Luiza Engler Stadelhofer

Aplicação de Diferentes Métodos de Randomização em Meta-heurísticas

Joinville

2019

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Luiza Engler Stadelhofer

APLICAÇÃO DE DIFERENTES MÉTODOS DE RANDOMIZAÇÃO EM META-HEURÍSTICAS

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Rafael Stubs Parpinelli
Orientador

Joinville, Novembro de 2019

APLICAÇÃO DE DIFERENTES MÉTODOS DE RANDOMIZAÇÃO EM META-HEURÍSTICAS

Luiza Engler Stadelhofer

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação Integral do CCT/UDESC.

Banca Examinadora

Rafael Stubs Parpinelli - Doutor (orientador)

Chidambaram Chidambaram - Doutor

Gilmário Barbosa dos Santos - Doutor

Agradecimentos

Primeiramente, agradeço ao professor Rafael Stubs Parpinelli pela confiança, orientação e motivação ao longo do período de desenvolvimento deste trabalho. Assim também como todos os professores do departamento de Ciência da Computação da UDESC, que contribuíram para o meu crescimento através do conhecimento transmitido e das experiências e oportunidades proporcionadas.

Também agradeço a minha família, mais especificamente a minha mãe Roselena, pelo suporte e amor dado durante toda a minha vida. E por fim, agradeço aos meus amigos, Caroline, Mateus e Claudia, assim como meu companheiro Felipe, por estarem ao meu lado ao longo dessa jornada e por acreditarem em mim e na minha capacidade, quando nem eu mesma acreditei.

"If I have seen further it is by standing on the shoulders of giants." - Isaac Newton

Resumo

Meta-heurísticas são algoritmos estocásticos, portanto, a randomização é um dos seus princípios fundamentais. A randomização permite que o algoritmo não só seja capaz de escapar de mínimos ou máximos locais, fazendo uma busca global, como também ajuda em buscas locais na vizinhança da atual melhor solução. Levando isso em consideração, temos a disposição diversos métodos de randomização que podem ser utilizados nos algoritmos, como a distribuição Uniforme, Gaussiana, e de Cauchy, além dos mapas caóticos, como o Logístico e de Kent. Entretanto, grande parte dos estudos não fazem uma análise de qual é a melhor distribuição para se utilizar. Por isso, este trabalho foca na análise do impacto que diferentes distribuições probabilísticas possuem sobre algumas meta-heurísticas. As meta-heurísticas selecionadas para realização desta análise foram o Algoritmo Seno e Cosseno (SCA) e a Evolução Diferencial Auto-adaptativa (jDE), escolhidas devido a simplicidade de implementação. As distribuições utilizadas foram a Gaussiana e o mapa caótico Logístico, além da distribuição Uniforme, para comparação. Os algoritmos foram aplicados em dois tipos de problemas diferentes: inicialmente, em um conjunto selecionado de funções *benchmark* e depois em um problema do mundo real conhecido como Predição da Estrutura de Proteínas no modelo 2D-AB. Os resultados dos experimentos mostram que para as funções *benchmark* a distribuição Uniforme apresenta-se como a melhor opção de método de randomização para o SCA e o jDE. Entretanto, quando aplicados em um problema do mundo real, o SCA mostra melhores resultados com o mapa Logístico. Ademais, o SCA, quando adaptado com uma distribuição Gaussiana com desvio-padrão variável em seus principais parâmetros, obteve resultados superiores ao algoritmo original.

Palavras-chaves: Métodos de randomização, meta-heurísticas, mapas caóticos.

Abstract

Metaheuristics are stochastic algorithms, therefore, randomization is one of its fundamental principles. Randomization allows the algorithms not only to be able to escape from local minimums or maximums, performing a global search, but also helps in local searches in the neighborhood of the current best solution. Taking this into consideration, we have several randomization methods that can be used in these algorithms, such as the Uniform, Gaussian, and Cauchy distributions, as well as chaotic maps, such as Logistic and Kent. However, most studies do not analyze the best distribution that can be used. Therefore, this paper focuses on the analysis of the impact that different probability distributions have on some metaheuristics. The metaheuristics selected for this analysis were the Sine Cosine Algorithm (SCA) and the Self-Adaptive Differential Evolution (jDE), chosen due the simplicity of their implementation. The distributions used were the Gaussian distribution and the Logistic chaotic map, in addition to the Uniform distribution, used for comparison. The algorithms were applied to two different types of problems: initially, on a selected set of benchmark functions, and then on a real-world problem known as Protein Structure Prediction in the 2D-AB model. The results of the experiments show that for benchmark functions the Uniform distribution proves to be the best randomization method option for SCA and jDE. However, when applied to a real-world problem, SCA shows better results with the Logistic map. Moreover, SCA, when adapted with a Gaussian distribution with a variable standard deviation in its main parameters, obtains results superior to the original algorithm.

Keywords: Randomization methods, metaheuristics, chaotic maps.

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

DE Differential Evolution

jDE Self-adaptive Differential Evolution

SCA Sine Cosine Algorithm

GWO Grey Wolf Optimizer

JADE Adaptive Differential Evolution With Optional External Archive

SHADE Success-History based Adaptive DE

PSP Protein Structure Prediction

Conteúdo

1 Introdução

Na área de otimização, a classificação dos algoritmos pode ser dada através da sua natureza, dividindo os mesmos em duas categorias, algoritmos determinísticos e algoritmos estocásticos: em algoritmos determinísticos, a trajetória e valores de suas variáveis, assim como sua função objetivo são repetíveis; por exemplo, o algoritmo subida de encosta é determinístico, pois a partir de um mesmo ponto de partida ele seguirá o mesmo caminho sempre (YANG, 2010). Por outro lado, algoritmos estocásticos sempre terão algum tipo de aleatoriedade. Algoritmos genéticos são um bom exemplo, onde o vetor de soluções será diferente a cada rodada, visto que o algoritmo utiliza um gerador de números pseudo-aleatórios (YANG, 2010).

Algoritmos estocásticos possuem dois tipos, heurísticas e meta-heurísticas: explicando de forma vaga, heurística significa "descobrir através de tentativa e erro" - soluções boas para um problema complexo podem ser encontradas em um tempo razoável, mas não existe garantia que a solução ótima será encontrada (YANG, 2010). Mais adiante, foram desenvolvidas as meta-heurísticas, que significam "além" ou "alto-nível" e elas geralmente possuem um desempenho melhor do que as heurísticas (YANG, 2010).

Devido ao fato que meta-heurísticas são algoritmos estocásticos pode-se afirmar que a randomização é um dos princípios fundamentais para seus processos: a randomização permite que o algoritmo seja capaz de escapar de mínimos ou máximos locais, fazendo uma busca global, como também ajuda em buscas locais na vizinhança da atual melhor solução (YANG, 2013). Além disso, a randomização está fortemente ligada às propriedades de convergência do algoritmo (CAPONETTO et al., 2003), pois ajuda a descobrir novos pontos no espaço de busca, movendo as soluções em direção à diferentes regiões do espaço (FISTER et al., 2015).

Levando a importância da randomização para as meta-heurísticas em consideração, tem-se a disposição diversos métodos de randomização que podem ser utilizados nos algoritmos, como a distribuição Uniforme, Gaussiana, e de Cauchy, além dos mapas caóticos, como o Logístico e de Kent (FISTER et al., 2015). Entretanto, grande parte dos estudos não fazem uma análise de qual é a melhor distribuição para se usar, optando por

utilizar a distribuição Uniforme, por ser o gerador aleatório padrão de muitas linguagens.

Além disso, estudos como o de Resse (2009) mostram que o melhor gerador de números aleatórios para se utilizar depende não só do tipo de problema com o qual está se lidando, mas também do número de parâmetros, como o número de variáveis, a escolha da função, o espaço de busca e a acurácia desejada. Ou seja, utilizar o mesmo método de randomização para todos os tipos de problemas e em todas as partes do algoritmo que requerem randomização, pode não ser a melhor opção.

Com essa motivação, a contribuição deste trabalho se encontra na análise do impacto que diferentes distribuições probabilísticas possuem sobre algumas meta-heurísticas, como o Algoritmo Seno e Cosseno (SCA) (MIRJALILI, 2016) e a Evolução Diferencial Auto-adaptativa (jDE) (BREST; ZUMER; MAUCEC, 2006), que foram escolhidos baseados na simplicidade de implementação dos mesmos. A ideia desta análise é verificar a possibilidade de se utilizar diferentes distribuições em diferentes partes do algoritmo, para favorecer os processos de intensificação e diversificação do espaço de busca, assim como alguns algoritmos já fazem, como o JADE, proposto por Zhang e Sanderson (2009), e o SHADE, proposto por Tanabe e Fukunaga (2013).

1.1 Objetivos

Para guiar o processo de desenvolvimento do trabalho em questão, foram definidos um objetivo geral e objetivos específicos.

1.1.1 Objetivo Geral

Este trabalho tem como objetivo geral a aplicação de diferentes métodos de randomização em algumas meta-heurísticas e a análise do impacto dos mesmos nos resultados dos algoritmos.

1.1.2 Objetivos Específicos

- Implementar as meta-heurísticas selecionadas, jDE e SCA;
- Fazer as adaptações propostas, aplicando os diferentes métodos de randomização

nos algoritmos;

- Realizar experimentos aplicando os algoritmos propostos em um conjunto de funções *benchmark*;
- Analisar os resultados dos experimentos através de médias, desvios-padrão e gráficos de convergência e diversidade;
- Apresentar uma nova proposta de adaptação baseada nos resultados dos experimentos com *benchmarks*;
- Realizar a adaptação proposta no algoritmo SCA;
- Aplicar os algoritmos adaptados no problema de Predição da Estrutura de Proteínas;
- Analisar os resultados dos experimentos aplicados no problema do mundo real, também através de médias, desvios-padrão e gráficos de convergência e diversidade.

1.2 Metodologia

O desenvolvimento deste trabalho foi dividido em três partes principais: o estudo dos métodos de randomização, heurísticas, funções *benchmark* e problema de Predição da Estrutura de Proteínas (PSP); a implementação e adaptação dos algoritmos; e a análise dos resultados dos experimentos.

Inicialmente, foi realizado um levantamento de diferentes meta-heurísticas e métodos de randomização aplicados no trabalho em questão. Em seguida, foi feita uma revisão da literatura sobre os mesmos, buscando uma melhor compreensão dos tópicos. Além disso, foi realizado um estudo sobre funções *benchmark* e sobre o problema do mundo real abordado neste trabalho, o PSP.

Em sequência, os algoritmos foram implementados utilizando as linguagens C++ e Python, sendo adaptados através da aplicação de diferentes métodos de randomização em todas as partes dos algoritmos que necessitavam de aleatoriedade, e então foram aplicados às funções *benchmark* estudadas, assim como ao problema PSP, para obtenção dos resultados.

Depois, os resultados foram analisados e comparados a partir da geração de gráficos de convergência e diversidade, e medidas como médias e desvio-padrão. Ao fim

desse processo, chegou-se a conclusão da necessidade de uma nova proposta de adaptação, que foi então realizada.

Por fim, essa nova adaptação foi aplicada ao problema PSP e comparada com o algoritmo original, tendo seus resultados analisados a partir das mesmas medidas utilizadas anteriormente.

1.3 Estrutura

Este trabalho está dividido em 6 capítulos. O capítulo 2 apresenta a fundamentação teórica do trabalho, dando embasamento para os tópicos a serem abordados. O capítulo 3 apresenta os trabalhos relacionados, onde foi realizado um mapeamento sistemático da literatura. O capítulo 4 apresenta a proposta desenvolvida neste trabalho. O capítulo 5 apresenta os experimentos realizados, tanto utilizando as funções *bechmark* como o PSP e, por fim, o capítulo 6 apresenta as considerações finais deste estudo.

2 Fundamentação Teórica

Neste capítulo serão abordados conceitos e explicações relacionadas aos tópicos apresentados neste trabalho, para proporcionar uma melhor compreensão sobre os mesmos.

2.1 Distribuições Probabilísticas

Ocorre frequentemente que, ao realizar uma experiência, tem-se interesse principalmente em algumas funções do resultado, em oposição ao resultado em si - por exemplo, ao jogar dados, muitas vezes o interesse se encontra na soma dos dois dados e não no resultado real, individual de cada dado (ROSS, 2010). Ou seja, pode-se querer saber que a soma é sete e não se preocupar se o resultado real foi (1, 6) ou (2, 5) ou (3, 4) ou (4, 3) ou (5, 2) ou (6, 1). Essas quantidades de interesse, ou mais formalmente, essas funções de valor real definidas no espaço de amostra são conhecidas como variáveis aleatórias (ROSS, 2010).

Existem dois tipos de variáveis aleatórias, variáveis aleatórias discretas e variáveis aleatórias contínuas: variáveis discretas podem assumir apenas um número contável de valores, enquanto variáveis contínuas (as que serão utilizadas neste trabalho) possuem um conjunto incontável de possíveis valores (ROSS, 2010).

Sendo X uma variável aleatória; pode-se dizer que X é uma variável contínua se existir uma função não-negativa $f(x)$, definida para todos os números reais $x \in (-\infty, \infty)$, de forma que para qualquer conjunto B de números reais a seguinte propriedade é válida (ROSS, 2010):

$$P\{X \in B\} = \int_B f(x)dx \tag{2.1}$$

A função $f(x)$ é chamada de função densidade da variável aleatória X - em outras palavras, a equação 2.1 mostra que a probabilidade de X estar no conjunto B pode ser obtida integrando a função densidade sobre o conjunto B (ROSS, 2010).

Variáveis aleatórias são tão importantes em experimentos aleatórios que às

vezes o espaço amostral original do experimento é ignorado e foca-se na distribuição probabilística da variável aleatória (MONTGOMERY; RUNGER, 2003). A distribuição probabilística de uma variável aleatória X é uma descrição das probabilidades associadas a possíveis valores de X; e em alguns casos, é conveniente expressar a probabilidade em termos de uma fórmula (MONTGOMERY; RUNGER, 2003).

A seguir serão apresentadas algumas das distribuições probabilísticas mais conhecidas, para melhor compreensão.

2.1.1 Distribuição Uniforme

A distribuição Uniforme é um método de randomização que distribui uniformemente os valores em um intervalo formado por seus dois parâmetros: a, o limite inferior; e b, o limite superior - essa distribuição é caracterizada por possuir a mesma probabilidade para todos os sub-intervalos com mesmo tamanho, e sua função de densidade é a seguinte (FISTER et al., 2015):

$$P(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{caso contrário.} \end{cases} \tag{2.2}$$

Uma representação visual em 1D da distribuição Uniforme pode ser vista na figura 2.1, onde 2000 pontos foram randomicamente gerados utilizando a mesma. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

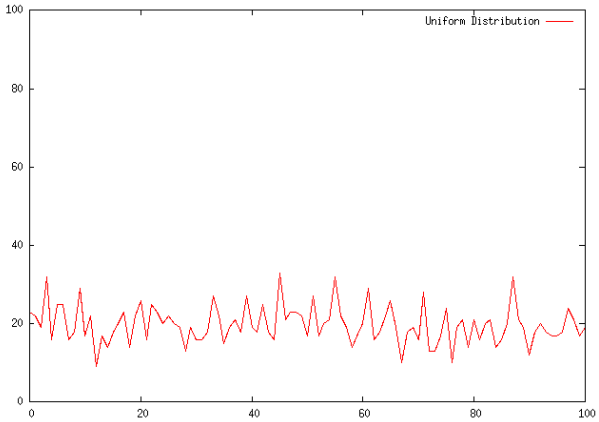

Figura 2.1: Representação Visual da Distribuição Uniforme

Fonte: Própria autora.

2.1.2 Distribuição Gaussiana

A distribuição Gaussiana ou Normal é definida por dois parâmetros: a média (μ) e o desvio-padrão (σ) (FISTER et al., 2015). Essa distribuição tem uma propriedade, onde aproximadamente dois terços dos valores caem em até um desvio-padrão da média, e a sua função de densidade é apresentada em 2.3 (FISTER et al., 2015):

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2.3}$$

Uma representação visual em 1D da distribuição Gaussiana pode ser vista na figura 2.2, onde 2000 pontos foram randomicamente gerados utilizando a mesma, com parâmetros $\mu = 50$ e $\sigma = 15$. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados com certo número; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

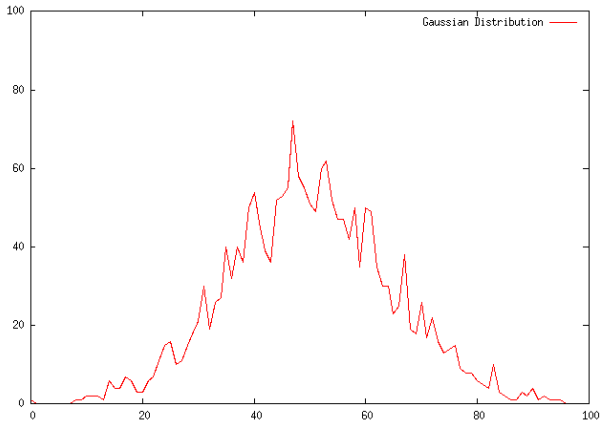

Figura 2.2: Representação Visual da Distribuição Gaussiana

Fonte: Própria autora.

2.1.3 Distribuição de Cauchy

A distribuição de Cauchy é também definida por dois parâmetros: α e β, parâmetros que afetam a média e a extensão da distribuição respectivamente, e sua função de densidade é como a seguir (VESTERSTROM; THOMSEN, 2004):

$$P(x) = \frac{1}{\beta\pi(1 + (\frac{x-\alpha}{\beta})^2)} \tag{2.4}$$

Uma representação visual em 1D da distribuição de Cauchy pode ser vista na figura 2.3, onde 2000 pontos foram randomicamente gerados utilizando a mesma, com

parâmetros $\alpha = 50$ e $\beta = 10$. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

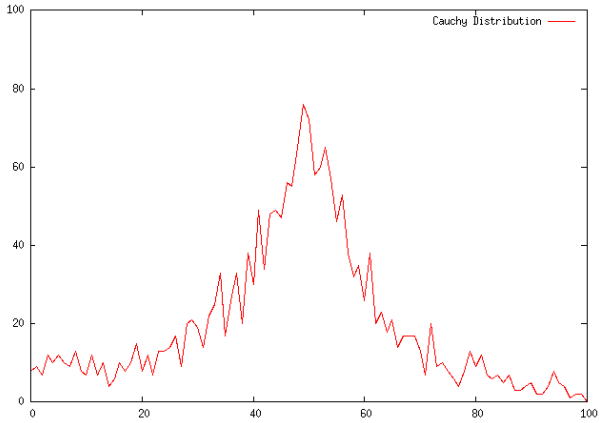

Figura 2.3: Representação Visual da Distribuição de Cauchy

Fonte: Própria autora.

2.1.4 Distribuição Exponencial

A distribuição Exponencial descreve o tempo entre eventos em um processo de Poisson e é definida por dois parâmetros: *x0*, que define o ponto inicial da distribuição e γ, que define a inclinação da curva da função de densidade da probabilidade, que é definida pela fórmula 2.5 (*YU; LAM; LI, 2012) [1].

$$P(x) = \begin{cases} \gamma e^{-\gamma(x-x0)}, & x \geq 0 \\ 0, & \text{x} < 0 \end{cases} \tag{2.5}$$

Uma representação visual em 1D da distribuição Exponencial pode ser vista na figura 2.4, onde 2000 pontos foram randomicamente gerados utilizando a mesma, com parâmetros $x0 = 0$ e $\gamma = 0,5$. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

[1] O asterisco encontrado nessa e em algumas outras referências representa os artigos resultantes do mapeamento sistemático da literatura, apresentado mais a frente no trabalho.

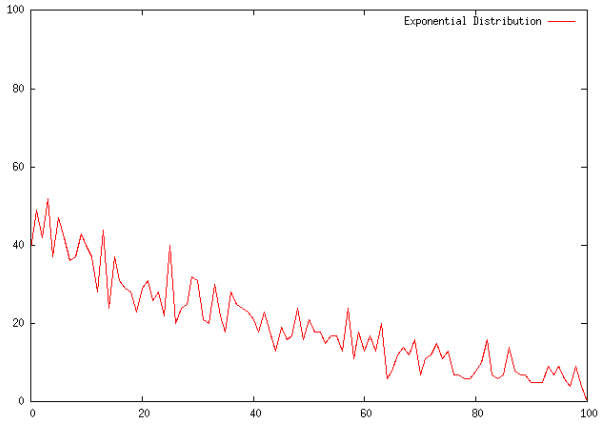

Figura 2.4: Representação Visual da Distribuição Exponencial

Fonte: Própria autora.

2.1.5 Distribuição Rayleigh

A distribuição de Rayleigh possui apenas um parâmetro: σ, que é responsável por controlar a planicidade da forma da função de densidade, que é definida pela equação 2.6 (*YU; LAM; LI, 2012).

$$P(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, \text{x} \geq 0 \tag{2.6}$$

Uma representação visual em 1D da distribuição Rayleigh pode ser vista na figura 2.5, onde 2000 pontos foram randomicamente gerados utilizando a mesma, com parâmetro $\sigma = 0,2$. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

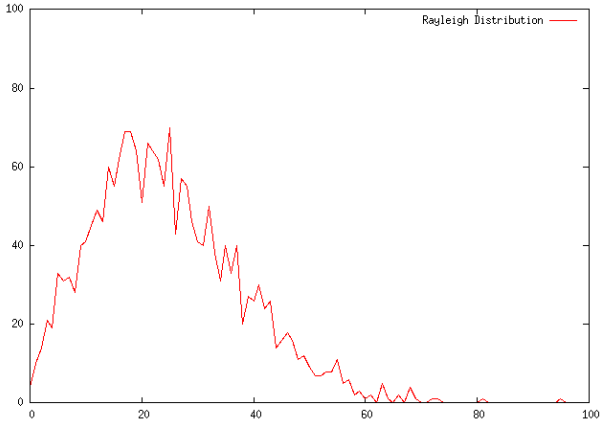

Figura 2.5: Representação Visual da Distribuição Rayleigh

Fonte: Própria autora.

2.1.6 Distribuição Beta

A distribuição Beta padrão possui dois parâmetros: a e b e sua função de densidade é dada pela equação 2.7, onde Γ define a função Gamma (ALI, 2007).

$$P(x) = \frac{\Gamma(a+b)}{\Gamma(a) + \Gamma(b)} x^{(a-1)}(1-x)^{(b-1)}, \text{ a, b} > 0 \tag{2.7}$$

Uma representação visual em 1D da distribuição Beta pode ser vista na figura 2.6, onde 2000 pontos foram randomicamente gerados utilizando a mesma, com parâmetros $a = 2$ e $b = 2$. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

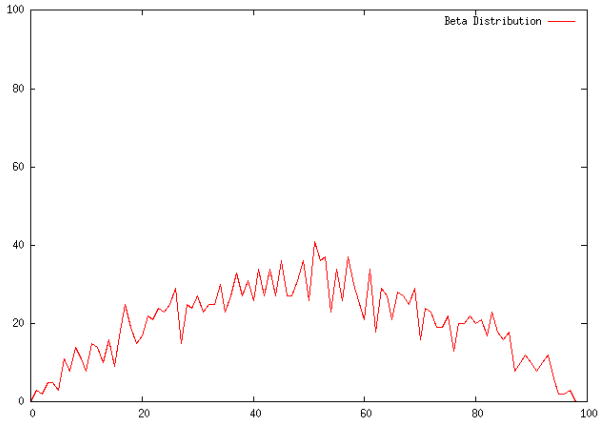

Figura 2.6: Representação Visual da Distribuição Beta

Fonte: Própria autora.

2.2 Mapas Caóticos

Um sistema dinâmico consiste em um conjunto de possíveis estados juntamente com uma regra que determina o estado atual em termos de estados passados - esta regra deve ser determinística, o que significa que pode-se determinar o estado atual unicamente a partir dos estados passados; além disso, nenhuma aleatoriedade é permitida nesta definição de sistemas dinâmicos determinísticos (ALLIGOOD; SAUER; YORKE, 1996).

Existem sistemas dinâmicos determinísticos nos quais a evolução temporal tem uma dependência forte nas condições iniciais, ou seja, as equações diferenciais que governam a evolução do sistema são muito sensíveis às condições iniciais (CATTANI et al.,

2017). Dessa maneira, medidas feitas em um estado do sistema em um certo momento podem não nos permitir prever a situação futura do sistema, apesar do fato de que as equações governantes são bem conhecidas (CATTANI et al., 2017), visto que esses sistemas se comportam de maneira imprevisível (FISTER et al., 2015). Por definição, essas equações são chamadas de caóticas.

Em alguns casos, é muito difícil estudar a evolução de um sistema integrando suas equações diferenciais, e às vezes também é difícil construir um modelo matemático exato para estudar o sistema - nesses casos, é possível conseguir uma boa descrição de um processo caótico usando um modelo iterativo algébrico chamado mapeamento (CATTANI et al., 2017).

Existem inúmeros sistemas caóticos que são estudados utilizando a abordagem de mapeamento (CATTANI et al., 2017). Algumas dessas abordagens são apresentadas a seguir.

2.2.1 Mapa Logístico

O mapa Logístico é um mapa caótico. É determinado pela equação 2.8 onde $x_n \in [0, 1]$ e r é um parâmetro que quando igual a 4 apresenta comportamento caótico (FISTER et al., 2015).

$$x_{n+1} = r x_n (1 - x_n) \tag{2.8}$$

Uma representação visual em 1D do mapa Logístico pode ser vista na figura 2.7, onde 2000 pontos foram randomicamente gerados utilizando a mesmo. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

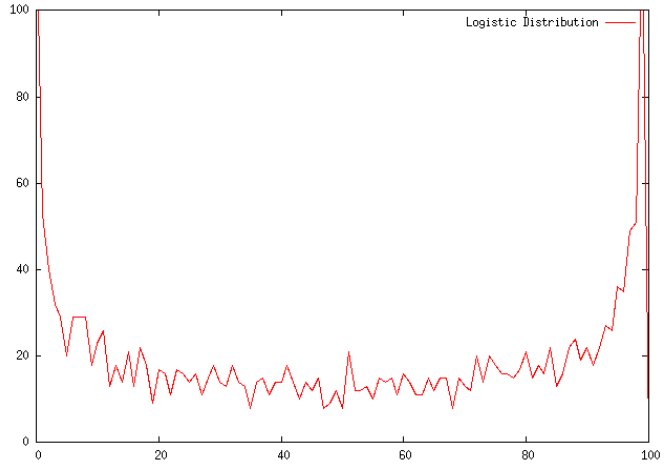

Figura 2.7: Representação Visual do Mapa Logístico

Fonte: Própria autora.

2.2.2 Mapa Sinusoidal

Esse mapa caótico pode ser definido pela fórmula 2.9, onde quando a = 2,3 ele exibe comportamento caótico (*GANDOMI; YANG, 2014).

$$x_{n+1} = ax_n^2 sin(\pi x_n) \tag{2.9}$$

Uma representação visual em 1D do mapa Sinusoidal pode ser vista na figura 2.8, onde 2000 pontos foram randomicamente gerados utilizando a mesmo. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

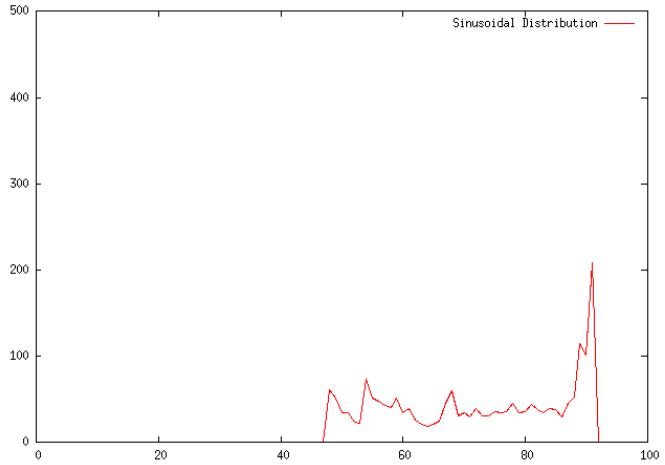

Figura 2.8: Representação Visual do Mapa Sinusoidal

Fonte: Própria autora.

2.2.3 Mapa Piecewise

O mapa caótico Piecewisse, é definido pela fórmula 2.10, onde P é o parâmetro de controle, normalmente escolhido entre 0,0 e 0,5 (*GANDOMI; YANG, 2014).

$$x_{n+1} = \begin{cases} \frac{x_n}{P}, & 0 \leq x_n < P \\ \frac{x_n - P}{0{,}5 - P}, & P \leq x_n < \frac{1}{2} \\ \frac{1 - P - x_n}{0{,}5 - P}, & \frac{1}{2} \leq x_n < 1 - P \\ \frac{1 - x_n}{P}, & 1 - P \leq x_n < 1 \end{cases} \tag{2.10}$$

Uma representação visual em 1D do mapa Piecewise pode ser vista na figura 2.9, onde 2000 pontos foram randomicamente gerados utilizando a mesmo. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

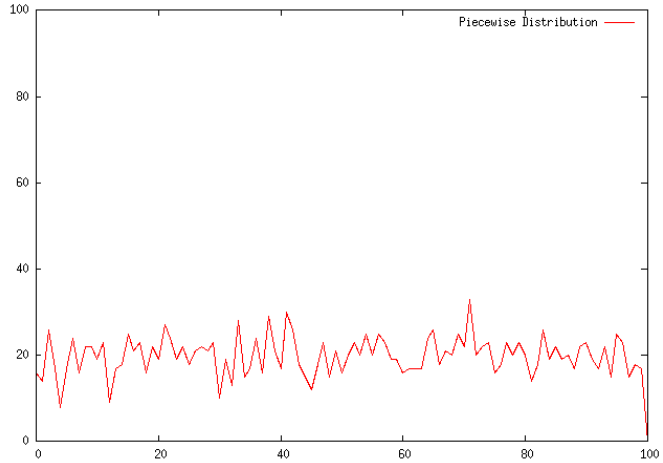

Figura 2.9: Representação Visual do Mapa Piecewise

Fonte: Própria autora.

2.2.4 Mapa Tent

O mapa caótico Tent é similar ao mapa Logístico e é definido pela equação 2.11 (*GANDOMI; YANG, 2014).

$$x_{n+1} = \begin{cases} \frac{x_n}{0{,}7}, & x_n < 0,7 \\ \frac{10}{3}(1 - x_n) & x_n \geq 0,7 \end{cases} \tag{2.11}$$

Uma representação visual em 1D do mapa Tent pode ser vista na figura 2.10,

onde 2000 pontos foram randomicamente gerados utilizando a mesmo. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

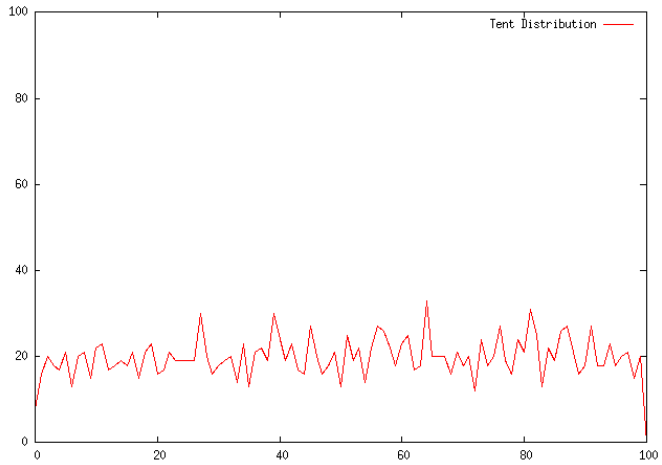

Figura 2.10: Representação Visual do Mapa Tent

Fonte: Própria autora.

2.2.5 Mapa Circle

O mapa Circle é representado pela equação 2.12, e quando seus parâmetros a e b forem igualados a 0,5 e 0,2, respectivamente, ele apresenta um comportamento caótico (*GANDOMI; YANG, 2014).

$$x_{n+1} = x_n + b - (\frac{a}{2\pi})sin(2\pi x_n)mod(1) \tag{2.12}$$

Uma representação visual em 1D do mapa Circle pode ser vista na figura 2.11, onde 2000 pontos foram randomicamente gerados utilizando a mesmo. O gráfico é composto pelo eixo x, onde tem-se a quantidade de pontos gerados; e pelo eixo y, onde tem-se os respectivos números que foram gerados, em um intervalo de $[0, 100]$.

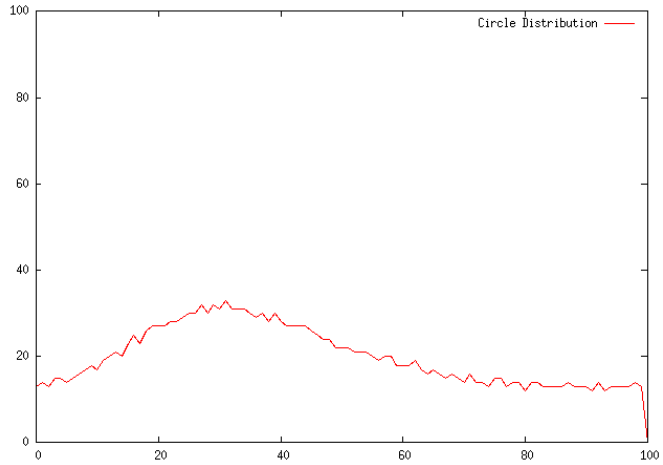

Figura 2.11: Representação Visual do Mapa Circle

Fonte: Própria autora.

2.3 Meta-heurísticas

Como dito anteriormente, as meta-heurísticas são métodos que significam "além" ou "alto-nível" e possuem, geralmente, um desempenho melhor do que as heurísticas (YANG, 2010). Além disso, todos os algoritmos meta-heurísticos usam uma combinação entre aleatoriedade/randomização e busca local (YANG, 2010). As meta-heurísticas podem ser consideradas como uma maneira eficiente de produzir soluções aceitáveis através da tentativa e erro, para problemas complexos, em um tempo razoavelmente prático (YANG, 2013).

Em geral, o desempenho de meta-heurísticas depende de dois componentes chaves, intensificação e diversificação: intensificação significa focar a busca em uma região local explorando a informação de que uma solução ótima atual se encontra nessa região (SAHIB; ABDULNABI; MOHAMMED, 2018). A intensificação usa qualquer informação do problema de interesse para gerar novas soluções que são melhores do que as soluções encontradas anteriormente; a vantagem disso é que usualmente ela leva a níveis de convergência altos, e a desvantagem é que ela pode ficar facilmente presa em ótimos locais (SAHIB; ABDULNABI; MOHAMMED, 2018).

Já a diversificação ajuda a gerar soluções diversas com o objetivo de explorar o espaço de busca em uma escala global, permitindo assim explorar esse espaço de uma forma mais eficiente, e gerar soluções distantes das soluções atuais; a vantagem disso é que ela é menos provável de ficar presa em ótimos locais, e a otimalidade global pode se tornar mais alcançável, entretanto, sua desvantagem é que pode levar à uma velocidade

de convergência muito lenta (SAHIB; ABDULNABI; MOHAMMED, 2018).

O balanço entre esses dois componentes é extremamente importante para a eficiência e desempenho do algoritmo - muita intensificação e pouca diversificação pode causar o algoritmo a ficar preso em ótimos locais, o que pode dificultar ou até prevenir o algoritmo de encontrar o ótimo global (YANG, 2013). Por outro lado, muita diversificação e pouca intensificação pode dificultar o objetivo do algoritmo de convergir, diminuindo seu desempenho (YANG, 2013).

A seguir serão apresentadas e explicadas as meta-heurísticas que serão aplicadas neste trabalho.

2.3.1 Evolução Diferencial

A Evolução Diferencial (DE) é um algoritmo evolucionário para otimização global sobre espaços contínuos (BREST; ZUMER; MAUCEC, 2006). De forma geral, o algoritmo cria novos candidatos à solução combinando o indivíduo parente e diversos outros indivíduos da mesma população (BREST; ZUMER; MAUCEC, 2006). O candidato apenas substitui o parente se possuir um valor *fitness* melhor (BREST; ZUMER; MAUCEC, 2006). A Evolução Diferencial possui três parâmetros:

- F: fator de amplificação do vetor de diferença (escala de mutação);
- CR: parâmetro de controle de *crossover*;
- NP: tamanho da população.

A população inicial é selecionada de forma randômica entre os limites inferior e superior (BREST; ZUMER; MAUCEC, 2006). Então a DE realiza um processo chamado evolução: para cada geração a DE emprega as operações de mutação e *crossover* para produzir um vetor *trial* (BREST; ZUMER; MAUCEC, 2006). Logo após, a operação de seleção é utilizada para escolher os vetores para a próxima geração (BREST; ZUMER; MAUCEC, 2006).

2.3.1.1 Mutação

A mutação cria um vetor mutante para cada vetor da população. Eles podem ser criados usando uma estratégia de mutação, como por exemplo: rand/1; best/1; current to best/1;

best/2; rand/2; entre outras (BREST; ZUMER; MAUCEC, 2006). A estratégia de mutação utilizada neste trabalho é a rand/1, que é dada pela seguinte fórmula (BREST; ZUMER; MAUCEC, 2006):

$$v_{i,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G}) \tag{2.13}$$

Onde os índices r1, r2 e r3 representam valores inteiros randômicos, divergentes uns dos outros e do índice i, gerados no intervalo $[1, NP]$. (BREST; ZUMER; MAUCEC, 2006).

2.3.1.2 Crossover

Depois da mutação, uma operação de *crossover* binária é realizada para formar o vetor *trial* final, de acordo com o vetor i da população e seu vetor mutante correspondente (BREST; ZUMER; MAUCEC, 2006). A fórmula utilizada é a seguinte (BREST; ZUMER; MAUCEC, 2006):

$$u_{i,j\ ,G} = \begin{cases} V_{i,j,G} & \text{se } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{caso contrário.} \end{cases} \tag{2.14}$$

Onde x é o vetor da população; v é o vetor mutante; i é o índice no intervalo $[1, NP]$; j é o índice no intervalo $[1, D]$ sendo D a quantidade de dimensões do problema; e j_{rand} é um índice escolhido de forma aleatória dentro do intervalo $[1, NP]$.

2.3.1.3 Seleção

A seleção seleciona, de acordo com o valor *fitness* do vetor da população e do seu vetor *trial* correspondente, qual vetor irá sobreviver e se tornar um membro da próxima geração (BREST; ZUMER; MAUCEC, 2006).

2.3.1.4 Evolução Diferencial Auto-Adaptativa

A Evolução Diferencial Auto-Adaptativa (jDE) usa um mecanismo auto-adaptativo para controlar os parâmetros F e CR (BREST; ZUMER; MAUCEC, 2006). Cada indivíduo da população é estendido para possuir seus próprios valores de F e CR (BREST; ZUMER; MAUCEC, 2006). Melhores valores para esses parâmetros de controle levam a melhores

indivíduos, que consequentemente, possuem mais chances de sobreviver e produzir descendentes, propagando esses melhores valores de parâmetros (BREST; ZUMER; MAUCEC, 2006).

Então, os valores para os parâmetros de controle são calculados como a seguir (BREST; ZUMER; MAUCEC, 2006):

$$F_{i,G+1} = \begin{cases} Fl + rand1 * Fu & \text{se } rand2 < T1 \\ F_{i,G} & \text{caso contrário.} \end{cases} \tag{2.15}$$

$$CR_{i,G+1} = \begin{cases} rand3 & \text{se } rand4 < T2 \\ CR_{i,G} & \text{caso contrário.} \end{cases} \tag{2.16}$$

Onde $randj, j \in \{1, 2, 3, 4\}$ são valores uniformemente distribuídos em um intervalo de $[0, 1]$; e F1, Fu, T1 e T2 são valores fixos com os respectivos números: 0,1, 0,9, 0,1 e 0,1 (BREST; ZUMER; MAUCEC, 2006).

Como pode-se observar no diagrama da figura 2.12, tem-se que o algoritmo utiliza a randomização em cinco partes do seu processo e estão anotados em caixas cinzas no diagrama: ao inicializar os agentes no espaço de busca; ao calcular os novos parâmetros F e CR; ao realizar a operação de mutação e por fim, ao realizar a operação de *crossover*. O jDE utiliza, por padrão, a distribuição uniforme para realizar todas essas randomizações.

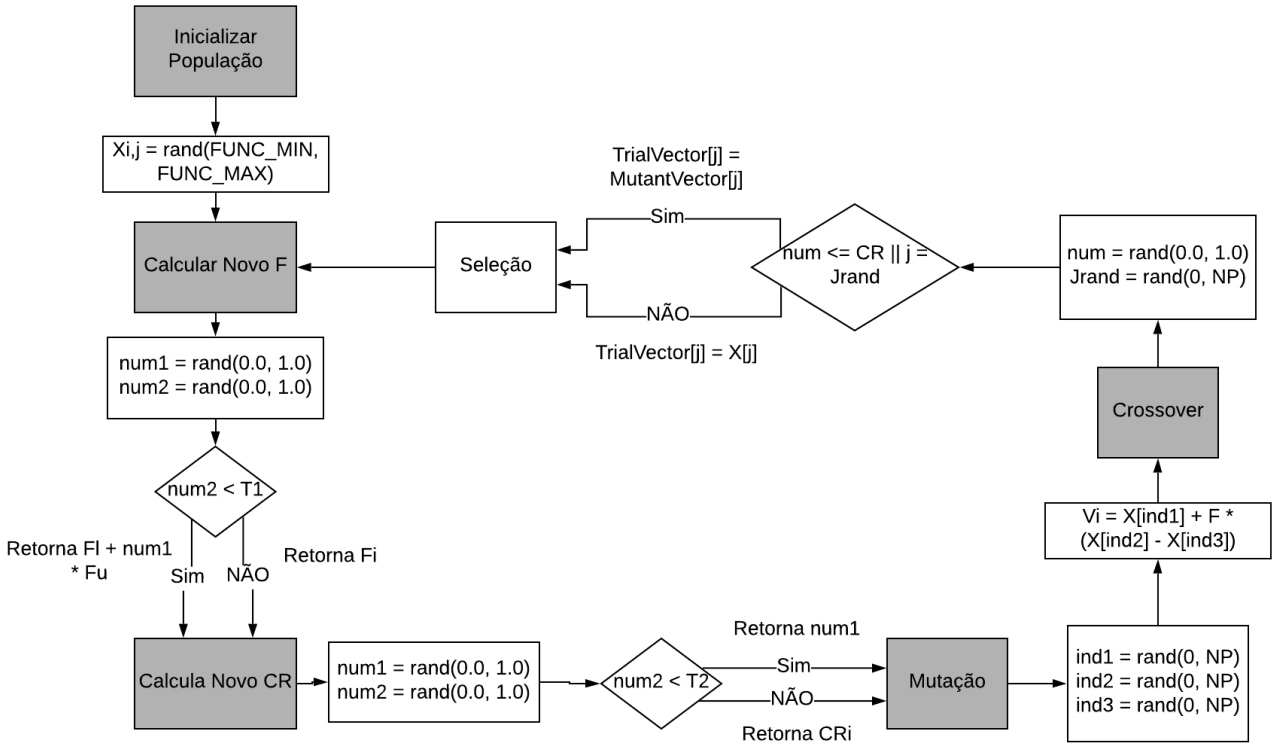

Figura 2.12: Diagrama de Blocos do jDE

Fonte: Própria autora.

2.3.2 Algoritmo Seno e Cosseno

O SCA (*Sine Cosine Algorithm*) cria múltiplas soluções randômicas iniciais e requer que as mesmas flutuem em direção ou na direção oposta da melhor solução usando um modelo matemático baseado nas funções seno e cosseno (MIRJALILI, 2016).

A seguinte equação de *update* é utilizada para realizar as fases de intensificação e diversificação do algoritmo (MIRJALILI, 2016):

$$x_i^{t+1} = \begin{cases} x_i^t + r1 * seno(r2) * |r3 * p_i^t - x_i^t| & \text{se } r4 < 0,5 \\ x_i^t + r1 * cosseno(r2) * |r3 * p_i^t - x_i^t| & \text{se } r4 \geq 0,5 \end{cases} \quad (2.17)$$

Onde p_i^t é o ponto de destino, ou seja, a melhor solução até então encontrada pelo algoritmo; e r1, r2, r3 e r4 são os quatro parâmetros do SCA (MIRJALILI, 2016):

- r1 - parâmetro que faz o balanço entre intensificação e diversificação, mudando de forma auto-adaptativa o intervalo das funções seno e cosseno a partir da seguinte

equação:

$$r1 = a - t * \frac{a}{T} \tag{2.18}$$

Onde t é a iteração atual; T é a quantidade máxima de iterações; e a é uma constante com o valor de 2;

- r2 - valor aleatório dentro do intervalo $[0, 2\pi]$ que define o quão grande deve ser o movimento em direção ou direção oposta ao ponto de destino;
- r3 - valor aleatório no intervalo $[0, 2]$ que estocasticamente enfatiza ($r3 > 1$) ou não ($r3 < 1$) o efeito do ponto de destino para calcular a distância do movimento;
- r4 - valor aleatório no intervalo $[0, 1]$ que faz a mudança entre seno e cosseno.

O SCA começa com um conjunto de soluções aleatórias, e então salva a melhor solução encontrada até o momento, atribui a mesma ao ponto de destino e atualiza todas as outras soluções com respeito a ela (MIRJALILI, 2016). Ao mesmo tempo, os intervalos das funções seno e cosseno são atualizados enquanto o contador de iterações aumenta (MIRJALILI, 2016). O algoritmo termina quando o número máximo de iterações é alcançado ou quando uma certa acurácia do resultado é alcançada (MIRJALILI, 2016).

Como pode-se observar no diagrama de blocos da figura 2.13, existem dois pontos no algoritmo que utilizam a randomização. O primeiro é ao iniciar os indivíduos da população no espaço de busca; e o segundo é ao atualizar os parâmetros r2, r3 e r4 do SCA. Para realizar essa randomização, o algoritmo por padrão utiliza a distribuição uniforme.

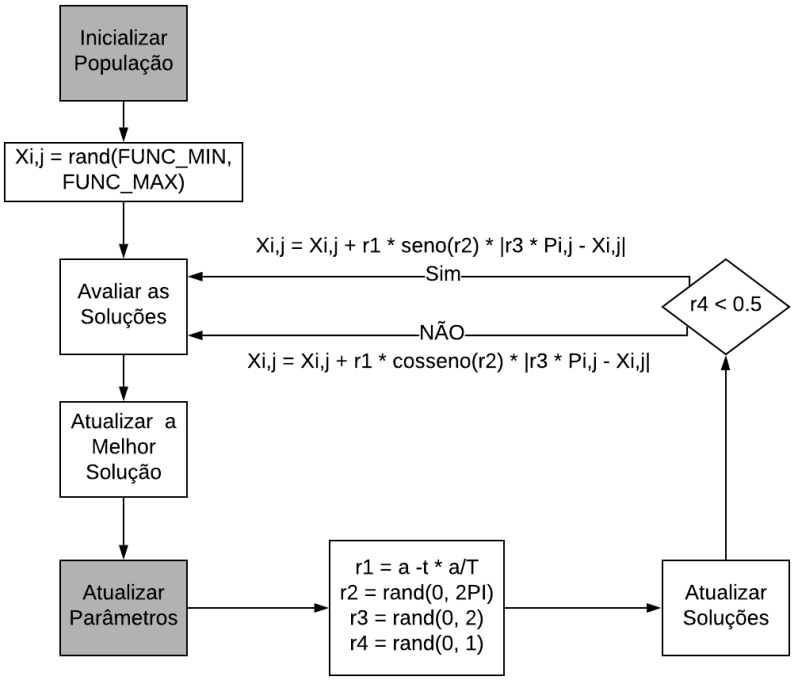

Figura 2.13: Diagrama de Blocos do SCA

Fonte: Própria autora.

2.4 Problema de Predição da Estrutura de Proteínas

Proteínas são estruturas básicas que compõem todos os seres vivos, e são compostas por uma cadeia de aminoácidos que são ligados entre si através de ligações peptídicas (PARPINELLI; LOPES, 2013). Nas ligações peptídicas, o grupo carboxila de um aminoácido se une ao grupo amino de outro aminoácido e a molécula de água produzida (JANA; DAS; SIL, 2018). A cadeia resultante, que forma a proteína, é referida como uma cadeia peptídica e é utilizada para formar uma longa sequência de aminoácidos (JANA; DAS; SIL, 2018). As proteínas são dobradas em uma estrutura única durante o processo de síntese e, suas funções biológicas são determinadas a partir dessas estruturas, tendo um papel importante no desenvolvimento de medicamentos, predição de doenças e outras aplicações (JANA; SIL; DAS, 2017).

A maioria das estruturas proteicas são determinadas utilizando métodos de cristalografia de raios-X e ressonância magnética nuclear (NMR), entretanto, esses métodos nem sempre são viáveis para se conduzir - por isso, pesquisadores foram motivados a desenvolver métodos computacionais para determinar a estrutura de proteínas, a partir de

dois passos: em primeiro lugar, um modelo físico correspondente a uma função de energia da proteína é desenvolvido e, em segundo lugar, o mínimo global da função de energia é obtido usando um algoritmo de otimização (JANA; SIL; DAS, 2017). Um desses modelos físicos é o modelo AB *off-lattice*, que será utilizado neste trabalho. É importante destacar que o problema PSP baseado no modelo *off-lattice* 2D e 3D é um problema de otimização complexo, NP-*hard* (JANA; DAS; SIL, 2018).

2.4.1 Modelo *off-lattice* 2D-AB

Nesse modelo as sequências de proteínas são compostas por apenas dois tipos de monômeros (moléculas que se combinam com outras para formar um polímero (RUDIN; CHOI, 2013)), classificados a partir das suas afinidades com a água: "A" para aminoácidos hidrofóbicos e "B" para aminoácidos hidrofílicos (ou polares) - além disso, os monômeros possuem uma unidade de comprimento de distância entre eles, de modo que cada monômero é conectado com o próximo na cadeia através de uma ligação que forma um ângulo relativo ao seu antecessor (PARPINELLI; LOPES, 2013).

No modelo AB, uma proteína composta de n monômeros necessita de n - 2 ângulos para ser representada, e esses ângulos são definidos no intervalo $[-\pi, \pi]$ (PARPINELLI; LOPES, 2013). Um exemplo de representação pode ser vista na figura 2.14, onde tem-se uma sequência com oito aminoácidos e seis ângulos.

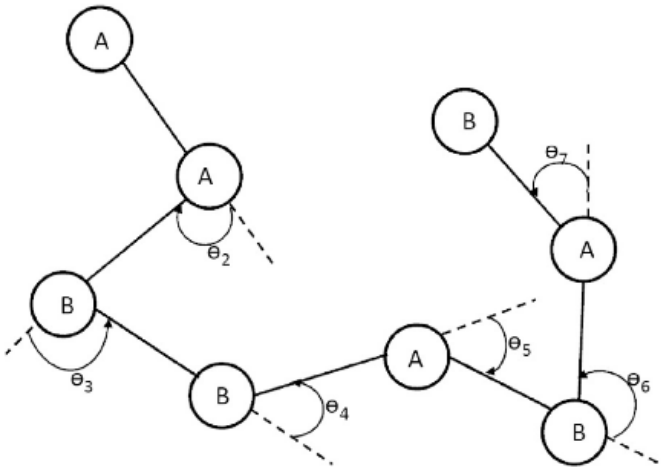

Figura 2.14: Representação do modelo *off-lattice* 2D-AB para a sequência proteica AABBABBAB

Fonte: (JANA; SIL; DAS, 2017)

O modelo também define valores de energia para os monômeros: "A" possui

energia 1 e "B" possui energia -1 e, considerando dois monômeros genéricos i e j, e os tipos ξ_i e ξ_j, tem-se que a interação entre os monômeros leva a valores diferentes de energia potencial C: valores positivos representam atração e valores negativos, repulsão - ligações "AA" têm energia 1 (atração forte); ligações "BB" têm energia $\frac{1}{2}$ (atração fraca); e ligações "AB" ou "BA" têm energia $-\frac{1}{2}$ (repulsão fraca) (PARPINELLI; LOPES, 2013).

A energia E da estrutura de uma proteína com n monômeros é dada pela equação 2.19 (PARPINELLI; LOPES, 2013).

$$E(\vec{\theta}, \vec{\xi}) = \sum_{i=1}^{n-1} V_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} V_2(d_{ij}, \xi_i, \xi_j) \tag{2.19}$$

Essa equação postula dois tipos de energias potenciais intermoleculares, V_1 e V_2, onde a primeira representa potenciais *backbone*, depende apenas do ângulo entre monômeros e é calculada a partir da equação 2.20; e a segunda, definida pela equação 2.21, representa a energia potencial presente nas interações sem limites e é conhecida como o potencial *Lennard-Jones* (PARPINELLI; LOPES, 2013).

$$V_1(\theta_i) = \frac{1}{4} * (1 - cos(\theta_i)) \tag{2.20}$$

$$V_2(d_{ij}, \xi_i, \xi_j) = 4 * (d_{ij}^{-12} - C(\xi_i, \xi_j) * d_{ij}^{-6}) \tag{2.21}$$

onde

$$C(\xi_i, \xi_j) = \frac{1}{8 * (1 + \xi_i + \xi_j + 5 * \xi_i * \xi_j)} \tag{2.22}$$

A equação 2.22 é a energia potencial da interação entre os monômeros i e j, e d_{ij} é a distância entre esses monômeros na cadeia, de modo que $i < j$ (PARPINELLI; LOPES, 2013).

3 Revisão da Literatura

Neste capítulo será apresentado o Mapeamento Sistemático da Literatura (MSL) realizado, com intuito de proporcionar uma visão geral da área de pesquisa deste trabalho, que envolve a aplicação de métodos de randomização (diferentes do Uniforme) em meta-heurísticas. A metodologia utilizada para o desenvolvimento deste mapeamento é baseada nas diretrizes propostas por (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). O trabalho foi realizado no mês de Abril do ano de 2019.

3.1 Método de Busca

Para a execução da busca, foi definida uma string de busca, apresentada na tabela 3.1. Inicialmente, alguns termos de interesse foram estabelecidos, como: *probability distribution*, *randomization methods* e a versão em inglês britânico do mesmo termo; assim também como *chaotic series*, *chaotic sequence* e *chaotic map*. Essas expressões foram utilizadas para encobrir as possíveis representações utilizadas para os diferentes métodos de randomização, incluindo mapas caóticos. Além disso, o termo *metaheuristic* foi utilizado para representar o tipo de algoritmo que estamos buscando, que neste caso, são as meta-heurísticas.

("randomization methods" OR "randomisation methods" OR "probability distribution" OR "chaotic series" OR "chaotic sequence" OR "chaotic map") AND metaheuristic

Tabela 3.1: String de Busca

A *string* definida foi executada em três mecanismos de busca distintos: Engineering Village, Scopus e IEEE Xplore. Como pode-se observar na tabela 3.2, ao fazer a busca, reunindo o resultado de todas as bases, foi encontrado um total de 222 artigos, de onde destes apenas 143 estavam disponíveis na íntegra.

Mecanismo	Quantidade
Engineering Village	26
Scopus	71
IEEE Xplore	125
Total	222
Disponíveis	143
Considerados	43

Tabela 3.2: Resultados dos Mecanismos de Busca

3.2 Seleção dos Estudos Primários

Após a realização da busca, foi feito o *download* dos 143 artigos disponíveis e os mesmos passaram por um processo de seleção com o intuito de fazer a identificação dos estudos primários. Neste processo, foram analisados o título, as palavras-chave, o *abstract* e quando necessário, caso existisse dúvidas em relação a inclusão ou exclusão, a introdução e a conclusão dos artigos. Foram aplicados os seguintes critérios de inclusão:

- Artigos completos, de 4 páginas ou mais;
- Artigos que apresentam uma nova versão de alguma meta-heurística, utilizando um método de randomização diferente do original.

Os critérios de exclusão aplicados foram:

- Artigos publicados fora do período de 2009 até 2019;
- Artigos duplicados;
- Artigos que não propõem uma nova versão de uma meta-heurística, apenas a utilizam para aplicação em algum problema;
- Artigos que apresentam algoritmos que não são baseados em população;
- Artigos que apresentam como método de randomização apenas a distribuição Uniforme.

Após a aplicação dos critérios de inclusão e exclusão mencionados, restaram 43 artigos, como visto na tabela 3.2. As referências dos artigos podem ser encontradas na seção de Bibliografia deste trabalho, indicados pelo símbolo *. Após esse processo, foi realizado um processo de extração e análise dos dados coletados, apresentado na seção 3.4.

3.3 Questões de Pesquisa

Para compreender melhor os trabalhos encontrados no mapeamento, foram definidas algumas questões de pesquisa a serem respondidas:

- **QP1** - Qual o método de randomização que o estudo utiliza?
- **QP2** - Qual foi o motivo para realizar a adaptação do algoritmo com este método de randomização?
- **QP3** - Em que parte do algoritmo este método de randomização foi aplicado?

3.4 Análise dos Resultados

Nessa seção serão apresentados os resultados do Mapeamento Sistemático da Literatura, começando com uma visão geral, com informações pertinentes sobre os estudos; e depois apresentando as respostas para as questões de pesquisa propostas. É válido destacar que dos 43 estudos inclusos, 5 não serão incluídos nas análises e nos gráficos - isto porque esses estudos são repetidos, possuindo a mesma abordagem de estudos já inclusos, mudando apenas os problemas aos quais essas abordagens são aplicadas.

3.4.1 Visão Geral

A primeira característica analisada nos artigos foi o ano de publicação dos mesmos. Como pode-se observar na figura 3.1, tem-se que a quantidade de artigos abordando o tema de diferentes métodos de randomização em meta-heurísticas têm crescido ao longo dos anos, desde 2011 até 2018. Sendo assim, pode-se concluir que o interesse por essa área têm aumentado, e possui a tendência de aumentar ainda mais. Esse fato também nos mostra

que cada vez mais pesquisadores têm percebido a importância da randomização para as meta-heurísticas e decidido verificar quais são os melhores métodos para serem utilizados. Além disso, o ano de 2019 teve apenas 3 artigos publicados até então, mas precisamos levar em consideração que apenas os três primeiros meses do ano foram contabilizados no mapeamento - dessa forma, mais artigos podem ser publicados ao longo do ano.

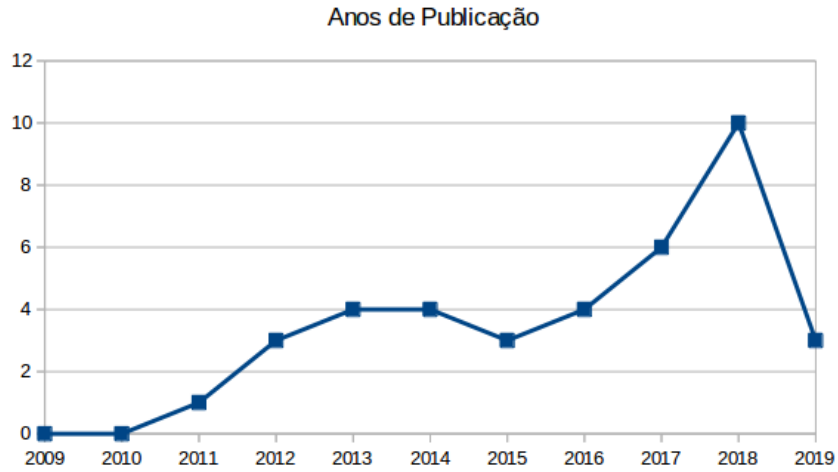

Figura 3.1: Anos de Publicação dos Estudos

Fonte: Própria autora.

Outro ponto interessante de se analisar é qual o algoritmo que os estudos utilizaram. Um artigo, do (*MITIĆ et al., 2018), apresentou diversos algoritmos, como o *Bat Algorithm, Firefly Algorithm, Accelerated Particle Swarm Optimization* e *Grey Wolf Optimizer*. Todos os outros estudos realizaram a adaptação de apenas uma única meta-heurística. Como pode-se observar na figura 3.2, diversos algoritmos foram utilizados. Entretanto, o ***Firefly Algorithm*** se destacou pela sua popularidade, sendo abordado em 8 estudos, seguido do ***Bat Algorithm*** e do ***Grey Wolf Optimizer***, com 4 estudos; o *Cuckoo Search Algorithm* foi usado em três artigos; e em sequência, o *Imperialist Competitive Algorithm* e o *Colliding Bodies Algorithm*, que foram utilizados em 2 estudos. A categoria Outros do gráfico engloba todos os outros algoritmos, que apareceram apenas uma vez nos artigos. Essa categoria inclui os seguintes algoritmos: *Real-Coded Chemical Reaction Optimization*, *Backtracking Search Optimization Algorithm*, *Grasshopper Optimization Algorithm*, *Dragonfly Algorithm*, *Fruit Fly Optimization Algorithm*, *League Championship Algorithm*, *Accelerated Particle Swarm Optimization*, *Intelligent Tuned Harmony Search*, *Crow Search Algorithm*, *Salp Swarm Algorithm*, *Darwinian Particle Swarm Optimization*, *Symbiotic Organisms Search*, *Wind Driven Optimization*, *Particle Swarm Optimization*, *Self-Adaptive Differential Evolution*, *Monarch Butterfly Optimization Algorithm*, *Artificial*

Bee Colony e *Whale Optimization Algorithm*.

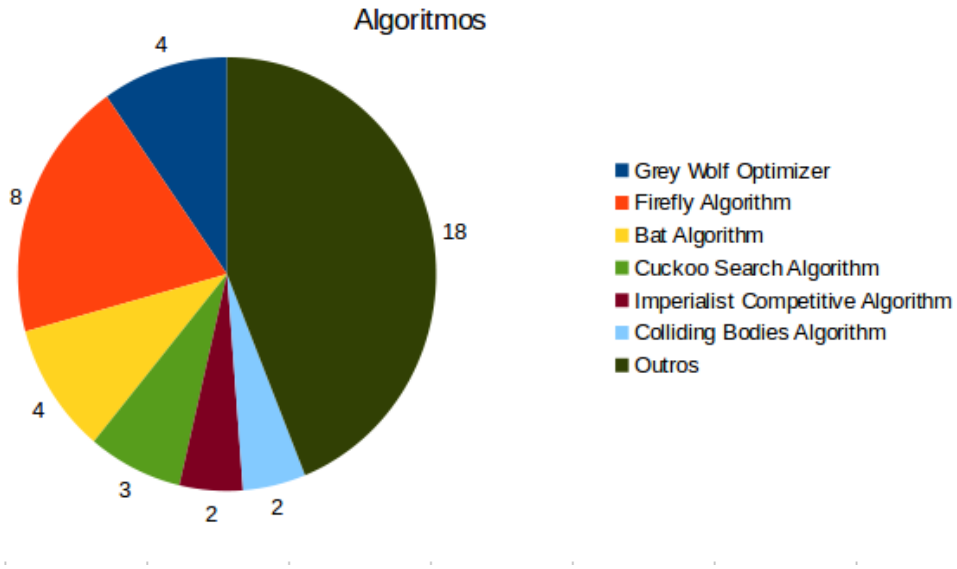

Figura 3.2: Algoritmos Utilizados nos Estudos

Fonte: Própria autora.

3.4.2 Métodos de Randomização (QP1)

Para responder a primeira questão de pesquisa, foram levantadas as distribuições utilizadas pelos autores ao adaptarem as meta-heurísticas apresentadas nos artigos. No total, 33 métodos de randomização foram encontrados. Na figura 3.3 tem-se os tipos de distribuições que foram usadas e as suas respectivas ocorrências, sendo que a soma total do gráfico se apresenta superior ao número de artigos, visto que na maioria dos artigos os autores utilizaram mais de um método nas abordagens. Como pode-se perceber, os mapas caóticos foram de longe o tipo de distribuição mais utilizada. Isso mostra como há um grande interesse na área pela aplicação de Teoria do Caos em algoritmos de otimização, e também que os mesmos foram selecionados por terem obtido um impacto positivo na literatura, mostrando como podem ser utilizados para se obter um melhor desempenho. Além dos mapas, tem-se a distribuição Gaussiana, com 5 ocorrências; o Levy Flights, com 4; e as distribuições de Cauchy, Exponencial, de Rayleigh, Beta, Fat-tailed e de Levy, com 1 ocorrência cada.

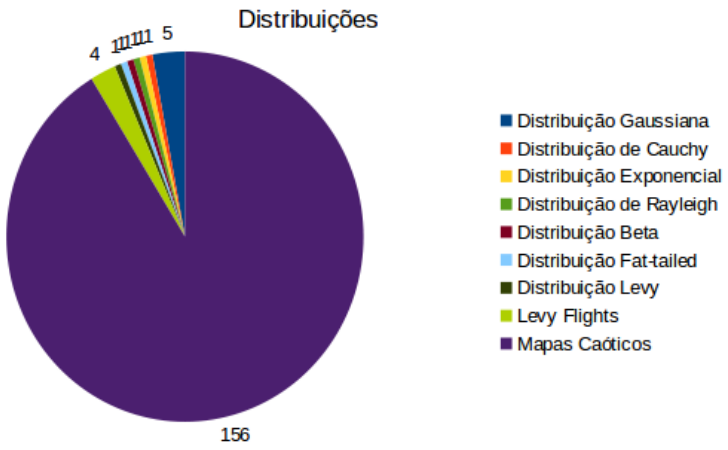

Figura 3.3: Distribuições Utilizadas nos Estudos

Fonte: Própria autora.

Na tabela 3.3 tem-se de forma mais específica cada mapa que foi usado nos estudos, onde a soma total representa as 156 ocorrências totais dos mapas caóticos. O mais utilizado foi o **mapa Logístico**, com 22 ocorrências. Isso não é surpreendente se for levado em consideração que ele é o mapa caótico mais conhecido, além de ser um mapa de simples implementação. Em sequência, tem-se o **mapa Sinusoidal**, com 17 ocorrências; o **mapa Gauss**, com 16; e assim por diante.

Além disso, as distribuições encontradas foram classificadas de acordo com a sua finalidade quando aplicadas à algoritmos de otimização: algumas intensificam o espaço de busca, outras diversificam e outras ainda podem apresentar ambas as finalidades - de acordo com os parâmetros utilizados na distribuição. Por exemplo, a distribuição Gaussiana possui dois parâmetros: a média e o desvio-padrão. Quando o desvio-padrão assume um valor pequeno, a distribuição realiza uma intensificação, e quando assume um valor grande, realiza uma diversificação, como pode-se visualizar na figura 3.4.

Essa mesma situação ocorre também com algumas outras distribuições como a Cauchy, onde quanto maior o parâmetro de escala mais a distribuição diversifica; a Exponencial, onde acontece a mesma coisa com seu parâmetro de *rate*; a de Rayleigh, onde ocorre a mesma situação com o parâmetro de escala; a distribuição Beta, que quando possui seus parâmetros de forma igual a 1, diversifica o espaço de busca, e caso contrário, intensifica; e por fim o Levy Flights, que por utilizar a distribuição de Levy, diversifica quando o parâmetro de escala é grande e intensifica quando o parâmetro de escala é pequeno. As outras distribuições, os chamados mapas caóticos, também possuem parâmetros. Entretanto, esses parâmetros são responsáveis apenas pelo comportamento caótico

Mapa	**Ocorrências**
Mapa Beta	1
Mapa Logístico	22
Mapa Chebyshev	9
Mapa Burgers	2
Mapa Circle	14
Mapa Sinusoidal	17
Mapa Gauss	16
Mapa Iterativo	8
Mapa Piecewise	11
Mapa Sine	8
Mapa Singer	10
Mapa Tent	15
Mapa de Intermitência	4
Mapa Liebovitch	5
Mapa Sawtooth	1
Mapa Hénon	2
Mapa Tinkerbell	2
Mapa ICMIC	1
Mapa Sinus	2
Mapa Arnold Cat	1
Mapa Delayed Logistic	1
Mapa Dissipative Standard	1
Mapa Ikeda	1
Mapa Lozi	1
Mapa Sinai	1

Tabela 3.3: Mapas Utilizados nos Estudos

exibido pelos mesmos, sendo assim, os valores aplicados a eles são fixos e retirados da literatura.

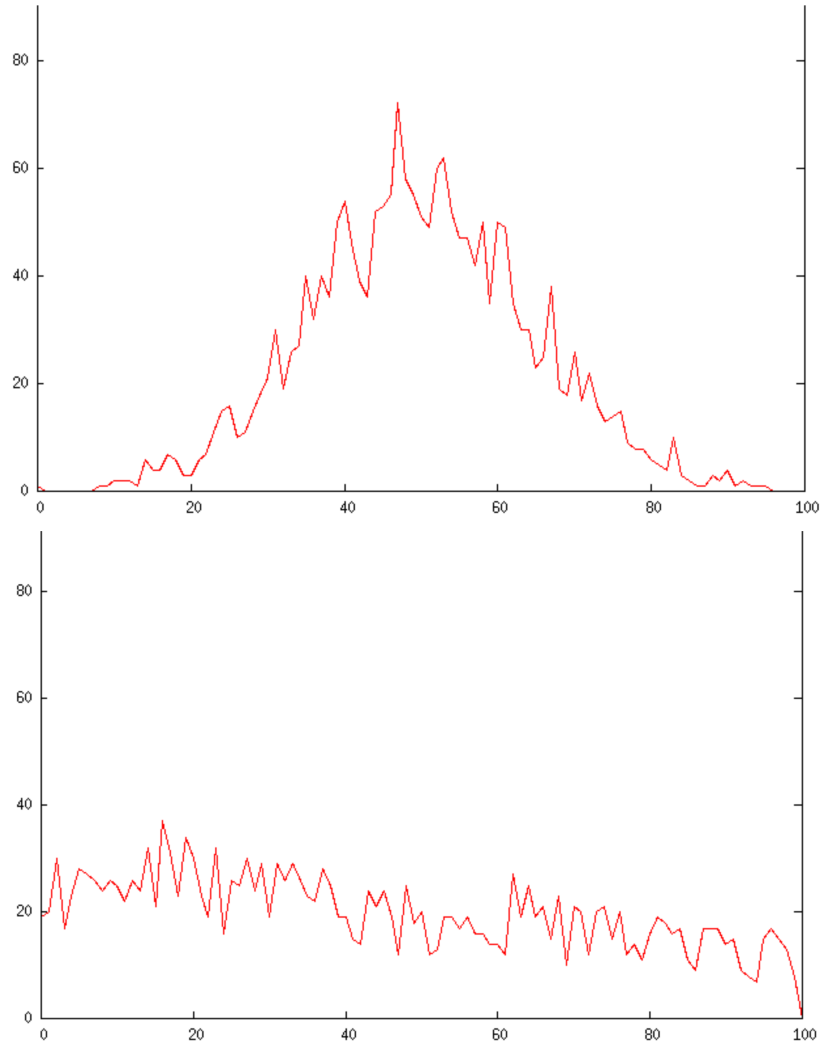

Figura 3.4: Representação 1D da Distribuição Gaussiana com Diferentes Desvios-padrões

Fonte: Própria autora.

Ademais, alguns mapas caóticos apresentaram a característica de intensificação, como o Mapa Beta, Logístico, Circle, Sinusoidal, Gauss, Chebyshev, Sine e Singer; e outros, apresentaram a característica de diversificação, como o Mapa Iterativo, Piecewise e Tent.

3.4.3 Motivação (QP2)

Diversas questões foram apontadas como motivação para realizar a adaptação dos algoritmos com os seus respectivos métodos de randomização, sendo que a maioria dos estudos apresentou mais de uma motivação. Algumas motivações mais óbvias foram citadas, como

por exemplo que a adaptação foi realizada para melhorar o desempenho do algoritmo de forma geral, melhorando sua eficiência e capacidade de convergência - citada por um total de 15 artigos. Outro ponto apresentado foi o impacto positivo que mapas caóticos tiveram na literatura, citado por 3 estudos. Ademais, foram citados por artigos individuais a necessidade de se estudar o impacto no resultado final do algoritmo, de diferentes distribuições como funções de pertubação, no trabalho de (*YU; LAM; LI, 2012); e foi citado como motivo no estudo de (*ISMAIL et al., 2013), que a distribuição Levy foi utilizada por ser a forma mais rápida de se encontrar um alvo randomicamente escondido.

As motivações restantes podem ser classificadas em quatro categorias - a análise de diversificação, a intensificação, a diversificação e o balanço entre busca global/local, como pode-se observar na figura 3.5. A categoria de análise de diversidade se refere ao trabalho de (*SENKERIK et al., 2018), que apontou a necessidade de se pesquisar o impacto que séries caóticas possuem na diversidade da população. A categoria de intensificação engloba artigos que apontaram a inclusão de métodos de randomização como forma de intensificar o processo de busca, acelerando a velocidade de convergência do algoritmo. Já a categoria de diversificação engloba estudos que fizeram essa inclusão para diversificar a população, com objetivo de evitar problemas como a convergência prematura e estagnação em ótimos locais. Por fim, a categoria de balanço entre busca global/local inclui artigos que tiveram como motivação melhorar o balanço do algoritmo entre busca local (intensificação) e busca global (diversificação).

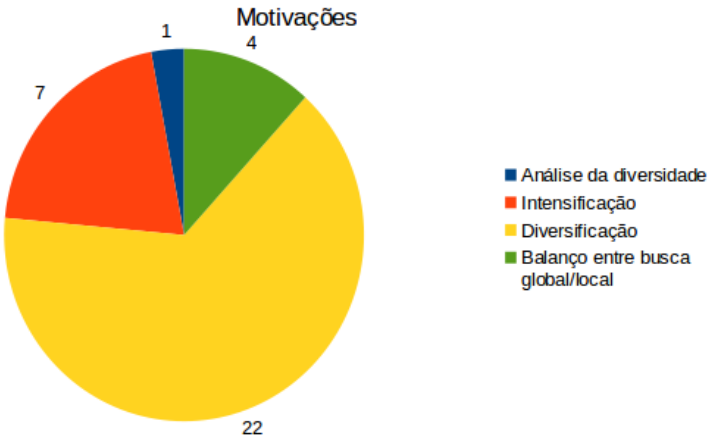

Figura 3.5: Categorias de Motivações Apresentadas nos Estudos

Fonte: Própria autora.

Pode-se perceber através das motivações que realizar a mudança de uma distribuição Uniforme para outra distribuição pode impactar imensamente os resultados do

algoritmo - melhorando seu desempenho, seu balanço entre intensificação e diversificação, e evitando problemas como a convergência lenta e prematura, e a estagnação em ótimos locais. Não só isso, mas também pode-se observar na figura 3.5 que a diversificação é uma das maiores motivações para a aplicação de diferentes métodos de randomização nos algoritmos - isso se dá pelo fato de que quer-se sempre manter a diversidade presente até as últimas iterações, visto que todos esses algoritmos populacionais sofrem de problemas como a convergência prematura.

3.4.4 Aplicação (QP3)

As respostas da terceira e última questão de pesquisa mostram que, na maioria dos casos, os métodos de randomização são aplicados em pontos específicos dos algoritmos. Por exemplo, no trabalho de (*YU; LAM; LI, 2012) as distribuições são incorporadas no operador de busca de vizinhança e na operação de decomposição; em (*ASKARZADEH; COELHO, 2014) o mapa caótico é incorporado no fator de escala relacionado ao processo de mutação; no estudo de (*COELHO; BERNERT; MARIANI, 2011), é incorporado no coeficiente de absorção e no parâmetro de randomização; no artigo de (*FARSHIN; SHARIFIAN, 2017), é incorporado na função de atualização dos vetores de coeficiente; em (*LIANG et al., 2018) é incorporado nos parâmetros de sonoridade e taxa de emissão de pulso; em (*LIN; LI, 2012) é incorporado no Levy Flights proposto para substituir o Random Walk; em (*COELHO et al., 2013a) a distribuição é inserida no parâmetro de randomização α; em (*WANG et al., 2016) é inserida no tamanho do passo α; em (*KAVEH; DADRAS; MONTAZERAN, 2018) as distribuições são inseridas de três formas distintas: primeiramente, na probabilidade de mudança do algoritmo; em segundo, são usadas para selecionar as variáveis candidatas para regeneração; e em terceiro, são usadas na regeneração da variável selecionada.

No trabalho de (*MITIĆ et al., 2015) são inseridas em um novo parâmetro chamado α, usado para a geração de origens de comida; em (*BINGOL; ALATAS, 2016) são incorporadas de seis formas diferentes, todas envolvendo os parâmetro r, r1 e r2 do algoritmo; em (*MITIĆ et al., 2018) são incorporadas no valor randômico r e no parâmetro de exploração C; em (*COELHO et al., 2018) a distribuição é incorporada nos parâmetros de consciência e tamanho do voo; no artigo de (*AHMED et al., 2018) são inseridas no parâmetro c3; em (*GANDOMI et al., 2013) são inseridas nos coeficientes de absorção de luz e atratividade; em (*COELHO et al., 2013b) o mapa caótico é inserido no coeficiente de

absorção de luz; no trabalho de (*WOOD, 2016) o mapa caótico é incorporado no fator de escala e a distribuição *fat-tailed* no fator de ajustamento estocástico; em (*TALATAHARI et al., 2012) são incorporados nos parâmetros randômicos da equação de assimilação; em (*AYALA et al., 2017), são incorporados nos parâmetros de controle na fase de mutualismo; em (*ISMAIL et al., 2013), a distribuição foi incorporada na função de atualização de posição das partículas; em (*GHOLIZADEH; ALIGHOLIZADEH, 2019) foram inseridas na função de mutação e de atualização de posição; em (*DING et al., 2018) o mapa Logístico foi aplicado nas estratégias de *encircling prey* e *bubble-net attacking* e o mapa Sinusoidal na mutação; em (*MORTAZAVI; KHAMSEH; NADERI, 2015) onde foram aplicados em uma porcentagem de colônias revoltadas; e em (*GHOLIZADEH; BAGHCHEVAN, 2017) onde foram aplicados na funções de atualização, mutação e geração de novos indivíduos para a próxima geração.

A segunda maioria dos casos aplica as distribuições em todos os parâmetros do algoritmo que necessitam de randomização. Como o trabalho de (*GANDOMI; YANG, 2014); de (*SAYED; THARWAT; HASSANIEN, 2019); de (*TURGUT; TURGUT; COBAN, 2014); de (*AYALA et al., 2016), onde as distribuições são incorporadas em todos os parâmetros de controle: R, T, c, g e α; de (*SENKERIK et al., 2018); de (*JADDI; ABDULLAH; HAMDAN, 2015);e o de (*FENG et al., 2018), onde se utiliza a distribuição gaussiana na atualização do pior indivíduo e as séries caóticas em todos os outros pontos de randomização.

E além desses casos, os métodos também foram aplicados no mecanismo de balanço entre diversificação e intensificação do algoritmo, como no caso do trabalho de (*SAXENA; KUMAR; DAS, 2019), onde o mapa Beta é aplicado no vetor de controle responsável por realizar a busca global e local ao longo das iterações do GWO; no trabalho de (*SAXENA; SHEKHAWAT; KUMAR, 2018), onde os mapas são aplicados no parâmetro c, de redução da zona de conforto, o qual é o mecanismo de ponte que liga a intensificação e a diversificação do algoritmo; e no trabalho de (*OLIVEIRA et al., 2017) onde foram aplicados no vetor de controle e no parâmetro C.

Alguns artigos mostraram que os métodos também foram incorporados na geração da população inicial das meta-heurísticas, como no trabalho de (*BOUSHAKI; KAMEL; BENDJEGHABA, 2017); no artigo de (*SURESH; LAL, 2017), onde o mapa caótico é aplicado tanto na geração da população inicial como na velocidade de *drift* dos agentes; no estudo de (ABRO; MOHAMAD-SALEH, 2014); e no trabalho de (*CHOU; NGO, 2017) onde o mapa Logístico é utilizado na inicialização da população, o mapa Gauss no parâmetro

de atratividade e o Levy Flights para realizar a movimentação das *fireflies*.

Um ponto importante, que recebe destaque, é a forma como o trabalho de (*WOOD, 2016) realizou a aplicação da distribuição no algoritmo. No estudo, o *Cuckoo Search Algorithm* é adaptado com várias melhorias, uma delas sendo na parte de troca do algoritmo, mais especificamente na equação que gera novos indivíduos para substituir indivíduos com *ranking* baixo ou médio (*WOOD, 2016). Nessa equação, existe um parâmetro F, chamado de fator de ajuste estocástico, que é retirado de uma distribuição *fat-tailed*, amostrada através de uma janela de amostragem entre os limites de probabilidade inferiores e superiores, que variam dinamicamente conforme as iterações progridem (*WOOD, 2016). Sendo assim, o campo distante de randomização é mais amostrado nas iterações iniciais (fazendo o algoritmo realizar uma diversificação / busca global no espaço de solução) e o campo próximo é mais amostrado nas iterações finais (fazendo o algoritmo realizar uma intensificação / busca local no espaço de solução) (*WOOD, 2016). Ou seja, o autor utiliza a distribuição como uma forma de favorecer os processos de intensificação e diversificação do espaço de busca.

Finalmente, pode-se concluir a partir da análise desses estudos, que o local de aplicação dos métodos de randomização pode depender do algoritmo, sendo aplicado em pontos específicos do mesmo. Entretanto, os métodos também podem ser aplicados em locais que favoreçam a intensificação ou a diversificação do algoritmo, como na geração da população inicial; além de locais que envolvem o balanço entre a busca global e local, como os mecanismos de ponte.

4 Proposta

Neste capítulo serão apresentadas as propostas de adaptação realizadas nos algoritmos abordados neste trabalho.

4.1 Motivação

De acordo com o mapeamento sistemático, apresentado na seção 3, tem-se diversas formas de se aplicar diferentes métodos de randomização em um algoritmo: em pontos específicos; em todos os pontos que necessitam de aleatoriedade; somente na geração da população inicial; entre outros. Sendo assim, foi-se decidido abordar duas destas aplicações neste trabalho: a aplicação em todos os pontos de randomização, por ser intuitiva e uma das mais fáceis de se implementar; e a aplicação em pontos específicos do algoritmo, por ser a abordagem mais utilizada dentre os artigos do mapeamento.

4.2 Aplicação em Todos os Pontos do Algoritmo

A abordagem que envolve aplicar o método de randomização em todos os pontos de aleatoriedade do algoritmo envolve apenas um passo: identificar quais são os momentos em que a meta-heurística necessita gerar um número aleatório. Os dois algoritmos utilizados neste trabalho, SCA e jDE, foram adaptados desta forma, de acordo com os diagramas apresentados nas figuras 2.13 e 2.12, que mostram através dos blocos cinzas quais são os pontos que necessitam de randomização nos algoritmos em questão. Os métodos de randomização aplicados foram a distribuição Gaussiana e o mapa caótico Logístico, por serem as distribuições mais utilizadas - além da distribuição Uniforme, usada para comparação. Essas versões adaptadas dos algoritmos foram então testadas com um conjunto de funções *benchmark* e um problema do mundo real, o PSP.

4.3 Aplicação em Pontos Específicos do Algoritmo

O primeiro passo necessário para realizar as adaptações dos algoritmos em pontos específicos foi analisar quais parâmetros dos mesmos são relevantes e possuem um grande impacto no comportamento de intensificação e diversificação dessas meta-heurísticas. Fazendo esta análise, tem-se que o algoritmo jDE, como visto na seção 2.3.1, possui três parâmetros: F, CR e NP. Entretanto, os parâmetros F e CR já são auto-adaptados por padrão, e o parâmetro restante, NP, não possui um grande impacto no comportamento do algoritmo - sendo assim, foi-se decidido não adaptar essa meta-heurística. Por outro lado, o algoritmo SCA, como visto na seção 2.3.2, possui quatro parâmetros, apresentados na tabela 4.1.

Parâmetro	**Função**	**Geração**	**Intervalo**
r1	Determina a direção do movimento	Auto-adaptado	[2, 0]
r2	Determina o tamanho do movimento	Uniforme	[0, 2π]
r3	Enfatiza ou não o efeito da melhor solução na definição da distância	Uniforme	[0, 2]
r4	Alterna entre as fórmulas de update com seno e cosseno	Uniforme	[0, 1]

Tabela 4.1: Parâmetros do SCA

Destes quatro parâmetros, o parâmetro r1 é auto-adaptado, e o r4 não possui grande relevância no comportamento do algoritmo, pois tem apenas a função de trocar entre as fórmulas de atualização das soluções. Sendo assim, foram identificados que os dois parâmetros restantes (r2 e r3) possuem grande importância para o algoritmo, e são bons candidatos para adaptação.

O segundo passo necessário é analisar como esses parâmetros relevantes afetam o comportamento do algoritmo, e se o comportamento resultante é interessante. De acordo com (QU et al., 2018), nas equações de atualização das soluções do SCA, apresentadas na seção 2.3.2, tem-se que os termos dados pelas equações 4.1 e 4.2 são responsáveis por guiar juntamente a habilidade de intensificação e diversificação do algoritmo: quando os valores desses termos forem maiores que 1 ou menores que -1, o algoritmo realiza uma diversificação, enquanto quando os valores estão no intervalo $[-1, 1]$, ele realiza uma intensificação.

$$r1 * sin(r2) \tag{4.1}$$

$$r1 * cos(r2) \tag{4.2}$$

Levando essa informação em consideração em conjunto com o fato de que r1 é um parâmetro que decresce linearmente de 2,0 até 0,0, pode-se fazer um mapeamento que mostra o intervalo dos possíveis valores dos termos 4.1 e 4.2 para cada valor de r1. Esses valores foram obtidos substituindo cada possível valor de r1 nas equações, além de utilizar o fato de que as funções seno e cosseno variam sempre dentro do intervalo [-1, 1]. Com isso, tem-se um mapeamento como descrito no quadro 4.2, que mostra que para cada valor de r1, independente do valor de r2, o valor resultante dos termos 4.1 e 4.2 estarão nos respectivos intervalos apresentados.

Valor de r1	**Intervalo do Termo**
2,0	[-2,0, 2,0]
1,9	[-1,9, 1,9]
...	...
1,0	[-1,0, 1,0]
...	...
0,1	[-0,1, 0,1]
0,0	[0,0, 0,0]

Tabela 4.2: Mapeamento de Valores

Com esse mapeamento, pode-se analisar como é o comportamento de intensificação/diversificação do SCA ao longo de suas iterações. No começo do algoritmo, para os valores de r1 de 2,0 até maiores que 1,0, tem-se que o algoritmo tem a possibilidade de diversificar ou intensificar, dependendo do valor do parâmetro r2. E conforme o processo de otimização do algoritmo avança, tem-se que com valores de r1 iguais ou menores a 1,0 o algoritmo intensifica, independente do valor de r2, visto que os possíveis valores para os termos 4.1 e 4.2 sempre estão dentro do intervalo de [-1, 1].

Baseado nisso, pode-se concluir que o comportamento do algoritmo no início do processo de otimização não é interessante. Pois, como dito anteriormente, no começo do algoritmo pode-se realizar uma intensificação ou uma diversificação, baseado no valor

do parâmetro r2. Entretanto, no algoritmo padrão, r2 é um valor gerado aleatoriamente de forma uniforme: sendo assim, pode-se acabar intensificando as soluções desde o começo do algoritmo, nunca realizando uma diversificação. Assim como também não tem-se nenhuma regra que permita que esses valores aleatórios tendam mais a estar no intervalo referente a diversificação no início do processo.

Esse comportamento é exatamente o oposto do desejado, visto que em qualquer meta-heurística quer-se sempre realizar uma diversificação no começo do algoritmo, para explorar de forma adequada todo o espaço de busca e conseguir encontrar o respectivo ótimo global pelo qual se procura.

Levando em consideração o comportamento de intensificação e diversificação do algoritmo SCA e os seus respectivos parâmetros que possuem o maior impacto nesse comportamento e que ainda não são adaptados de alguma forma, apresenta-se aqui uma proposta de adaptação dos parâmetros r2 e r3 do algoritmo SCA. A principal ideia da proposta apresentada é utilizar a distribuição Gaussiana no lugar da distribuição Uniforme para gerar valores aleatórios para os parâmetros r2 e r3. Entretanto, o desvio-padrão utilizado para essa distribuição vai ser adaptado para realizar uma incrementação linear, assim criando uma tendência para os valores que serão gerados.

Como explicado anteriormente, na seção 2.3.2, tem-se que o parâmetro r3 varia no intervalo de [0, 2], enfatizando (r3 $>$ 1) ou não (r3 $<$ 1) o efeito da melhor solução encontrada até então pelo algoritmo na definição da distância. Sendo assim, como quer-se que a melhor solução encontrada tenha um impacto menor no algoritmo no começo do processo de otimização e maior no final, o valor de r3 deve começar baixo e incrementar ao longo das iterações. Para isso, pode-se gerar esses valores de r3 através de uma distribuição Gaussiana com média 0.0 e desvio-padrão como na equação 4.3.

$$desvioR3 = iteracaoAtual * \frac{2.0}{maxIteracao} \tag{4.3}$$

Onde iteracaoAtual é o número da iteração na qual o algoritmo se encontra e maxIteracao é o número total de iterações do algoritmo. Dessa forma, tem-se que o desvio-padrão inicia com o valor 0 e incrementa o seu valor a cada iteração linearmente até chegar no valor 2 - sendo assim, o mesmo cria uma tendência a gerar valores menores (mais próximos de 0) para r3 no começo e valores maiores para r3 no final.

Para o parâmetro r2, quer-se fazer uma adaptação tal que o valor resultante

dos termos $sin(r2)$ e $cos(r2)$ decresça de 1 até -1, consequentemente fazendo com que o algoritmo realize uma diversificação no começo do processo de otimização e uma intensificação no final. Como o valor para esse parâmetro é gerado aleatoriamente, não pode-se garantir que os valores obrigatoriamente seguirão esse padrão, entretanto, utilizando a distribuição Gaussiana, pode-se fazer com que esses valores tendam a seguir o padrão desejado.

Para isso, a proposta para o parâmetro r2 é gerá-lo com uma distribuição Gaussiana e mudar seu intervalo de geração - de $[0,\ 2\pi]$ para $[\frac{\pi}{2},\ \frac{3\pi}{2}]$ para quando o algoritmo optar por usar a função seno na equação de atualização das soluções, e para $[0,\ \pi]$ para quando o algoritmo utilizar a função cosseno. Além disso, a distribuição Gaussiana usada possui média $\frac{\pi}{2}$ e desvio-padrão como na equação 4.4 para a função seno; e média 0 e desvio-padrão como na equação 4.5 para a função cosseno.

$$desvioR2Seno = \frac{\pi}{2} + iteracaoAtual * \frac{(\frac{3\pi}{2} - \frac{\pi}{2})}{maxIteracao} \tag{4.4}$$

$$desvioR2Cosseno = iteracaoAtual * \frac{\pi}{maxIteracao} \tag{4.5}$$

Além dessa adaptação, substituindo a distribuição Uniforme pela Gaussiana e modificando a maneira como o desvio-padrão é gerado para os parâmetros r2 e r3, também foi implementado, como parte da proposta, uma técnica chamada *crowding*. O *crowding* original, usado neste trabalho, compara os indivíduos da *offspring*, ou seja, os indivíduos que tiveram suas soluções atualizadas na atual geração, com alguns indivíduos selecionados aleatoriamente da população não atualizada: o indivíduo mais similar ao indivíduo proveniente da *offspring* é substituído pelo mesmo se possuir um *fitness* pior do que o dele (KUNDU et al., 2013). O fator de *crowding*, que se refere ao tamanho do grupo selecionado aleatoriamente para comparação, é normalmente definido como 2 ou 3 (KUNDU et al., 2013).

5 Experimentos

Neste capítulo serão apresentados os experimentos realizados neste trabalho, utilizando os algoritmos adaptados de acordo com as descrições da seção 4.

5.1 Experimentos com Benchmarks

Os experimentos realizados para esta etapa incluem utilizar os algoritmos jDE e SCA, adaptados de acordo com a aplicação em todos os pontos de aleatoriedade dos algoritmos, para a otimização de funções *benchmark*. A comparação das versões adaptadas com os diferentes métodos de randomização e da versão original dos algoritmos, que usam a distribuição Uniforme, foi realizada através do cálculo da média e desvio-padrão das execuções de cada algoritmo, além da análise dos gráficos de convergência e diversidade dos mesmos.

5.1.1 Configurações

As configurações utilizadas nos experimentos, para os dois algoritmos, foram as seguintes:

- Tamanho da população: 30;
- Gerações: 2000;
- Dimensões: 20;
- Execuções: 10;

O tamanho da população, dimensões e número de execuções foram retirados do trabalho de Mirjalili (2016). Já a quantidade de gerações também foram retiradas deste trabalho, entretanto, seu número foi quadruplicado visto que o original (500 gerações) era muito baixo. Além disso, em relação à distribuição Gaussiana, foi-se utilizada uma média igual a 0,5 e o desvio-padrão de 0,5, para a geração de números entre [0, 1] e, uma

média aleatória entre os limites inferior e superior da função e um desvio-padrão dado pela equação 5.1, para a geração da população inicial.

$$desvio = limiteSuperior - \frac{limiteInferior + limiteSuperior}{2,0} \tag{5.1}$$

5.1.2 Funções Benchmark

O conjunto de funções *benchmark* escolhido possui no total oito funções: Sphere, Rosenbrock, Rastrigin, Schaffer, Ackley, Griewank, Schwefel e Zakharov. Destas, apenas duas funções são unimodais, ou seja, não possuem mínimos locais, apenas um único mínimo global - Sphere e Zakharov. As outras seis funções são multimodais, ou seja, possuem diversos mínimos locais. As fórmulas das funções e seus respectivos mínimos globais podem ser vistos na tabela 5.1.

Função	**Fórmula**	**Mínimo Global**
Sphere	$\sum_{i=0}^{d} x_i^2$	0,0
Rosenbrock	$\sum_{i=0}^{d-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	1,0
Rastrigin	$\sum_{i=0}^{d} x_i^2 - 10cos(2\pi x_i)$	0,0
Schaffer	$(\sum_{i=0}^{d} x_i^2)^{0,25} * (sin(50 * (\sum_{i=0}^{d} x_i^2)^{0,1}))^2 + 1$	0,0
Ackley	$-20exp(-0,2\sqrt{\frac{1}{d}\sum_{i=0}^{d} x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)) + 20 + exp(1)$	0,0
Griewank	$\sum_{i=0}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$	0,0
Schwefel	$-\frac{(\sum_{i=0}^{d} x_i sin(\sqrt{\lvert x_i \rvert}))}{d}$	-418,9829
Zakharov	$\sum_{i=0}^{d} x_i^2 + (\sum_{i=0}^{d} 0,5ix_i)^2 + (\sum_{i=0}^{d} 0,5ix_i)^4$	0,0

Tabela 5.1: Descrição do Conjunto de Funções Usadas

5.1.3 Resultados e Análises

Os resultados dos experimentos executados podem ser observados na tabela 5.2, onde o primeiro valor encontrado é a média das execuções e o segundo valor, depois do símbolo $\pm$, é o desvio-padrão. Cada linha da tabela representa uma função do conjunto, e cada coluna da tabela representa uma das distribuições utilizadas, sendo que para cada uma destas, tem-se a separação entre os dois algoritmos, jDE e SCA. A última linha da tabela representa o somatório de vezes que determinado algoritmo foi estatisticamente o melhor

para certa função.

Os melhores resultados para cada função podem ser encontrados em negrito na tabela, sendo que os mesmos foram determinados da seguinte maneira: inicialmente, para cada função foi-se determinado o melhor resultado baseado na média das execuções de cada algoritmo, onde foi-se escolhido o que mais se aproximava do mínimo global daquela função. Em seguida, para poder verificar quais dos outros resultados eram equivalentes ao melhor, dois testes estatísticos foram utilizados, Kruskall-Vallis e o teste de Dunn. O teste Kruskall-Vallis foi realizado para verificar se existe ou não uma diferença estatística entre os conjuntos de resultados. No caso de existir alguma diferença, o teste de Dunn foi aplicado posteriormente, com o objetivo de apontar quais resultados são estatisticamente equivalentes e quais não são. Essa análise dos resultados dos testes é baseada em valores chamados de p-values. Com 95% de confiança, dois resultados são estatisticamente diferentes se o p-value entre eles for menor que 0,05.

Função	Uniforme		Gaussiana		Logístico	
	jDE	SCA	jDE	SCA	jDE	SCA
Sphere	3,9883e-53 ± 4,6228e-53	**1,3604e-132 ± 4,0813e-132**	2,6314e-43 ± 7,4168e-43	**4,1389e-71 ± 1,2406e-70**	2,2783e-13 ± 6,5587e-13	1,3873e-30 ± 4,1617e-30
Rosenbrock	**5,2117e+00 ± 3,3498e+00**	1,8567e+01 ± 0,2201e+00	2,8926e+01 ± 2,6698e+01	1,8660e+01 ± 0,2155e+00	**1,4413e+01 ± 1,7316e+01**	1,8897e+01 ± 0,0918e+00
Rastrigin	**0,0000e+00 ± 0,0000e+00**	**0,0000e+00 ± 0,0000e+00**	1,9899e+00 ± 1,7798e+00	**0,0000e+00 ± 0,0000e+00**	4,3778e+00 ± 2,0487e+00	**0,0000e+00 ± 0,0000e+00**
Schaffer	3,3123e-01 ± 1,0099e-01	**1,9088e-32 ± 5,7240e-32**	2,3277e+00 ± 1,1624e+00	**1,2143e-16 ± 3,6114e-16**	7,7709e+00 ± 7,9093e-01	9,4477e-14 ± 1,7980e-13
Ackley	3,9968e-15 ± 0,0000e+00	**4,4409e-16 ± 0,0000e+00**	2,5008e-05 ± 7,5023e-05	**4,4409e-16 ± 0,0000e+00**	4,1998e+00 ± 1,5783e+00	**4,4409e-16 ± 0,0000e+00**
Griewank	**0,0000e+00 ± 0,0000e+00**	**0,0000e+00 ± 0,0000e+00**	**0,0000e+00 ± 0,0000e+00**	**0,0000e+00 ± 0,0000e+00**	1,4099e-15 ± 3,4216e-15	**0,0000e+00 ± 0,0000e+00**
Schwefel	**-4,1602e+02 ± 3,9725e+00**	-1,1764e+02 ± 1,6606e+01	**-3,9766e+02 ± 1,8576e+01**	-1,3163e+02 ± 9,4722e+00	**-4,0546e+02 ± 8,7620e+00**	-1,3729e+02 ± 1,1659e+01
Zakharov	1,5326e-09 ± 2,4289e-09	**9,4473e-127 ± 1,9111e-126**	4,6938e-07 ± 5,8719e-07	**1,5898e-54 ± 4,7694e-54**	3,5924e-03 ± 4,4548e-03	9,7949e-38 ± 2,9384e-37
Destaques	4	6	2	6	2	3

Tabela 5.2: Resultados do Experimento com Benchmarks

Pode-se observar que para a maioria das funções, seis do total de oito, o mínimo global foi encontrado por pelo menos um dos algoritmos. As duas exceções foram a função Rosenbrock, na qual tanto os algoritmos originais assim como as adaptações tiveram dificuldade para convergir para o ponto ótimo; e a função Schwefel, na qual o algoritmo jDE chegou no ótimo em algumas execuções, mas a média das dez execuções não convergiu. Além disso, pode-se concluir a partir dos destaques, que os algoritmos que apresentaram o melhor desempenho dentre todos foram o SCA com distribuição Uniforme e o SCA com distribuição Gaussiana, sendo que os mesmos apresentaram as melhores convergências para seis das oito funções. Isso nos mostra como o algoritmo SCA teve um melhor desempenho quando comparado com o algoritmo jDE.

Em sequência, ficaram os algoritmos jDE com distribuição Uniforme, seguido do SCA com distribuição Logística e, por fim, os algoritmos que mostraram o pior desempenho foram o jDE com mapa caótico Logístico e o jDE com distribuição Gaussiana, apresentando uma boa convergência em apenas duas das oito funções do conjunto de *benchmarks*.

Além disso, em relação às distribuições, a Uniforme se destacou em todas as oito funções do conjunto, enquanto a Gaussiana mostrou melhores resultados em sete das oito *benchmarks* e, o mapa Logístico em apenas cinco. Sendo assim, pode-se concluir que a distribuição Uniforme continuou sendo a melhor escolha de método de randomização para esse caso, e também que a distribuição Logística mostrou os piores resultados dentre as distribuições apresentadas, algo um pouco surpreendente, se for levado em consideração o impacto positivo que a mesma apresenta na literatura.

Os gráficos de convergência e de diversidade dos algoritmos para cada uma das oito funções podem ser visualizados nas figuras 5.1 até 5.8. O cálculo da diversidade é realizado a partir de uma fórmula baseada no conceito de momento de inércia, e na utilização de centroides - tendo como referência o trabalho de (MORRISON; JONG, 2001).

Como pode-se observar, a convergência dos algoritmos para a maioria das funções, com exceção de algumas, foi extremamente rápida e abrupta. Sendo assim, para melhor visualização do comportamento de convergência, alguns dos gráficos estão sendo apresentados com um *zoom* no início das iterações. Essa convergência rápida se deu porque, em algumas funções, os algoritmos conseguiram alcançar o ótimo global

facilmente - como no caso da função Sphere, por exemplo. Em outros casos, como no da função Rosenbrock, os algoritmos convergiram de forma rápida para um ótimo local, onde ficaram presos pelo resto das gerações.

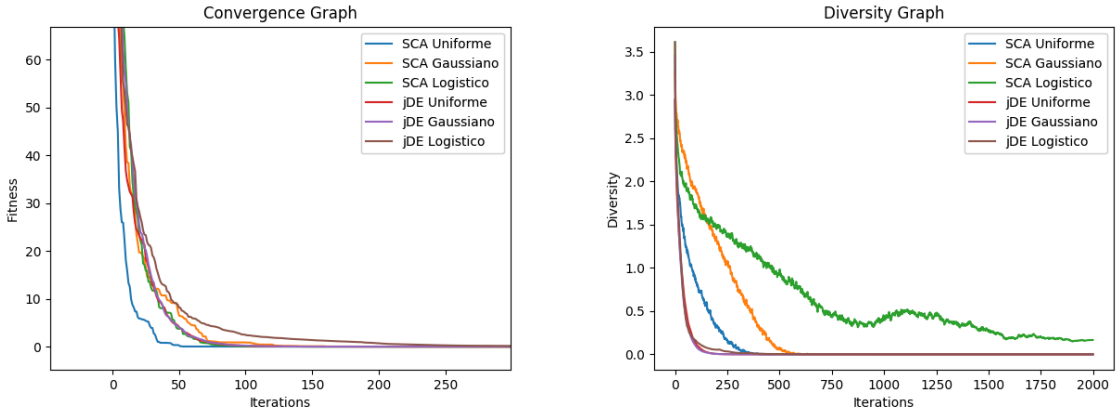

Figura 5.1: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Sphere

Fonte: Própria autora.

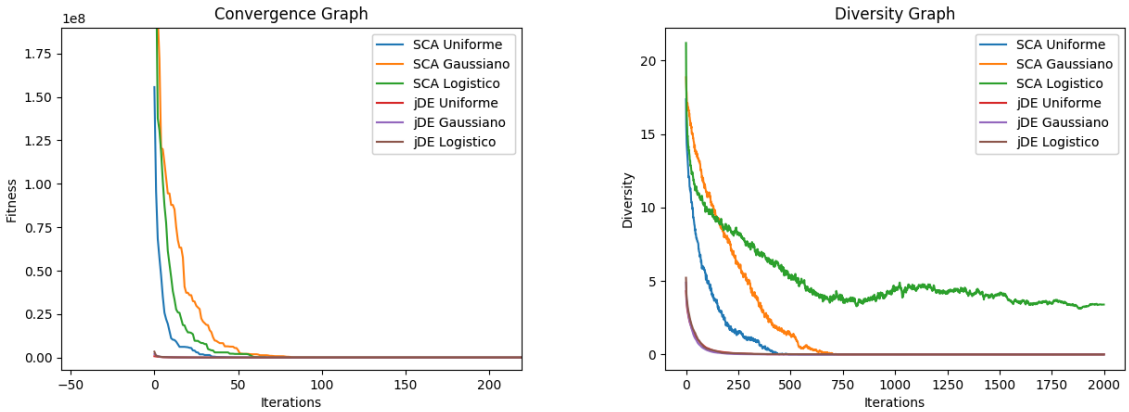

Figura 5.2: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Rosenbrock

Fonte: Própria autora.

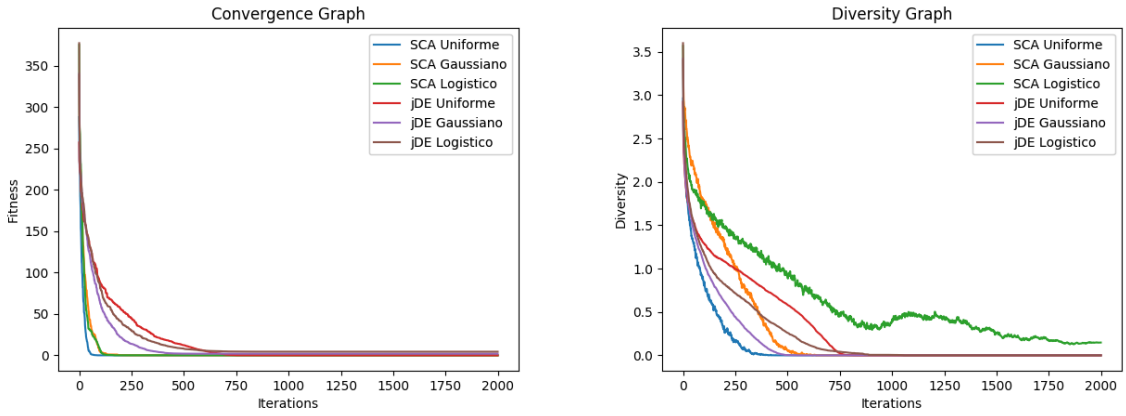

Figura 5.3: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Rastrigin

Fonte: Própria autora.

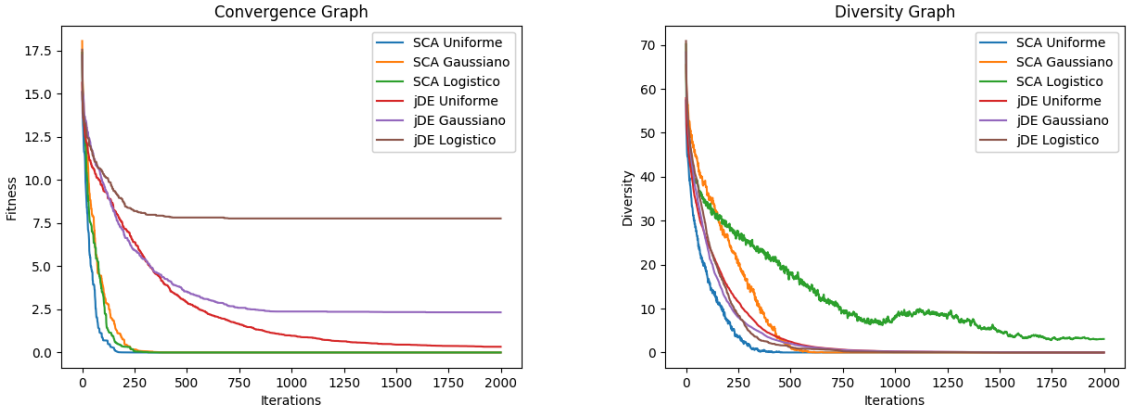

Figura 5.4: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Schaffer

Fonte: Própria autora.

Além disso, também pode-se perceber a partir dos gráficos, que a diversidade dos algoritmos na maioria dos casos se encontra baixa - após algumas iterações, a diversidade chega a zero. A única exceção é o algoritmo SCA Logístico, que apresenta um comportamento diferente das outras meta-heurísticas, mantendo a diversidade mais alta durante o processo e decrescendo a curva de forma mais lenta.

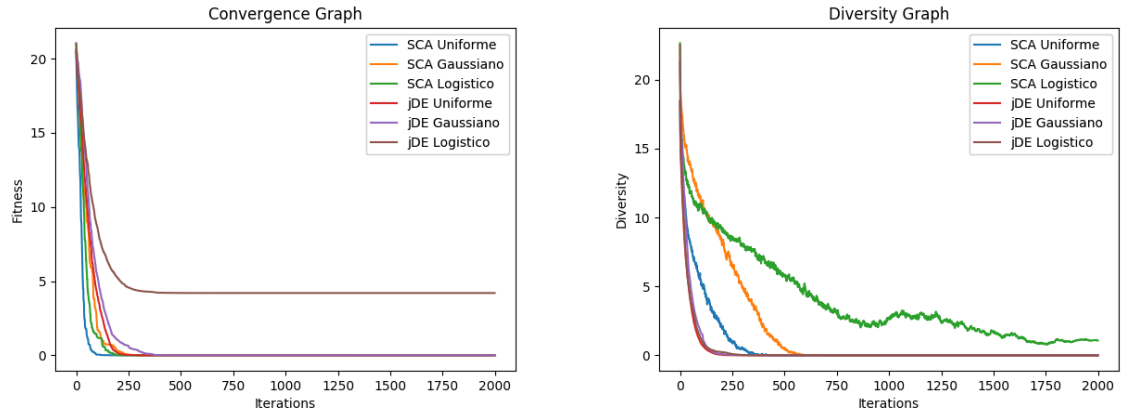

Figura 5.5: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Ackley

Fonte: Própria autora.

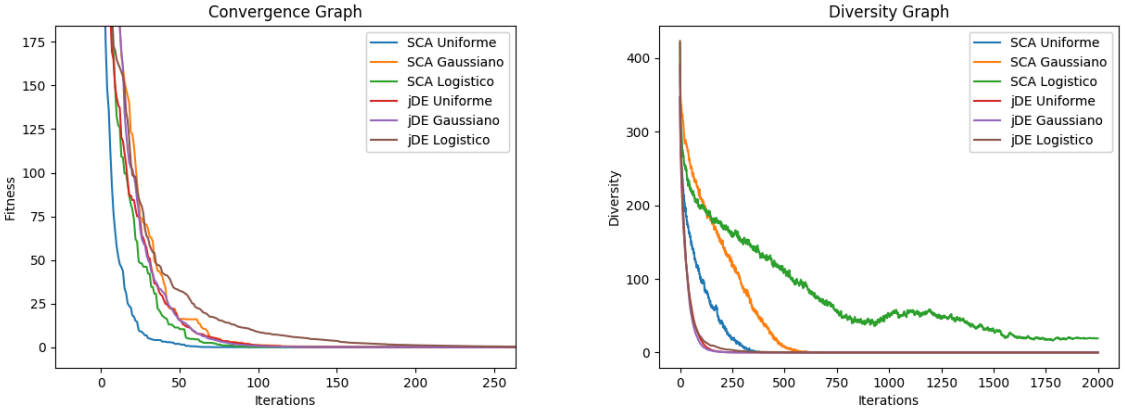

Figura 5.6: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Griewank

Fonte: Própria autora.

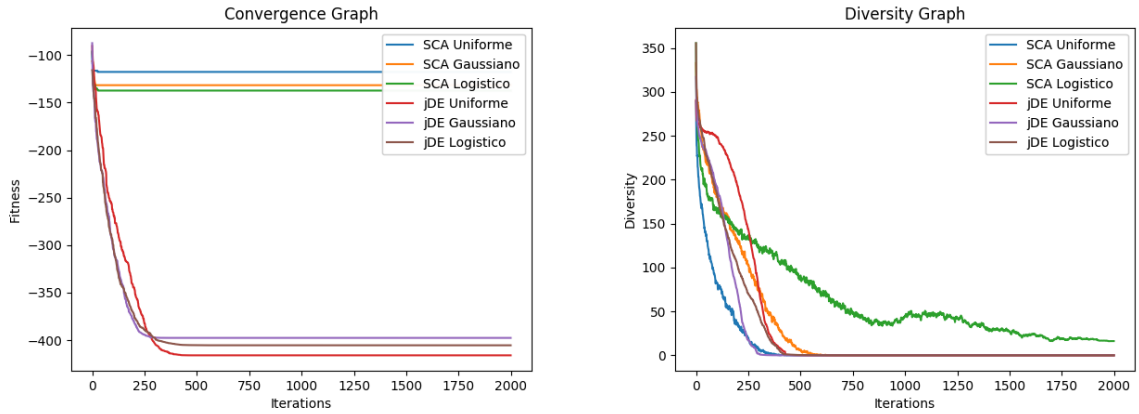

Figura 5.7: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Schwefel

Fonte: Própria autora.

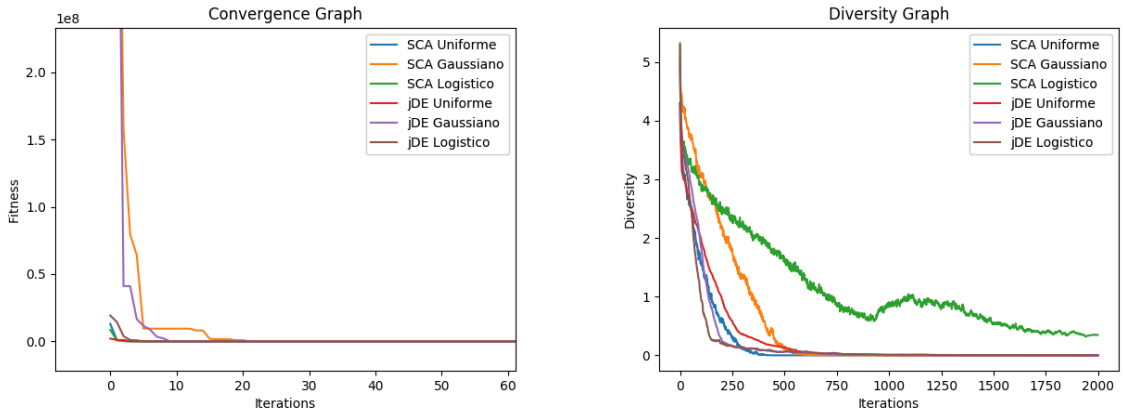

Figura 5.8: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Algoritmos para a Função Zakharov

Fonte: Própria autora.

5.2 Experimentos com PSP

Dois experimentos distintos foram executados aplicando algoritmos no problema da Predição da Estrutura de Proteínas. O primeiro experimento envolve utilizar as mesmas adaptações usadas nos testes com *benchmarks*, e possui o objetivo de verificar se o comportamento mostrado pelos algoritmos nas funções de teste se mantém o mesmo em um problema do mundo real, como o PSP. Já o segundo experimento envolve utilizar a aplicação descrita na seção 4.3, com o objetivo de verificar se a abordagem desenvolvida, usando uma aplicação em pontos específicos do algoritmo (parâmetros r2 e r3), com uma distribuição Gaussiana com desvio-padrão adaptado e uma técnica de *crowding*, possui um desempenho superior quando comparado com o algoritmo SCA original.

5.2.1 Configurações

As configurações utilizadas nos experimentos, foram as seguintes:

- Tamanho da população: 30;
- Gerações: 10000;
- Execuções: 30;

Além destas, para a segunda parte dos experimentos, tem-se os parâmetros da técnica de *crowding*: o fator de *crowding* foi definido como 3; e a porcentagem da população que é atualizada, ou seja, a quantidade de filhos ou *offspring* gerada é igual a 80% da população original.

5.2.2 Sequências de Proteínas

O conjunto de proteínas utilizadas envolvem quatro sequências *benchmark* de proteínas, utilizadas através do modelo *off-lattice* 2D-AB. As sequências, assim como o número de aminoácidos presentes nelas, ou seja, a quantidade de dimensões que a instância do problema possui, são apresentadas na tabela 5.3.

N	**Sequência**
13	ABBABBABABBAB
21	BABABBABABBABBABABBAB
34	ABBABBABABBABBABABBABABBABBABABBAB
55	BABABBABABBABBABABBABABBABBABABBABBABABBAB

Tabela 5.3: Conjunto de Sequências Proteicas Usadas

5.2.3 Resultados e Análises da Aplicação em Todos os Pontos

Os resultados do primeiro experimento com PSP, com aplicação em todos os pontos de aleatoriedade do algoritmo, podem ser observados na tabela 5.4, estruturada da mesma forma que a tabela de resultados anterior: o primeiro valor encontrado é a média das execuções e o segundo valor é o desvio-padrão. Cada linha da tabela representa uma sequência de proteínas, e cada coluna da tabela representa uma das distribuições utilizadas, sendo que para cada uma destas, tem-se a separação entre os dois algoritmos, jDE e SCA. A última linha da tabela representa o somatório de vezes que determinado algoritmo foi estatisticamente o melhor para certa função, sendo que os melhores resultados foram determinados a partir dos testes estatísticos de Kruskall-Vallis e de Dunn.

Função	Uniforme		Gaussiana		Logístico	
	jDE	SCA	jDE	SCA	jDE	SCA
F_{13}	**-1,9189e+00 ± 3,6629e-01**	-0,5874e+00 ± 0,0921e+00	**-1,7748e+00 ± 3,9848e-01**	-0,5902e+00 ± 0,0612e+00	-1,4509e+00 ± 7,0879e-01	-0,7968e+00 ± 0,1745e+00
F_{21}	**-4,0581e+00 ± 3,8556e-01**	-1,0345e+00 ± 0,1064e+00	**-3,5875e+00 ± 7,5970e-01**	-1,0738e+00 ± 0,0967e+00	-2,9871e+00 ± 1,2664e+00	-1,2132e+00 ± 0,1360e+00
F_{34}	**-6,6902e+00± 8,9389e-01**	-0,9526e+00 ± 0,2652e+00	**-5,7044e+00 ± 1,3987e+00**	-1,0135e+00 ± 0,2500e+00	-4,8845e+00 ± 1,8677e+00	-1,2386e+00 ± 0,2906e+00
F_{55}	**-9,6476e+00 ± 1,3193e+00**	-1,2564e+00 ± 0,1734e+00	2,5832e+01 ± 1,8098e+02	-1,2265e+00 ± 0,1684e+00	2,6790e+01 ± 1,8101e+02	-1,4580e+00 ± 0,3562e+00
Destaques	4	0	3	0	0	0

Tabela 5.4: Resultados do Primeiro Experimento com PSP

Através dos destaques, pode-se observar que o algoritmo jDE com distribuição Uniforme teve o melhor desempenho dentre todos, possuindo o melhor resultado em todas as quatro instâncias do problema. Em seguida, tem-se o jDE com distribuição Gaussiana, que mostrou resultados equivalentes ao do jDE Uniforme em três das instâncias. E por fim, todos os outros algoritmos apresentaram um desempenho ruim, não alcançando os melhores resultados em nenhuma das instâncias do problema. Além disso, pode-se perceber como as versões do algoritmo jDE, de modo geral, obtiveram resultados superiores as versões do algoritmo SCA, mostrando que entre os dois, o jDE se apresenta como a melhor escolha para um problema do mundo real complexo como o PSP.

Em relação às diferentes distribuições utilizadas, tem-se que para o algoritmo jDE, a distribuição Uniforme continuou sendo a melhor escolha, apresentando a maior consistência em seus resultados. A distribuição Gaussiana mostrou desempenho similar nas primeiras instâncias do problema, entretanto, na instância com 55 dimensões, uma das suas 30 execuções teve dificuldade para convergir, ficando presa em um ótimo local - isso fez com que seu resultado para essa sequência decaísse bastante. O mapa caótico Logístico teve o mesmo problema na instância de 55 dimensões, além de mostrar resultados piores em todas as outras instâncias, sendo a pior escolha de método de randomização para ser utilizado no algoritmo jDE com esse tipo de abordagem de adaptação. Essa conclusão foi

inesperada, visto que na literatura esse mapa caótico tem mostrado bons resultados em outros problemas e algoritmos.

Para o algoritmo SCA, tem-se que todas as suas versões apresentaram resultados bem próximos, entretanto, por mais que a diferença tenha sido pequena, os testes estatísticos mostraram que a versão que utiliza o mapa caótico Logístico apresentou resultados superiores. Sendo assim, a melhor escolha de método de randomização para o algoritmo SCA com a adaptação em todos os pontos de aleatoriedade, seria o mapa Logístico. Em sequência, pelo desempenho, tem-se a distribuição Gaussiana e por fim, a distribuição Uniforme. Essa conclusão foi exatamente o oposto do que foi visto no algoritmo jDE, e nos experimentos realizados anteriormente nas funções *benchmark*. Isso comprova que o tipo de método de randomização que deve ser utilizado, ou seja, que trará um melhor desempenho, está diretamente ligado ao algoritmo usado e ao problema na qual o mesmo será aplicado. Sendo assim, a escolha para método de randomização varia dependendo do problema, do algoritmo e também do tipo de abordagem utilizada, ou seja, em que pontos do algoritmo que esse método de randomização é aplicado.

Os gráficos de convergência e de diversidade dos algoritmos para cada uma das quatro sequências proteicas podem ser visualizados nas figuras 5.9 até 5.12.

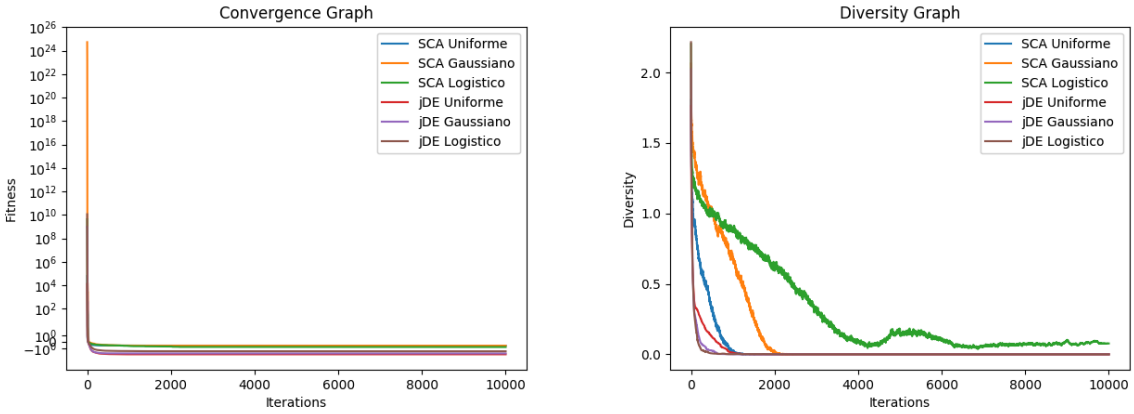

Figura 5.9: Gráfico de Convergência dos Algoritmos (esquerda) e Gráfico de Diversidade dos Algoritmos (direita) para a Sequência F13

Fonte: Própria autora.

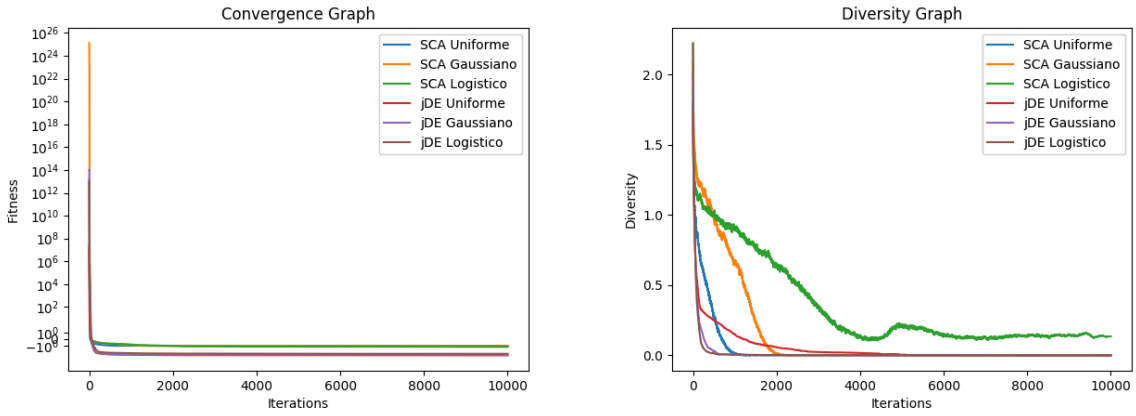

Figura 5.10: Gráfico de Convergência dos Algoritmos (esquerda) e Gráfico de Diversidade dos Algoritmos (direita) para a Sequência F21

Fonte: Própria autora.

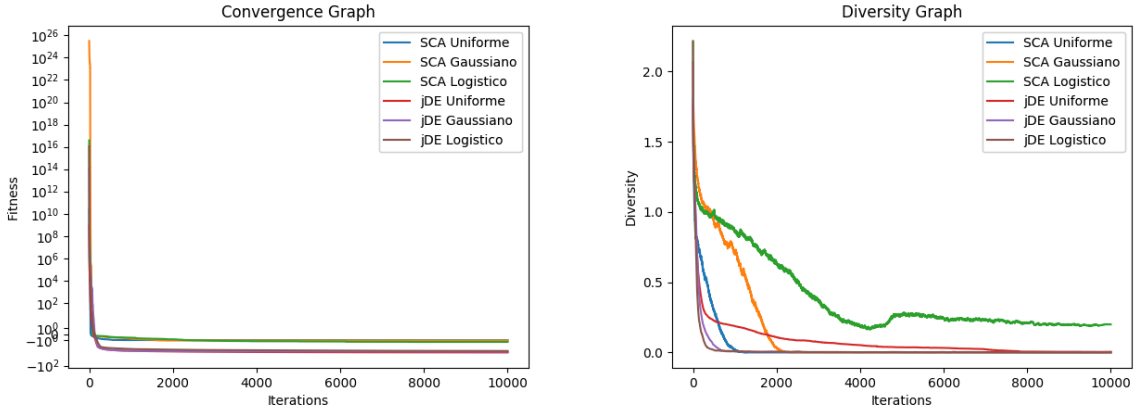

Figura 5.11: Gráfico de Convergência dos Algoritmos (esquerda) e Gráfico de Diversidade dos Algoritmos (direita) para a Sequência F34

Fonte: Própria autora.

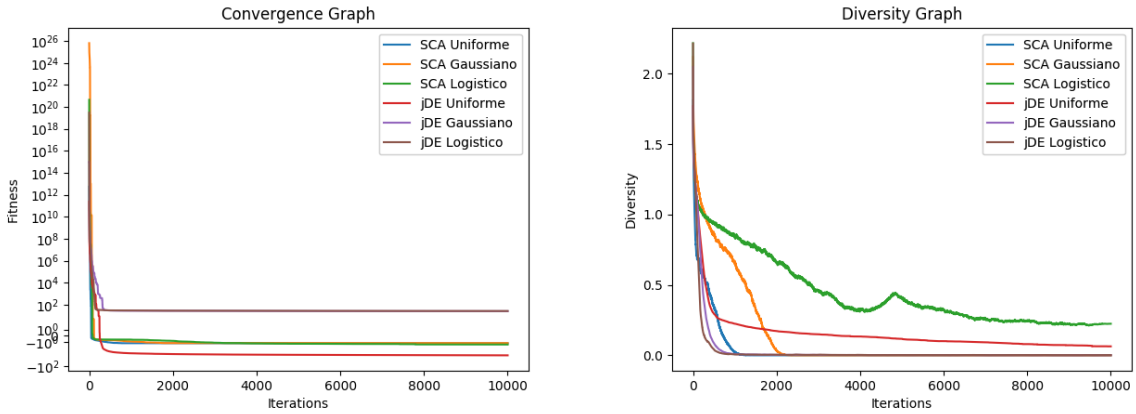

Figura 5.12: Gráfico de Convergência dos Algoritmos (esquerda) e Gráfico de Diversidade dos Algoritmos (direita) para a Sequência F55

Fonte: Própria autora.

Analisando os gráficos, pode-se observar que a convergência dos algoritmos foi extremamente rápida e abrupta para todas as instâncias do problema. Em média, nas primeiras 30 gerações do processo de otimização, os algoritmos já convergem para um ótimo, e aproveitam o restante das gerações para refinar as soluções a partir deste ótimo encontrado. Além disso, pode-se perceber que a diversidade dos algoritmos teve um comportamento similar para todas as instâncias do problema PSP e, com exceção do SCA com mapa Logístico, os algoritmos perderam a diversidade, chegando a zero, após cerca de 2000 a 4000 gerações.

5.2.4 Resultados e Análises da Aplicação em Pontos Específicos

Os resultados dos experimentos executados, a partir da abordagem apresentada em 4.3, podem ser observados na tabela 5.5, estruturada da mesma forma que as tabelas de resultados anteriores.

Função	SCA Padrão	SCA Adaptado
F_{13}	-0,5874e+00 ± 0,0921e+00	**-1,9444e+00 ± 0,3340e+00**
F_{21}	-1,0345e+00 ± 0,1064e+00	**-2,5988e+00 ± 0,4972e+00**
F_{34}	-0,9526e+00 ± 0,2652e+00	**-2,9768e+00 ± 0,6780e+00**
F_{55}	-1,2564e+00 ± 0,1734e+00	**-3,3634e+00 ± 1,0257e+00**
Destaques	0	4

Tabela 5.5: Resultados do Segundo Experimento com PSP

Através dos destaques, pode-se perceber que para todas as instâncias do problema PSP a versão adaptada do algoritmo SCA obteve um desempenho superior ao do algoritmo original. Isso nos mostra como a adaptação realizada foi eficaz e comprova que, assim como visto nos experimentos iniciais, em certos casos a utilização de um método de randomização diferente do Uniforme é benéfica para o algoritmo. Com isso, pode-se concluir que não só o método de randomização usado para gerar números aleatórios afeta diretamente os resultados e o desempenho dos algoritmos estocásticos, como também que deve ser feita uma escolha de qual método utilizar. Ou seja, o método de randomização acaba se tornando um outro parâmetro das meta-heurísticas, que além de ser extremamente importante, dever ser escolhido com cuidado, levando em consideração vários aspectos como: o algoritmo que está sendo usado; o problema ao qual o algoritmo está tentando solucionar; e em que partes do algoritmo este método foi aplicado.

Além disso, fica claro que a abordagem utilizada nessa adaptação, fazendo a troca do método de randomização em pontos específicos do algoritmo, trouxe benefícios, como a melhora dos resultados. Isso explica o porquê dessa abordagem ter sido a mais utilizada nos artigos encontrados no mapeamento sistemático da literatura. Entretanto, é válido apontar que essa abordagem tem suas desvantagens, como por exemplo a dificuldade de implementação. Para desenvolvê-la, é necessário compreender a fundo como o algoritmo em questão funciona e como é seu comportamento ao longo das gerações do processo de otimização. Além disso, é preciso identificar os pontos de interesse da meta-heurística, os parâmetros que mais afetam seu desempenho. Esse processo pode ser em muitos casos demorado e trabalhoso, enquanto a abordagem apresentada nos experimentos iniciais, trocando o método utilizado em todos os pontos de aleatoriedade do algoritmo, é muito mais simples.

Os gráficos de convergência e de diversidade dos algoritmos SCA original e SCA adaptado para cada uma das quatro sequências proteicas podem ser visualizados nas figuras 5.13 até 5.16.

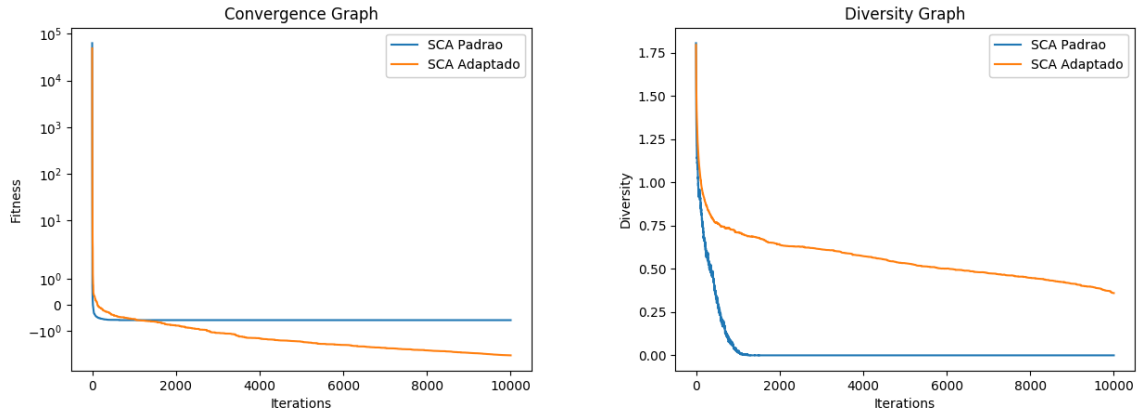

Figura 5.13: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Dois Algoritmos para a Sequência F13

Fonte: Própria autora.

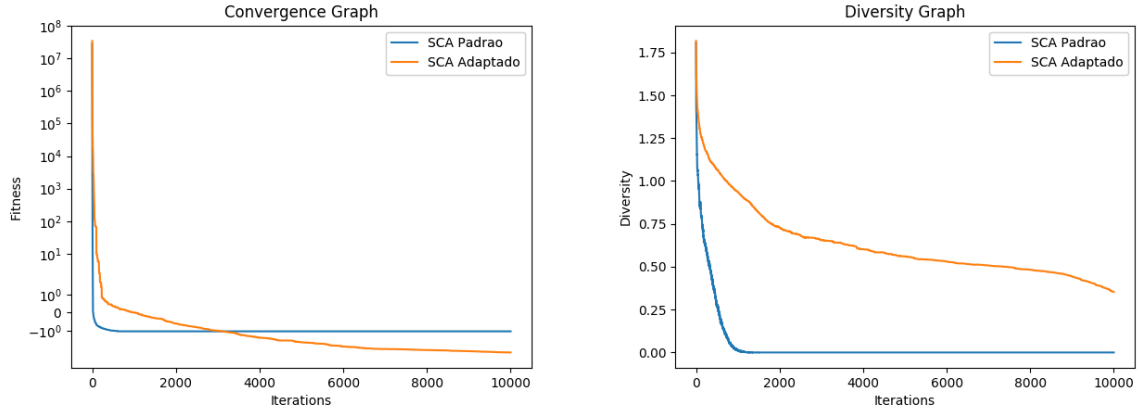

Figura 5.14: Gráfico de Convergência dos Algoritmos (esquerda) e Gráfico de Diversidade dos Algoritmos (direita) para a Sequência F21

Fonte: Própria autora.

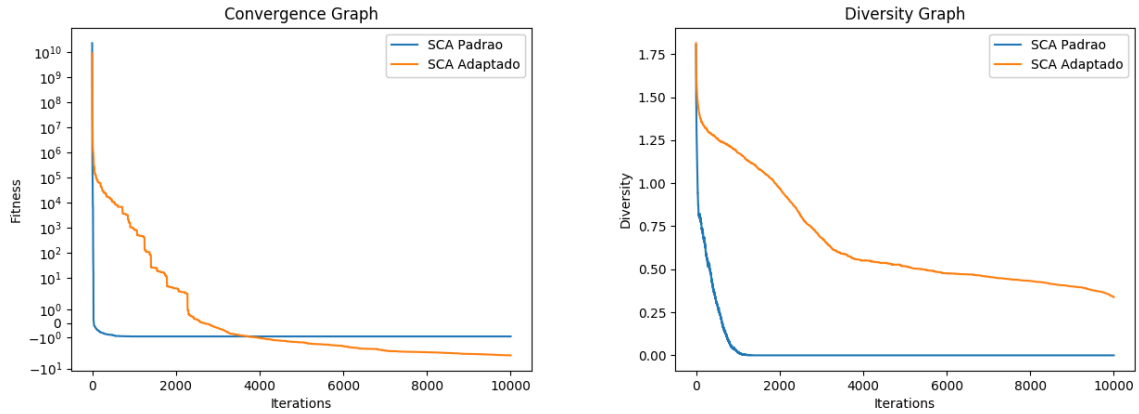

Figura 5.15: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Dois Algoritmos para a Sequência F34

Fonte: Própria autora.

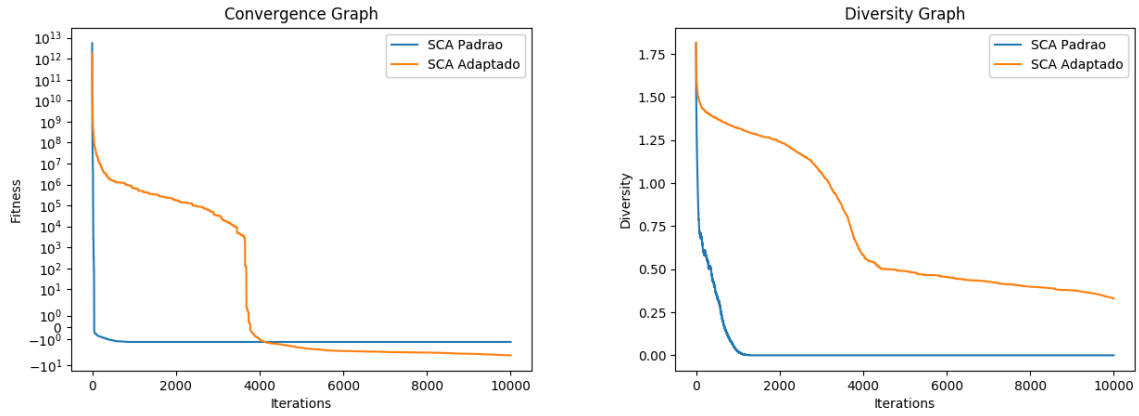

Figura 5.16: Gráfico de Convergência (esquerda) e Gráfico de Diversidade (direita) dos Dois Algoritmos para a Sequência F55

Fonte: Própria autora.

Analisando os gráficos, pode-se observar que a convergência do algoritmo SCA padrão foi rápida e abrupta para todas as instâncias do problema e, que em todos os casos, o algoritmo ficou preso em um ótimo local pela maioria das iterações. Já o SCA adaptado mostrou uma convergência rápida apenas nas duas primeiras funções, F13 e F21, enquanto que nas sequências F34 e F55, sua convergência foi mais lenta. Um ponto a se destacar é que o SCA adaptado não ficou preso permanentemente em ótimos locais - pode-se visualizar que sua curva de convergência continua a decrescer até o final das gerações, mesmo na sequência F55, onde pode-se observar que o algoritmo conseguiu escapar de um forte atrator local.

Além disso, pode-se perceber que a diversidade do algoritmo original chega a zero após algumas iterações, enquanto o SCA adaptado apresenta uma diversidade maior durante o processo, decrescendo de forma lenta. Esse comportamento é um resultado conjunto entre a utilização da distribuição Gaussiana com a técnica de *crowding*, responsável por desacelerar a perda de diversidade.

Outro ponto importante para se analisar é como o desempenho do SCA adaptado se compara com o desempenho de outros algoritmos da literatura para o problema PSP 2-D AB, além do SCA original. Para isso, apresenta-se para comparação três algoritmos: *Improved PSO Algorithm* (PSO) (ZHANG; LI, 2007); *Improved Energy Landscape Paving* (ELP+) (LIU et al., 2009); e *Conformational Space Annealing* (CSA) (KIM; LEE; LEE, 2005).

Função	PSO	ELP+	CSA	SCA Adaptado
F_{13}	-3,2941	-3,2941	-3,2941	-3,1766
F_{21}	-6,1977	-6,1980	-6,1980	-3,5372
F_{34}	-10,7036	-10,7453	-10,8060	-4,1649
F_{55}	-18,4236	-18,9301	-18,9110	-7,1168

Tabela 5.6: Resultados dos Algoritmos para Comparação

Como pode-se visualizar na tabela 5.6, tem-se as energias mínimas (melhores resultados de um conjunto de execuções) que foram encontradas pelos algoritmos para cada uma das sequências de proteínas. Pode-se perceber, através destes valores absolutos, que o algoritmo SCA adaptado possui um desempenho próximo dos outros apenas na primeira sequência de proteína, com 13 aminoácidos. Nos outros casos, seus resultados são inferiores em comparação com esses três algoritmos da literatura. Entretanto, para realizar uma comparação justa, deve-se levar em consideração o esforço computacional usado pelos diferentes algoritmos: os estudos que apresentaram o ELP+ e CSA não deixaram explícito o número de avaliações de função ou iterações usadas. Já de acordo com (ZHANG; LI, 2007), o PSO utilizou um total de 20 mil iterações, enquanto o SCA adaptado, algoritmo apresentado aqui, utilizou 10 mil iterações no total. Dessa forma, talvez o SCA adaptado apresentasse melhores resultados com mais iterações - mas essa hipótese não foi testada nos experimentos por questão de tempo de execução. O algoritmo, com os parâmetros citados nas configurações, já demorava um tempo considerável para realizar suas 30 execuções, visto que foi implementado de forma sequencial, e não paralela.

6 Considerações Finais

Este trabalho, em resumo, apresentou uma introdução para o problema de métodos de randomização em meta-heurísticas nas quais existem diversos métodos à disposição. Na maioria dos casos apenas a distribuição Uniforme é utilizada por ser o gerador de números aleatórios padrão de muitas linguagens. Também foi apresentada a fundamentação teórica, explicando os principais conceitos deste trabalho, como o que são distribuições probabilísticas, mapas caóticos, meta-heurísticas, intensificação e diversificação do espaço de busca, entre outros. Além disso, realizou-se o mapeamento sistemático da literatura, com a intenção de buscar uma visão geral da área de pesquisa em questão. Também foram apresentadas as propostas de adaptação para os algoritmos jDE e SCA e, por fim, foi realizado um conjunto de experimentos, com o objetivo de verificar algumas hipóteses levantadas sobre o tema, como verificar se a utilização de uma distribuição diferente da Uniforme pode ser benéfica para o desempenho dos algoritmos.

Os resultados obtidos com o mapeamento mostram como a área de pesquisa que engloba os métodos de randomização em meta-heurísticas têm crescido ao longo dos anos, e possui a tendência de crescer ainda mais. Além disso, foi possível perceber como certos algoritmos têm se destacado na área de otimização, sendo utilizados em vários estudos, como foi o caso do *Firefly Algorithm*. Outro ponto interessante foi a dominância dos mapas caóticos sendo utilizados como métodos de randomização; isso mostra como há uma grande popularidade em aplicar Teoria do Caos na área de otimização. Além disso, pôde-se observar também as diversas motivações apontadas pelos estudos para realizar essas adaptações dos algoritmos: muitos acreditam que a simples troca de uma distribuição por outra pode impactar imensamente nos resultados. Além de melhorar o desempenho dos algoritmos de forma geral, a troca de métodos pode evitar problemas como convergência lenta e prematura, além de estagnação em ótimos locais.

Com o mapeamento pôde-se compreender também as diversas maneiras como os métodos de randomização podem ser aplicados nas meta-heurísticas - desde pontos específicos nos algoritmos, até o mecanismo de ponte entre intensificação e diversificação. Com tudo isso, pôde-se concluir que os mapas caóticos se apresentam como uma boa escolha de método de randomização, visto sua imensa popularidade. Além disso, o local

de aplicação do método de randomização pode depender do algoritmo, entretanto, levando em consideração a importância da intensificação e diversificação para as meta-heurísticas, pôde-se concluir que a aplicação de diferentes métodos de randomização no balanço entre essas duas operações pode ser uma boa escolha.

Além disso, foi possível concluir, através dos experimentos realizados neste trabalho, que o método de randomização utilizado para geração de números aleatórios é um parâmetro das meta-heurísticas. Ao invés de apenas utilizar a distribuição Uniforme por padrão, o desenvolvedor pode escolher qual o melhor método a se utilizar, visto que nem sempre a distribuição Uniforme trará o melhor desempenho. Outra conclusão alcançada com os experimentos foi que a decisão de qual método se utilizar depende de diversos aspectos, como: o algoritmo que está sendo usado; o problema ao qual o algoritmo está tentando solucionar; e em que partes do algoritmo este método pode ser utilizado.

Neste trabalho, ainda foi possível verificar que para as funções *benchmark*, os melhores resultados obtidos com o algoritmo jDE foram utilizando a distribuição Uniforme e, com o algoritmo SCA, foram utilizando o mapa caótico Logístico. Já para o problema PSP, a distribuição Uniforme foi a melhor opção de método de randomização para o jDE, enquanto que para o SCA, a versão que apresentou os melhores resultados foi o SCA com a distribuição Gaussiana com desvio-padrão adaptado aplicado nos parâmetros r2 e r3 do algoritmo.

Como trabalhos futuros, sugere-se o desenvolvimento de um estudo com mesmo foco do trabalho em questão, aplicando diferentes métodos de randomização em meta-heurísticas, mas realizando experimentos com outros algoritmos (fora o jDE e o SCA, já estudados); outros métodos de randomização, como as distribuições de Cauchy, Rayleigh e mapas caóticos não explorados aqui, como o Piecewise, Sinusoidal, entre outros; outros pontos de aplicação, como somente na geração da população inicial, ou nos mecanismos de ponte entre intensificação e diversificação dos algoritmos.

Bibliografia

ABRO, A. G.; MOHAMAD-SALEH, J. Multiple-global-best guided artificial bee colony algorithm for induction motor parameter estimation. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING COMPUTER SCIENCES*, v. 22, p. 620–636, 01 2014.

*AHMED, S. et al. Feature selection using salp swarm algorithm with chaos. In: *Proceedings of the 2Nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*. New York, NY, USA: ACM, 2018. (ISMSI '18), p. 65–69. ISBN 978-1-4503-6412-6.

ALI, M. Synthesis of the β-distribution as an aid to stochastic global optimization. *Computational Statistics Data Analysis*, v. 52, n. 1, p. 133 – 149, 2007. ISSN 0167-9473.

ALLIGOOD, K. T.; SAUER, T. D.; YORKE, J. A. *Chaos: An Introduction to Dynamical Systems*. [S.l.]: Springer, 1996.

*ASKARZADEH, A.; COELHO, L. d. S. A backtracking search algorithm combined with burger's chaotic map for parameter estimation of pemfc electrochemical model. *International Journal of Hydrogen Energy*, v. 39, n. 21, p. 11165 – 11174, 2014. ISSN 0360-3199.

*AYALA, H. V. H. et al. Multiobjective symbiotic search algorithm approaches for electromagnetic optimization. *IEEE Transactions on Magnetics*, v. 53, n. 6, p. 1–4, June 2017. ISSN 0018-9464.

*AYALA, H. V. H. et al. Multiobjective wind driven optimization approach applied to transformer design. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.: s.n.], 2016. p. 4642–4647.

*BINGOL, H.; ALATAS, B. Chaotic league championship algorithms. *Arabian Journal for Science and Engineering*, v. 41, n. 12, p. 5123–5147, Dec 2016. ISSN 2191-4281.

*BOUSHAKI, S. I.; KAMEL, N.; BENDJEGHABA, O. A new quantum chaotic cuckoo search algorithm for data clustering. *Expert Systems with Applications*, v. 96, p. 358 – 372, 2017. ISSN 0957-4174.

BREST, J.; ZUMER, V.; MAUCEC, M. S. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: *2006 IEEE International Conference on Evolutionary Computation*. [S.l.: s.n.], 2006. p. 215–222. ISSN 1089-778X.

CAPONETTO, R. et al. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 3, p. 289–304, June 2003. ISSN 1089-778X.

CATTANI, M. et al. Deterministic chaos theory: Basic concepts. *Revista Brasileira de Ensino de Física*, Scielo, v. 39, 00 2017. ISSN 1806-1117.

*CHOU, J.-S.; NGO, N.-T. Modified firefly algorithm for multidimensional optimization in structural design problems. *Structural and Multidisciplinary Optimization*, v. 55, n. 6, p. 2013–2028, Jun 2017. ISSN 1615-1488.

*CHOU, J.-S.; NGOC-TRI, N.; PHAM, A.-D. Shear strength prediction in reinforced concrete deep beams using nature-inspired metaheuristic support vector regression. *Journal of Computing in Civil Engineering*, v. 30, p. 04015002, 01 2016.

*CHOU, J.-S.; PHAM, A.-D. Smart artificial firefly colony algorithm-based support vector regression for enhanced forecasting in civil engineering. *Computer-Aided Civil and Infrastructure Engineering*, v. 30, p. 715–732, 09 2015.

*CHOU, J.-S.; PHAM, A.-D. Nature-inspired metaheuristic optimization in least squares support vector regression for obtaining bridge scour information. *Information Sciences*, v. 399, p. 64 – 80, 2017. ISSN 0020-0255.

*CHOU, J.-S. et al. Evolutionary metaheuristic intelligence to simulate tensile loads in reinforcement for geosynthetic-reinforced soil structures. *Computers and Geotechnics*, v. 66, p. 1 – 15, 2015. ISSN 0266-352X.

*COELHO, L. et al. Electromagnetic optimization based on gaussian crow search approach. In: . [S.l.: s.n.], 2018. p. 1107–1112.

*COELHO, L. d. S.; BERNERT, D. L. d. A.; MARIANI, V. C. A chaotic firefly algorithm applied to reliability-redundancy optimization. In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. [S.l.: s.n.], 2011. p. 517–521. ISSN 1941-0026.

*COELHO, L. d. S. et al. A multiobjective firefly approach using beta probability distribution for electromagnetic optimization problems. *IEEE Transactions on Magnetics*, v. 49, n. 5, p. 2085–2088, May 2013. ISSN 0018-9464.

*COELHO, L. d. S. et al. Firefly approach optimized wavenets applied to multivariable identification of a thermal process. In: *Eurocon 2013*. [S.l.: s.n.], 2013. p. 2066–2071.

*DING, T. et al. A mixed-strategy-based whale optimization algorithm for parameter identification of hydraulic turbine governing systems with a delayed water hammer effect. *Energies*, v. 11, p. 2367, 09 2018.

*FARSHIN, A.; SHARIFIAN, S. A chaotic grey wolf controller allocator for software defined mobile network (sdmn) for 5th generation of cloud-based cellular systems (5g). *Computer Communications*, v. 108, p. 94 – 109, 2017. ISSN 0140-3664.

*FENG, Y. et al. Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with gaussian mutation. *Memetic Computing*, v. 10, n. 2, p. 135–150, Jun 2018. ISSN 1865-9292.

FISTER, I. et al. Analysis of randomisation methods in swarm intelligence. *International Journal of Bio-Inspired Computation*, v. 7, p. 36–49, 01 2015.

*GANDOMI, A. et al. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, v. 18, n. 1, p. 89 – 98, 2013. ISSN 1007-5704.

*GANDOMI, A. H.; YANG, X.-S. Chaotic bat algorithm. *Journal of Computational Science*, v. 5, n. 2, p. 224 – 232, 2014. ISSN 1877-7503. Empowering Science through Computing + BioInspired Computing.

*GHOLIZADEH, S.; ALIGHOLIZADEH, V. Reliability-based optimum seismic design of rc frames by a metamodel and metaheuristics. *The Structural Design of Tall and Special Buildings*, v. 28, p. e1552, 09 2019.

*GHOLIZADEH, S.; BAGHCHEVAN, A. Multi-objective seismic design optimization of steel frames by a chaotic meta-heuristic algorithm. *Engineering with Computers*, v. 33, n. 4, p. 1045–1060, Oct 2017. ISSN 1435-5663.

*ISMAIL, B. et al. Novel levy based particle swarm optimization algorithm for electrical power grid. In: . [S.l.: s.n.], 2013. p. 466–473. ISBN 978-0-7695-4979-8.

*JADDI, N. S.; ABDULLAH, S.; HAMDAN, A. R. Optimization of neural network model using modified bat-inspired algorithm. *Applied Soft Computing*, v. 37, p. 71 – 86, 2015. ISSN 1568-4946.

JAMIL, M.; YANG, X.-S. A literature survey of benchmark functions for global optimization problems. *Int. J. of Mathematical Modelling and Numerical Optimisation*, v. 4, 08 2013.

JANA, N.; SIL, J.; DAS, S. Continuous fitness landscape analysis using a chaos-based random walk algorithm. *Soft Computing*, 10 2018.

JANA, N. D.; DAS, S.; SIL, J. *A Metaheuristic Approach to Protein Structure Prediction: Algorithms and Insights from Fitness Landscape Analysis*. [S.l.]: Springer, 2018.

JANA, N. D.; SIL, J.; DAS, S. Selection of appropriate metaheuristic algorithms for protein structure prediction in ab off-lattice model: a perspective from fitness landscape analysis. *Information Sciences*, v. 391-392, p. 28 – 64, 2017. ISSN 0020-0255.

*KAVEH, A.; DADRAS, A.; MONTAZERAN, A. H. Chaotic enhanced colliding bodies algorithms for size optimization of truss structures. *Acta Mechanica*, v. 229, n. 7, p. 2883–2907, Jul 2018. ISSN 1619-6937.

KIM, S.-Y.; LEE, S.; LEE, J. Structure optimization by conformational space annealing in an off-lattice protein model. *Physical review. E, Statistical, nonlinear, and soft matter physics*, v. 72, 08 2005.

KUNDU, S. et al. Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization. *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 07 2013.

*LIANG, H. et al. A hybrid bat algorithm for economic dispatch with random wind power. *IEEE Transactions on Power Systems*, v. 33, n. 5, p. 5052–5061, Sep. 2018. ISSN 0885-8950.

*LIN, J.-H.; LI, Y.-L. A metaheuristic optimization algorithm for unsupervised robotic learning. In: *2012 IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom)*. [S.l.: s.n.], 2012. p. 113–117.

LIU, J. et al. Structure optimization of the two-dimensional off-lattice hydrophobic-hydrophilic model. *Journal of biological physics*, v. 35, p. 245–53, 09 2009.

MIRJALILI, S. Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, v. 96, 01 2016.

*MITIĆ, M. et al. Chaotic metaheuristic algorithms for learning and reproduction of robot motion trajectories. *Neural Computing and Applications*, v. 30, n. 4, p. 1065–1083, Aug 2018. ISSN 1433-3058.

*MITIĆ, M. et al. Chaotic fruit fly optimization algorithm. *Knowledge-Based Systems*, v. 89, p. 446 – 458, 2015. ISSN 0950-7051.

MONTGOMERY, D. C.; RUNGER, G. C. *Applied Statistics and Probability for Engineers*. [S.l.]: John Wiley & Sons, 2003.

MORRISON, R. W.; JONG, K. D. Measurement of population diversity. In: . [S.l.: s.n.], 2001. v. 2310, p. 31–41.

*MORTAZAVI, A.; KHAMSEH, A. A.; NADERI, B. A novel chaotic imperialist competitive algorithm for production and air transportation scheduling problems. *Neural Computing and Applications*, v. 26, n. 7, p. 1709–1723, Oct 2015. ISSN 1433-3058.

*OLIVEIRA, J. et al. Chaos-based grey wolf optimizer for higher order sliding mode position control of a robotic manipulator. *Nonlinear Dynamics*, v. 90, n. 2, p. 1353–1362, Oct 2017. ISSN 1573-269X.

PARPINELLI, R.; LOPES, H. An ecology-based evolutionary algorithm applied to the 2d-ab off-lattice protein structure prediction problem. In: . [S.l.: s.n.], 2013. p. 64–69.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, v. 64, p. 1 – 18, 2015. ISSN 0950-5849.

QU, C. et al. A modified sine-cosine algorithm based on neighborhood search and greedy levy mutation. *Computational Intelligence and Neuroscience*, v. 2018, p. 1–19, 07 2018.

REESE, A. Random number generators in genetic algorithms for unconstrained and constrained optimization. *Nonlinear Analysis: Theory, Methods Applications*, v. 71, n. 12, p. e679 – e692, 2009. ISSN 0362-546X.

ROSS, S. M. *Introduction to Probability Models*. [S.l.]: Elsevier Inc., 2010.

RUDIN, A.; CHOI, P. Chapter 1 - introductory concepts and definitions. In: RUDIN, A.; CHOI, P. (Ed.). *The Elements of Polymer Science Engineering (Third Edition)*. Third edition. Boston: Academic Press, 2013. p. 1 – 62. ISBN 978-0-12-382178-2.

SAHIB, M. A.; ABDULNABI, A. R.; MOHAMMED, M. A. Improving bacterial foraging algorithm using non-uniform elimination-dispersal probability distribution. *Alexandria Engineering Journal*, v. 57, n. 4, p. 3341 – 3349, 2018. ISSN 1110-0168.

*SAXENA, A.; KUMAR, R.; DAS, S. β-chaotic map enabled grey wolf optimizer. *Applied Soft Computing*, v. 75, p. 84 – 105, 2019. ISSN 1568-4946.

*SAXENA, A.; SHEKHAWAT, S.; KUMAR, R. Application and development of enhanced chaotic grasshopper optimization algorithms. *Modelling and Simulation in Engineering, Hindawi*, v. 2018, 04 2018.

*SAYED, G. I.; THARWAT, A.; HASSANIEN, A. E. Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. *Applied Intelligence*, v. 49, n. 1, p. 188–205, Jan 2019. ISSN 1573-7497.

SENKERIK, R. et al. Differential evolution and chaotic series. In: . [S.l.: s.n.], 2018. p. 1–5.

*SENKERIK, R. et al. On the population diversity for the chaotic differential evolution. In: . [S.l.: s.n.], 2018. p. 1–8.

*SURESH, S.; LAL, S. Multilevel thresholding based on chaotic darwinian particle swarm optimization for segmentation of satellite images. *Applied Soft Computing*, v. 55, p. 503 – 522, 2017. ISSN 1568-4946.

*TALATAHARI, S. et al. Imperialist competitive algorithm combined with chaos for global optimization. *Communications in Nonlinear Science and Numerical Simulation*, v. 17, n. 3, p. 1312 – 1319, 2012. ISSN 1007-5704.

TANABE, R.; FUKUNAGA, A. Success-history based parameter adaptation for differential evolution. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2013. p. 71–78. ISSN 1089-778X.

*TURGUT, O. E.; TURGUT, M. S.; COBAN, M. T. Design and economic investigation of shell and tube heat exchangers using improved intelligent tuned harmony search algorithm. *Ain Shams Engineering Journal*, v. 5, n. 4, p. 1215 – 1231, 2014. ISSN 2090-4479.

VESTERSTROM, J.; THOMSEN, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: *Proceedings of the 2004 Congress on Evolutionary Computation.* [S.l.: s.n.], 2004. v. 2, p. 1980–1987.

*WANG, G. G. et al. A novel cuckoo search with chaos theory and elitism scheme. In: *2014 International Conference on Soft Computing and Machine Intelligence.* [S.l.: s.n.], 2014. p. 64–69.

*WANG, G.-G. et al. Chaotic cuckoo search. *Soft Computing*, v. 20, n. 9, p. 3349–3362, Sep 2016.

*WOOD, D. A. Hybrid cuckoo search optimization algorithms applied to complex wellbore trajectories aided by dynamic, chaos-enhanced, fat-tailed distribution sampling and metaheuristic profiling. *Journal of Natural Gas Science and Engineering*, v. 34, p. 236 – 252, 2016. ISSN 1875-5100.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms.* [S.l.]: Luniver Press, 2010.

YANG, X.-S. Efficiency Analysis of Swarm Intelligence and Randomization Techniques. *Journal of Computational and Theoretical Nanoscience*, v. 9, 03 2013.

*YU, J. J. Q.; LAM, A. Y. S.; LI, V. O. K. Real-coded chemical reaction optimization with different perturbation functions. In: *2012 IEEE Congress on Evolutionary Computation.* [S.l.: s.n.], 2012. p. 1–8. ISSN 1089-778X.

ZHANG, J.; SANDERSON, A. C. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 5, p. 945–958, Oct 2009. ISSN 1089-778X.

ZHANG, X.; LI, T. Improved particle swarm optimization algorithm for 2d protein folding prediction. In: . [S.l.: s.n.], 2007. p. 53 – 56. ISBN 1-4244-1120-3.