Additive Separable Functions II		
20-nonseparable Shifted and Rotated Elliptic Function	$f_8(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{elliptic}}(z_i)$	Unimodal, Parcialmente Separável
20-nonseparable Shifted and Rotated Rastrigin's Function	$f_9(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{rastrigin}}(z_i)$	Multimodal, Parcialmente Separável
20-nonseparable Shifted and Rotated Ackley's Function	$f_{10}(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{ackley}}(z_i)$	Multimodal, Parcialmente Separável
20-nonseparable Shifted Schwefel's Function	$f_{11}(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{schwefel}}(z_i)$	Unimodal, Parcialmente Separável
Overlapping Functions		
Shifted Rosenbrock's Function	$f_{12}(z) = \sum_{i=1}^{D-1} [100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2]$	Multimodal, Separável
Shifted Schwefel's Function with Conforming Overlapping Subcomponents	$f_{13}(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{schwefel}}(z_i)$	Unimodal, Não-separável, Sobreposição Cooperativa
Shifted Schwefel's Function with Conflicting Overlapping Subcomponents	$f_{14}(z) = \sum_{i=1}^{\lvert S\rvert} w_i f_{\text{schwefel}}(z_i)$	Unimodal, Não-separável, Sobreposição Conflitante
Fully Non-separable Functions		
Shifted Schwefel's Function	$f_{15}(z) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i\right)^2$	Unimodal, Totalmente Não-separável

Fonte: LI, X. et al. (2013, p. 9)

D é a dimensão do problema e $|S|$ são subconjuntos de D personalizados para determinada função. Todas as funções do Quadro 2 são deslocadas. A *suite* de testes possui vetores específicos para cada função. Sempre que nosso algoritmo possui um vetor ótimo candidato e quer realizar uma avaliação de função, é feita uma transformação em que nosso vetor é manipulado juntamente com o vetor presente na *suite* de testes. Por exemplo, na f_1 o vetor z que é utilizado para avaliar a função, possui a transformação conforme

equação 5.1.

$$\mathbf{z} = T_{\text{osz}}(\mathbf{x} - \mathbf{x}^{\text{opt}}) \tag{5.1}$$

A transformação da equação 5.1 promove certas irregularidades locais na função. Outras transformações também são aplicadas nas outras funções para criar cenários desafiadores para os algoritmos (LI et al., 2013).

Além do deslocamento, outras características são importantes para o entendimento das funções. Algumas são unimodais, ou seja, possuem apenas um ponto de mínimo que é o global. Outras, são multimodais, apresentando diversos pontos de mínimos locais, dificultando a convergência dos algoritmos. Outra características é se a função é separável ou não. Ser separável de modo geral significa que os componentes da função não dependem uns dos outros, assim a função pode ser otimizada em cada subcomponente, por exemplo. Uma função não-separável é mais complexa de se otimizar, dado que seus subcomponentes possuem certa dependência e precisam ser tratados em conjunto. Por fim, as funções $f_1 3$ e $f_1 4$ possuem características de sobreposição, sendo uma cooperativa, ou seja, os subcomponentes dessa função possuem variáveis que cooperam entre si, e portanto a sua otimização pode ser feita em conjunto. Já a sobreposição conflitante, os subcomponentes entram em conflito, uma vez que a melhora de um subcomponente pode piorar outro (LI et al., 2013).

5.3.2 Experimento nas Funções LSGO do CEC'2013

Os experimentos executados nesta seção tem por objetivo testar o desempenho do algoritmo proposto na seção 4.1. Para isso, foi executado o desafio de *benchmark* proposto no CEC'2013 para o algoritmo C-DEEPSO e para o algoritmo C-DEEPSO-Powell. Ambos os algoritmos executaram com os mesmos parâmetros e estratégias evolutivas, sendo sua única diferença, o uso do método de Powell ou não. Os 15 problemas de minimização foram executados em cada algoritmo por 25 vezes, sendo que cada execução teria como critério de finalização o número máximo de avaliações de função de 3.0×10^6, nos moldes do exigido pelo teste (LI et al., 2013).

Tabela 2 – Parâmetros de inicialização do C-DEEPSO

C-DEEPSO $P_b S_g -$ rnd	$f_1 - f_{12}, f_{15}$	f_{13}, f_{14}
Avaliações de Função	3.0×10^6	3.0×10^6
Dimensão	1000	905
Tamanho da População	500	500
Tamanho da Memória	50	50

A Tabela 2 mostra os parâmetros de inicialização do C-DEEPSO em ambos os algoritmos. Os hiperparâmetros utilizados em cada um dos 15 problemas respeitaram a função

base utilizada para selecionar os valores conforme a Tabela 1. Devido a limitações da *suite* de testes usada, as 15 funções não tiveram seus hiperparâmetros otimizados diretamente pois isso demandaria executar o Algoritmo Genético nas dimensões conforme a Tabela 2. Essa otimização seria muito custosa e demorada, então optamos por otimizar as funções base e utilizar seus hiperparâmetros.

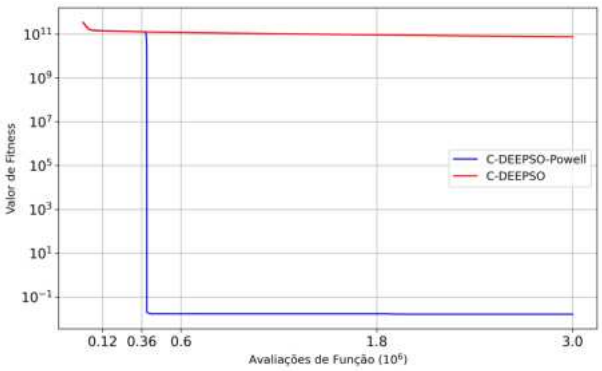
Para cada uma das funções, foram extraídos dados estatísticos em três grandes marcos da execução, sendo eles quando o teste atingisse 1.2×10^5 avaliações de função, 6.0×10^5 e finalmente 3.0×10^6. Cada execução retorna o melhor $fitness$ obtido na otimização. Das 25 execuções, foram calculadas a mediana, média, desvio padrão, o melhor resultado e o pior resultado de cada um dos três marcos. Por fim, curvas de convergência média de problemas selecionados para as 25 execuções foram feitas para ambos os algoritmos e estão presentes na Figura 7. Os testes foram realizados em uma máquina equipada com processador *multithread* AMD Ryzen 5 3600, 32 GB de memória RAM, executando Python 3.12.1 em um ambiente Windows 11. A Tabela 3 mostra os resultados das execuções feitas com o C-DEEPSO nos três marcos.

O C-DEEPSO-Powell foi adaptado para que o método de Powell fosse executado no meio dos três marcos. Além disso, foi distribuído um total de 3.0×10^5 avaliações de função de forma proporcional dentro de cada marco conforme a Tabela 4. Isso foi feito pois como os três marcos têm tamanhos muito diferentes, uma divisão por igual faria o método de Powell ser iniciado no primeiro marco e só terminasse no segundo. Portanto, quando o C-DEEPSO atingir 6.0×10^4 avaliações de função em seu contador, ele entregaria o *global best* para o método de Powell ser executado por 1.2×10^4 avaliações de função. Depois o *global best* seria devolvido a população e o C-DEEPSO continuaria sua execução até a próxima parada em 3.6×10^5, que seria o meio do segundo marco. Nesse momento, seriam executadas 6.0×10^4 avaliações de função no método de Powell para então devolver o controle para o C-DEEPSO. Por fim, ao atingir 1.8×10^6 avaliações de função, o método de Powell faria sua última melhoria no *global best* por 2.28×10^5 avaliações de função. Depois disso, o C-DEEPSO continuaria sendo executado até o limite de 3.0×10^6 avaliações de função. A Tabela 5 apresenta os resultados obtidos pelo C-DEEPSO-Powell nos três marcos.

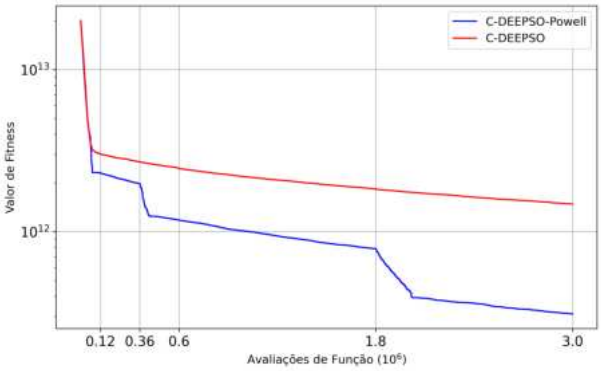
Tabela 4 – Parâmetros do método de Powell no experimento

Marcos	Qtd. Avaliações de Função	Início
1.2×10^5	1.2×10^4	6.0×10^4
6.0×10^5	6.0×10^4	3.6×10^5
3.0×10^6	2.28×10^5	1.8×10^6

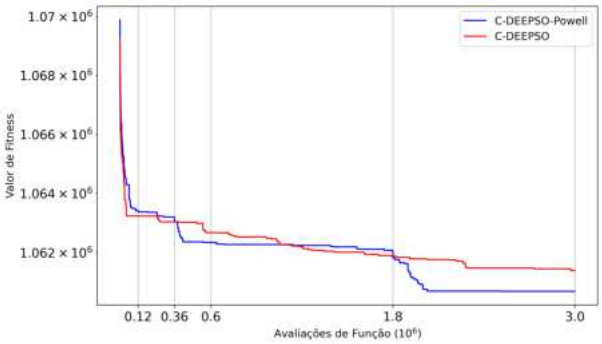
Figura 7 – Curvas de Convergência do C-DEEPSO e do C-DEEPSO-Powell



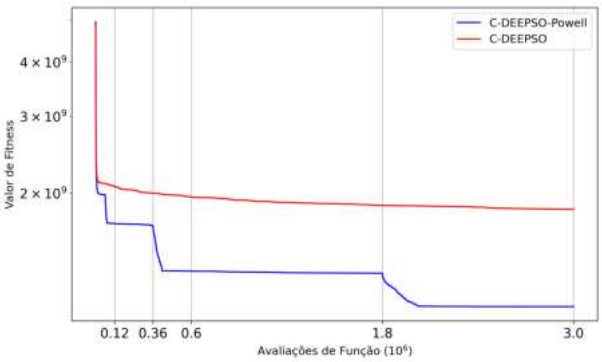
(a) Curvas de Convergência na f_1



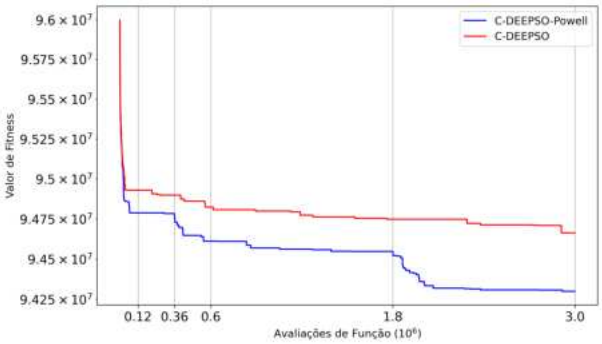
(b) Curvas de Convergência na f_4



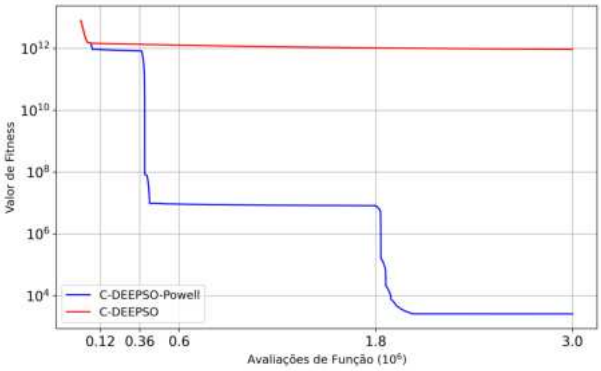
(c) Curvas de Convergência na f_6



(d) Curvas de Convergência na f_9



(e) Curvas de Convergência na f_{10}



(f) Curvas de Convergência na f_{12}

5.4 DISCUSSÃO DOS RESULTADOS

Ao compararmos os resultados do C-DEEPSO da Tabela 3 contra os resultados do C-DEEPSO-Powell da Tabela 5 podemos ver que houve pouco ou nenhum impacto da execução do Powell no primeiro marco. Isso pode ser explicado pelo fato do método de Powell precisar realizar buscas em linha em todas as dimensões para então chegar em um ponto promissor. Com poucas avaliações de função disponíveis no primeiro marco, ele não foi capaz de completar o percurso até esse ponto promissor, sendo interrompido para

devolução ao C-DEEPSO quase que de forma prematura.

Quando avaliamos as diferenças no segundo marco, é evidente a melhora em algumas funções como a f_1 e a f_{12} por exemplo. Dessa vez, o método de Powell teve avaliações de função suficientes para completar algumas iterações e retornar resultados melhores para o C-DEEPSO continuar sua execução.

No último marco os saltos de melhoria foram muito mais modestos que no segundo, embora a quantidade de avaliações de função disponíveis para o método de Powell tenha sido muito maior. A f_1 que antes havia dado um salto de 10^{11} para 10^{-2} em ordem de grandeza da média, se manteve em 10^{-2} no último marco. Já a f_{12} que antes tinha saltado de 10^{11} para 10^{6}, caiu ainda mais para 10^{3} no último marco.

Ao avaliarmos as curvas de convergência podemos ter uma ideia melhor do que ocorreu durante as execuções. Na Figura 7a temos as curvas de convergência da f_1 para ambos os algoritmos. É possível notar o enorme salto ocorrido ao meio do segundo marco, e quase que nenhuma melhoria no terceiro marco. Provavelmente o método de Powell convergiu para um mínimo e nem ele nem o C-DEEPSO foram capazes de retomar uma exploração global que os levasse a um ponto melhor. Diferente da função f_{12} em que podemos ver na Figura 7f em que é possível ver quedas nos três marcos. A execução do método de Powell foi capaz de melhorar de forma acentuada nos três momentos. Um desempenho parecido foi encontrado nas funções f_4, f_9, e f_{10}, como é possível ver nas figuras 7b, 7d e 7e. No entanto, a melhoria foi mais sutil.

Já para a função f_6 houve um empate, sugerindo que o método de Powell não tivesse feito nenhuma melhoria. No entanto, ao avaliar o gráfico de convergência média na Figura 7c podemos perceber que nos três marcos o método de Powell promoveu certa melhoria, mas de forma leve.

A primeira vista o C-DEEPSO-Powell foi superior ou empatou com o C-DEEPSO em todos os 15 problemas do CEC'2013, como é possível ver na Tabela 7. Nas funções f_3, f_5, f_6, f_8, f_9 e f_{10} houve um aparente empate, embora os valores do C-DEEPSO-Powell tenham sido melhores, mas não significativamente. Nas demais, a média e mediana são visivelmente melhores. Para confirmar o empate, realizamos um teste de hipótese da igualdade das médias utilizando *t-Student* com significância $\alpha = 0.05$ e os resultados estão na Tabela 6. Para esse teste, a hipótese nula (H0) foi aceita apenas na f_6, confirmando o empate. Nas demais, H0 foi rejeitada com o C-DEEPSO-Powell sendo a resposta.

Tabela 6 – Testes de Hipótese entre C-DEEPSO e C-DEEPSO-Powell

Função	p-valor	Ação	Resposta
f_3	0.00000	Rejeitar H0	C-DEEPSO-Powell
f_5	0.00000	Rejeitar H0	C-DEEPSO-Powell
f_6	0.29201	Aceitar H0	Ambos
f_8	0.03331	Rejeitar H0	C-DEEPSO-Powell
f_9	0.00000	Rejeitar H0	C-DEEPSO-Powell
f_{10}	0.00151	Rejeitar H0	C-DEEPSO-Powell

Resultados diferentes entre as diversas funções nos levam a buscar os motivos pelos quais em algumas delas como a f_1, f_{11}, f_{12}, f_{13}, f_{14} e f_{15}, os valores encontrados são muito melhores que nas demais. Uma explicação seria o fato de algumas dessas funções terem um formato mais quadrático que as demais. Powell (POWELL, 1964) provou que para funções quadráticas, seu algoritmo garante convergência para o mínimo global. O formato das funções quadráticas faz com que a minimização em linha nas direções calculadas durante o algoritmo, realize um passo maior para um vetor ótimo do que em funções mal-condicionadas, em que as direções promovem passos menores na tentativa de sair de zonas mais acidentadas. Assim, para funções mal-condicionadas, uma explicação é que o método de Powell precisa de muito mais iterações e por isso sua melhoria nos testes foi inferior.

Tabela 3 – Resultados C-DEEPSO

Funções		f_1	f_2	f_3	f_4	f_5
1.2×10^5	Melhor	1.25E+11	4.02E+04	2.16E+01	2.15E+12	1.72E+07
	Mediana	1.39E+11	4.26E+04	2.16E+01	3.14E+12	2.52E+07
	Pior	1.55E+11	4.62E+04	2.16E+01	4.47E+12	3.20E+07
	Média	**1.40E+11**	4.29E+04	2.16E+01	3.03E+12	2.53E+07
	DP	7.90E+09	1.67E+03	5.38E-03	5.75E+11	3.34E+06
6.0×10^5	Melhor	1.01E+11	3.98E+04	2.16E+01	1.79E+12	1.70E+07
	Mediana	1.19E+11	4.21E+04	2.16E+01	2.45E+12	2.41E+07
	Pior	1.34E+11	4.60E+04	2.16E+01	3.19E+12	3.18E+07
	Média	1.19E+11	4.23E+04	2.16E+01	2.46E+12	2.43E+07
	DP	8.73E+09	1.55E+03	6.30E-03	4.27E+11	3.20E+06
3.0×10^6	Melhor	6.34E+10	3.89E+04	2.16E+01	1.01E+12	1.65E+07
	Mediana	7.50E+10	4.11E+04	2.16E+01	1.57E+12	2.30E+07
	Pior	9.38E+10	4.56E+04	2.16E+01	1.98E+12	3.11E+07
	Média	7.52E+10	4.14E+04	2.16E+01	1.48E+12	2.29E+07
	DP	6.26E+09	1.46E+03	1.06E-02	3.18E+11	3.25E+06
Funções		f_6	f_7	f_8	f_9	f_{10}
1.2×10^5	Melhor	1.06E+06	2.77E+12	4.28E+16	1.66E+09	9.41E+07
	Mediana	1.06E+06	7.49E+12	1.51E+17	2.03E+09	9.50E+07
	Pior	1.07E+06	1.66E+13	2.16E+17	2.80E+09	9.54E+07
	Média	**1.06E+06**	8.40E+12	1.45E+17	2.08E+09	9.49E+07
	DP	2.61E+03	3.46E+12	3.85E+16	2.84E+08	3.28E+05
6.0×10^5	Melhor	1.06E+06	5.98E+11	3.14E+16	1.63E+09	9.41E+07
	Mediana	1.06E+06	1.35E+12	7.92E+16	1.95E+09	9.48E+07
	Pior	1.07E+06	4.41E+12	1.11E+17	2.43E+09	9.54E+07
	Média	**1.06E+06**	1.63E+12	7.53E+16	1.96E+09	9.48E+07
	DP	2.75E+03	8.18E+11	1.89E+16	2.07E+08	3.42E+05
3.0×10^6	Melhor	1.05E+06	1.74E+10	7.28E+15	1.48E+09	9.38E+07
	Mediana	1.06E+06	3.84E+10	2.10E+16	1.78E+09	9.47E+07
	Pior	1.07E+06	1.16E+11	3.97E+16	2.40E+09	9.54E+07
	Média	**1.06E+06**	5.04E+10	2.37E+16	1.83E+09	9.47E+07
	DP	2.81E+03	3.09E+10	8.87E+15	2.13E+08	3.72E+05
Funções		f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1.2×10^5	Melhor	3.45E+14	1.39E+12	2.75E+14	3.40E+14	3.32E+14
	Mediana	5.89E+14	1.45E+12	5.03E+14	9.05E+14	4.27E+14
	Pior	9.52E+14	1.50E+12	9.91E+14	1.92E+15	4.97E+14
	Média	5.95E+14	1.45E+12	5.51E+14	8.94E+14	4.26E+14
	DP	1.66E+14	3.25E+10	2.00E+14	3.70E+14	4.36E+13
6.0×10^5	Melhor	8.25E+13	1.22E+12	4.79E+13	8.27E+13	1.35E+14
	Mediana	1.23E+14	1.29E+12	1.01E+14	1.48E+14	1.86E+14
	Pior	2.48E+14	1.35E+12	2.24E+14	4.77E+14	2.74E+14
	Média	1.32E+14	1.28E+12	1.09E+14	1.71E+14	1.92E+14
	DP	3.69E+13	4.10E+10	4.69E+13	7.87E+13	3.54E+13
3.0×10^6	Melhor	5.66E+12	8.38E+11	1.43E+12	6.30E+12	3.07E+13
	Mediana	1.10E+13	9.33E+11	3.76E+12	9.31E+12	6.31E+13
	Pior	2.21E+13	1.07E+12	9.02E+12	1.61E+13	1.43E+14
	Média	1.20E+13	9.35E+11	4.14E+12	9.83E+12	6.89E+13
	DP	4.34E+12	5.93E+10	2.09E+12	2.30E+12	2.46E+13

Tabela 5 – Resultados C-DEEPSO-Powell

Funções		f_1	f_2	f_3	f_4	f_5
1.2×10^5	Melhor	1.32E+11	2.83E+04	2.11E+01	1.20E+12	1.72E+07
	Mediana	1.43E+11	3.15E+04	2.11E+01	2.27E+12	2.35E+07
	Pior	1.64E+11	3.47E+04	2.12E+01	3.15E+12	3.17E+07
	Média	1.43E+11	**3.17E+04**	**2.11E+01**	**2.30E+12**	**2.40E+07**
	DP	7.13E+09	1.74E+03	2.10E-02	4.54E+11	3.57E+06
6.0×10^5	Melhor	1.70E-02	2.46E+03	2.00E+01	5.97E+11	1.34E+07
	Mediana	1.70E-02	2.46E+03	2.00E+01	1.17E+12	1.88E+07
	Pior	1.70E-02	2.46E+03	2.00E+01	1.70E+12	2.42E+07
	Média	**1.70E-02**	**2.46E+03**	**2.00E+01**	**1.18E+12**	**1.87E+07**
	DP	1.00E-12	4.55E-13	4.48E-09	2.97E+11	2.74E+06
3.0×10^6	Melhor	1.55E-02	2.46E+03	2.00E+01	1.75E+11	1.23E+07
	Mediana	1.63E-02	2.46E+03	2.00E+01	3.23E+11	1.48E+07
	Pior	1.69E-02	2.46E+03	2.00E+01	5.19E+11	1.89E+07
	Média	**1.63E-02**	**2.46E+03**	**2.00E+01**	**3.11E+11**	**1.48E+07**
	DP	2.97E-04	4.55E-13	1.72E-06	9.44E+10	1.66E+06
Funções		f_6	f_7	f_8	f_9	f_{10}
1.2×10^5	Melhor	1.06E+06	5.52E+11	4.22E+16	1.35E+09	9.34E+07
	Mediana	1.06E+06	8.77E+11	1.42E+17	1.73E+09	9.49E+07
	Pior	1.07E+06	1.47E+12	3.01E+17	2.03E+09	9.54E+07
	Média	**1.06E+06**	**8.78E+11**	**1.37E+17**	**1.70E+09**	**9.48E+07**
	DP	1.99E+03	1.92E+11	6.38E+16	1.94E+08	4.17E+05
6.0×10^5	Melhor	1.06E+06	2.40E+09	2.26E+16	1.12E+09	9.34E+07
	Mediana	1.06E+06	5.03E+09	4.51E+16	1.29E+09	9.47E+07
	Pior	1.07E+06	1.29E+10	9.37E+16	1.69E+09	9.53E+07
	Média	**1.06E+06**	**5.32E+09**	**5.05E+16**	**1.32E+09**	**9.46E+07**
	DP	1.78E+03	2.13E+09	1.95E+16	1.52E+08	4.38E+05
3.0×10^6	Melhor	1.06E+06	6.35E+08	9.72E+15	8.77E+08	9.34E+07
	Mediana	1.06E+06	1.75E+09	1.50E+16	1.05E+09	9.44E+07
	Pior	1.06E+06	5.08E+09	5.22E+16	1.58E+09	9.48E+07
	Média	**1.06E+06**	**1.90E+09**	**1.80E+16**	**1.10E+09**	**9.43E+07**
	DP	1.57E+03	9.46E+08	9.11E+15	1.50E+08	3.76E+05
Funções		f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1.2×10^5	Melhor	1.39E+13	8.35E+11	3.91E+12	4.34E+12	3.05E+14
	Mediana	2.43E+13	9.15E+11	1.68E+13	1.16E+13	3.96E+14
	Pior	4.32E+13	9.64E+11	3.29E+13	7.33E+13	5.44E+14
	Média	**2.50E+13**	**9.18E+11**	**1.67E+13**	**1.49E+13**	**4.05E+14**
	DP	6.30E+12	3.21E+10	7.14E+12	1.36E+13	6.19E+13
6.0×10^5	Melhor	1.68E+11	7.52E+06	2.14E+10	2.21E+11	1.15E+10
	Mediana	3.84E+11	9.10E+06	3.04E+10	5.66E+11	6.59E+10
	Pior	1.30E+12	1.07E+07	4.31E+10	1.34E+12	1.94E+11
	Média	**4.31E+11**	**9.17E+06**	**3.00E+10**	**5.97E+11**	**7.67E+10**
	DP	2.20E+11	7.78E+05	4.82E+09	2.19E+11	5.48E+10
3.0×10^6	Melhor	7.64E+10	2.17E+03	1.05E+10	1.47E+11	2.33E+09
	Mediana	1.46E+11	2.57E+03	1.38E+10	3.17E+11	5.76E+09
	Pior	3.12E+11	3.07E+03	2.40E+10	4.63E+11	8.88E+09
	Média	**1.64E+11**	**2.57E+03**	**1.49E+10**	**3.09E+11**	**5.52E+09**
	DP	6.06E+10	2.33E+02	3.06E+09	8.27E+10	2.18E+09

Tabela 7 – Comparação entre C-DEEPSO e C-DEEPSO-Powell em 3×10^6 avaliações nas funções f_1-f_{15}

(a) Funções $f1$ a $f8$

Função	Métrica	C-DEEPSO	C-DEEPSO-Powell
$f1$	Melhor	6.34E+10	**1.55E-02**
	Mediana	7.50E+10	**1.63E-02**
	Pior	9.38E+10	**1.69E-02**
	Média	7.52E+10	**1.63E-02**
	DP	6.26E+09	**2.97E-04**
$f2$	Melhor	3.89E+04	**2.46E+03**
	Mediana	4.11E+04	**2.46E+03**
	Pior	4.56E+04	**2.46E+03**
	Média	4.14E+04	**2.46E+03**
	DP	1.46E+03	**4.55E-13**
$f3$	Melhor	2.16E+01	**2.00E+01**
	Mediana	2.16E+01	**2.00E+01**
	Pior	2.16E+01	**2.00E+01**
	Média	2.16E+01	**2.00E+01**
	DP	1.06E-02	**1.72E-06**
$f4$	Melhor	1.01E+12	**1.75E+11**
	Mediana	1.57E+12	**3.23E+11**
	Pior	1.98E+12	**5.19E+11**
	Média	1.48E+12	**3.11E+11**
	DP	3.18E+11	**9.44E+10**
$f5$	Melhor	1.65E+07	**1.23E+07**
	Mediana	2.30E+07	**1.48E+07**
	Pior	3.11E+07	**1.89E+07**
	Média	2.29E+07	**1.48E+07**
	DP	3.25E+06	**1.66E+06**
$f6$	Melhor	**1.05E+06**	1.06E+06
	Mediana	1.06E+06	1.06E+06
	Pior	1.07E+06	**1.06E+06**
	Média	1.06E+06	1.06E+06
	DP	2.81E+03	**1.57E+03**
$f7$	Melhor	1.74E+10	**6.35E+08**
	Mediana	3.84E+10	**1.75E+09**
	Pior	1.16E+11	**5.08E+09**
	Média	5.04E+10	**1.90E+09**
	DP	3.09E+10	**9.46E+08**
$f8$	Melhor	**7.28E+15**	9.72E+15
	Mediana	2.10E+16	**1.50E+16**
	Pior	**3.97E+16**	5.22E+16
	Média	2.37E+16	**1.80E+16**
	DP	**8.87E+15**	9.11E+15

(b) Funções $f9$ a $f15$

Função	Métrica	C-DEEPSO	C-DEEPSO-Powell
$f9$	Melhor	1.48E+09	**8.77E+08**
	Mediana	1.78E+09	**1.05E+09**
	Pior	2.40E+09	**1.58E+09**
	Média	1.83E+09	**1.10E+09**
	DP	2.13E+08	**1.50E+08**
$f10$	Melhor	9.38E+07	**9.34E+07**
	Mediana	9.47E+07	**9.44E+07**
	Pior	9.54E+07	**9.48E+07**
	Média	9.47E+07	**9.43E+07**
	DP	3.72E+05	**3.76E+05**
$f11$	Melhor	5.66E+12	**7.64E+10**
	Mediana	1.10E+13	**1.46E+11**
	Pior	2.21E+13	**3.12E+11**
	Média	1.20E+13	**1.64E+11**
	DP	4.34E+12	**6.06E+10**
$f12$	Melhor	8.38E+11	**2.17E+03**
	Mediana	9.33E+11	**2.57E+03**
	Pior	1.07E+12	**3.07E+03**
	Média	9.35E+11	**2.57E+03**
	DP	5.93E+10	**2.33E+02**
$f13$	Melhor	1.43E+12	**1.05E+10**
	Mediana	3.76E+12	**1.38E+10**
	Pior	9.02E+12	**2.40E+10**
	Média	4.14E+12	**1.49E+10**
	DP	2.09E+12	**3.06E+09**
$f14$	Melhor	6.30E+12	**1.47E+11**
	Mediana	9.31E+12	**3.17E+11**
	Pior	1.61E+13	**4.63E+11**
	Média	9.83E+12	**3.09E+11**
	DP	2.30E+12	**8.27E+10**
$f15$	Melhor	3.07E+13	**2.33E+09**
	Mediana	6.31E+13	**5.76E+09**
	Pior	1.43E+14	**8.88E+09**
	Média	6.89E+13	**5.52E+09**
	DP	2.46E+13	**2.18E+09**

6 CONCLUSÃO

Este trabalho experimentou a hibridização entre o algoritmo de otimização C-DEEPSO e o método de Powell. No contexto dos algoritmos de otimização, é comum a criação de novos híbridos com o objetivo de tentar utilizar as melhores características de cada um na exploração do espaço de busca. O método de Powell é muito eficiente como método de otimização sem a utilização de derivadas no contexto de exploração local, mas precisa de um bom ponto inicial para apresentar bons resultados. O C-DEEPSO possui características fortes do *Particle Swarm Optimization* (PSO) e do *Differential Evolution* (DE), o que lhe confere boa exploração global. A união dessas características foi explorada e testada neste trabalho.

Os experimentos na *suite* de testes do CEC'2013 demonstraram que a inclusão do método de Powell no C-DEEPSO pode melhorar significativamente os resultados em várias funções de *benchmark*. Em particular, o C-DEEPSO-Powell superou ou igualou o desempenho do C-DEEPSO em todos os 15 problemas testados, quando observado o resultado final dos algoritmos. Os testes de hipótese confirmaram que, em 14 das 15 funções, o C-DEEPSO-Powell apresentou resultados estatisticamente significantes.

Foi observado que as maiores melhorias ocorreram em funções com formato quadrático, sugerindo que esse híbrido deve ser priorizado nestes casos. Essa característica confirma a eficiência do método de Powell que garante convergência para um ótimo global para funções quadráticas. Foi possível perceber, através das curvas de convergência média, os saltos de melhoria, principalmente nas funções com forma mais próxima a quadrática. Em contrapartida, para as funções mal-condicionadas, essa melhoria foi muito mais tímida, sugerindo que o método de Powell foi pouco eficiente com aquela quantidade de avaliações de função.

A conclusão é de que a integração do método de Powell com o C-DEEPSO é promissora, mesmo nos testes com problemas de otimização em larga escala. A combinação da exploração global do C-DEEPSO com a exploração local do método de Powell trouxe maiores probabilidade de convergência para boa soluções, principalmente para problemas com funções objetivo de formato próximo ao quadrático.

Para trabalhos futuros, sugerimos a inclusão de heurísticas de busca local além do método de Powell, no C-DEEPSO. Uma busca local poderia ajudar o C-DEEPSO a sair de regiões de mínimos locais, para então utilizar o método de Powell. Além disso, uma outra abordagem seria utilizar pontos aleatórios no método de Powell, ao invés de sempre usar o *global best*. Isso poderia trazer um pouco mais de aleatoriedade ajudando o algoritmo a fugir de regiões de mínimo locais, dado que o *global best* pode acabar ficando retido nessas regiões por conta de execuções do método de Powell prematuras.

REFERÊNCIAS

ACTON, F. S. **Numerical methods that work**. Washington, D.C: Mathematical Association of America, 1990. ISBN 978-0-88385-450-1.

AVANCINI, J. V. d. C. **Criação de um framework para ajuste fino de parâmetros usando testes estatísticos**. Trabalho de Conclusão de Curso de Graduação — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2022. Orientadora: Carla Amor Divino Moreira Delgado; Coorientadora: Carolina Gil Marcelino. Disponível em: https://pantheon.ufrj.br/handle/11422/16281?locale=pt_BR.

BECK, A.; GOMES, W. Hybrid algorithms based on particle swarm optimization and the powell method for global optimization. In: **Proceedings of the Second International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering**. Civil-Comp Press, 2011. (CSC2011). ISSN 1759-3433. Disponível em: http://dx.doi.org/10.4203/ccp.97.42.

BENTO, D. et al. Genetic algorithm and particle swarm optimization combined with powell method. In: **AIP Conference Proceedings**. AIP, 2013. p. 578–581. ISSN 0094-243X. Disponível em: http://dx.doi.org/10.1063/1.4825557.

BEYER, H.-G.; SCHWEFEL, H.-P.; WEGENER, I. How to analyse evolutionary algorithms. **Theoretical Computer Science**, v. 287, n. 1, p. 101–130, set. 2002. ISSN 03043975. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0304397502001378.

BRENT, R. P. **Algorithms for minimization without derivatives**. Englewood Cliffs, N.J: Prentice-Hall, 1973. (Prentice-Hall series in automatic computation). ISBN 978-0-13-022335-7.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability: a guide to the theory of NP-completeness**. 27. print. ed. New York [u.a]: Freeman, 1979. (A series of books in the mathematical sciences). ISBN 978-0-7167-1045-5 978-0-7167-1044-8.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. 30. print. ed. Boston: Addison-Wesley, 1989. ISBN 978-0-201-15767-3.

GUIMARÃES, F. Capítulo 7. In: CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. (Ed.). **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra, 2012. ISBN 978-989-26-0583-8. Disponível em: https://ucdigitalis.uc.pt/pombalina/item/58522.

HOLLAND, J. H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. 1st mit press ed. ed. Cambridge, Mass: MIT Press, 1992. (Complex adaptive systems). ISBN 978-0-262-08213-6 978-0-262-58111-0.

HUANG, V.; QIN, A.; SUGANTHAN, P. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In: **2006 IEEE International Conference on Evolutionary Computation**. Vancouver, BC, Canada: IEEE, 2006.

p. 17–24. ISBN 978-0-7803-9487-2. Disponível em: http://ieeexplore.ieee.org/document/1688285/.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. Perth, WA, Australia: IEEE, 1995. v. 4, p. 1942–1948. ISBN 978-0-7803-2768-9. Disponível em: http://ieeexplore.ieee.org/document/488968/.

KIEFER, J. Sequential minimax search for a maximum. **Proceedings of the American Mathematical Society**, v. 4, n. 3, p. 502–506, jun. 1953. ISSN 0002-9939, 1088-6826. Disponível em: https://www.ams.org/proc/1953-004-03/S0002-9939-1953-0055639-3/.

LI, X. et al. **Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization**. [S.l.], 2013.

LIU, J. et al. Large-scale evolutionary optimization: A review and comparative study. **Swarm and Evolutionary Computation**, v. 85, p. 101466, mar. 2024. ISSN 22106502. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S2210650223002389.

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

MARCELINO, C. G. et al. Fundamentals of the C-DEEPSO algorithm and its application to the reactive power optimization of wind farms. In: **2016 IEEE Congress on Evolutionary Computation (CEC)**. Vancouver, BC, Canada: IEEE, 2016. p. 1547–1554. ISBN 978-1-5090-0623-6. Disponível em: http://ieeexplore.ieee.org/document/7743973/.

MIRANDA, V.; ALVES, R. Differential Evolutionary Particle Swarm Optimization (DEEPSO): A Successful Hybrid. In: **2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence**. Ipojuca, Brazil: IEEE, 2013. p. 368–374. ISBN 978-1-4799-3194-1. Disponível em: http://ieeexplore.ieee.org/document/6855877/.

MIRANDA, V.; FONSECA, N. EPSO - best-of-two-worlds meta-heuristic applied to power system problems. In: **Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)**. Honolulu, HI, USA: IEEE, 2002. v. 2, p. 1080–1085. ISBN 978-0-7803-7282-5. Disponível em: http://ieeexplore.ieee.org/document/1004393/.

PEREIRA, F. J. B. Capítulo 2. In: CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. (Ed.). **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra, 2012. ISBN 978-989-26-0583-8. Disponível em: https://ucdigitalis.uc.pt/pombalina/item/58522.

POWELL, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. **The Computer Journal**, v. 7, n. 2, p. 155–162, fev. 1964. ISSN 0010-4620, 1460-2067. Disponível em: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.2.155.

PRESS, W. H. (Ed.). **Numerical recipes: the art of scientific computing**. 3rd ed. ed. Cambridge, UK ; New York: Cambridge University Press, 2007. OCLC: ocn123285342. ISBN 978-0-521-88068-8 978-0-521-88407-5 978-0-521-70685-8.

STORN, R.; PRICE, K. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. **Journal of Global Optimization**, v. 23, p. 341–359, 01 1995.

TAKAHASHI, R. **Otimização Escalar e Vetorial**. Belo Horizonte: [s.n.], 2007. v. 2.

TAKAHASHI, R.; CUNHA, A. G. Capítulo 1. In: CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. (Ed.). **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra, 2012. ISBN 978-989-26-0583-8. Disponível em: https://ucdigitalis.uc.pt/pombalina/item/58522.

TUNAY, M.; ABIYEV, R. H. Hybrid local search based genetic algorithm and its practical application. **International Journal of Soft Computing and Engineering (IJSCE)**, v. 5, n. 2, maio 2015. ISSN 2231-2307. Disponível em: https://www.academia.edu/11793897/Hybrid_Local_Search_Based_Genetic_Algorithm_and_Its_Practical_Application.

VALLE, Y. D. et al. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. **IEEE Transactions on Evolutionary Computation**, v. 12, n. 2, p. 171–195, abr. 2008. ISSN 1941-0026, 1089-778X. Disponível em: http://ieeexplore.ieee.org/document/4358769/.