Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Elan Gonçalves Costa

Otimização de estrutura de dendrímeros

Rio de Janeiro

2021

Elan Gonçalves Costa

Otimização de estrutura de dendrímeros

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes.

Orientador: Prof. Dr. Douglas Mota Dias

Rio de Janeiro

2021

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

C837 Costa, Elan Gonçalves.
Otimização de estrutura dendrímeros / Elan Gonçalves Costa. – 2021.
105f.

Orientador: Douglas Mota Dias.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Otimização estrutural - Teses. 3. Nanotecnologia - Teses. 4. Algorítmos - Teses. 5. Computação evolutiva - Teses. I. Dias, Douglas Mota. II. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. III. Título.

CDU 004.032.26

Bibliotecária: Júlia Vieira – CRB7/6022

Elan G Costa

Assinatura

10/06/2021

Data

Elan Gonçalves Costa

Otimização de estrutura de dendrímeros

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes.

Aprovado em: 11 de Maio de 2021

Banca Examinadora:

Prof. Dr. Douglas Mota Dias, D.Sc. (Orientador)

Faculdade de Engenharia - UERJ

Prof. Dr. Jorge Luís Machado do Amaral

Faculdade de engenharia - UERJ

Prof. Dr. Bruno Araujo Cautiero Horta

Instituto de Química - UFRJ

Profa. Dra. Marley Maria Bernardes Rebuzzi Vellasco

Departamento de Engenharia Elétrica – PUC-Rio

Rio de Janeiro

2021

AGRADECIMENTO

Agradeço primeiramente a Deus, por ter me dado força, dedicação e me permitido chegar a esse momento.

Ao meu orientador Douglas Mota Dias, pela confiança, dedicação e (muita) paciência durante a realização desse trabalho. Mesmo com a distância e as diferenças de fuso horário sempre que precisei esteve disponível para me auxiliar.

Aos professores Jorge Luís Machado do Amaral, Bruno Araujo Cautiero Horta e Marley Maria Bernardes Rebuzzi Vellasco, por aceitarem participar da banca de avaliação deste trabalho.

Ao professor José Franco Machado do Amaral, pelo estímulo a seguir na área acadêmica.

Aos meus pais, Egidio e Rosângela, por todo o suporte durante a realização dos meus trabalhos.

Aos meus irmãos, Elaine e Eduardo, pela compreensão das minhas ausências durante esse período.

Ao meu amigo Carlos e minha irmã Edlaine, pelas diversas dúvidas tiradas durante o desenvolvimento do texto deste trabalho.

A minha namorada Brenda, pelo apoio nos momentos difíceis e pelos muitos bons momentos.

Aos colegas Allan, George, José Rodrigo e Erito, que me auxiliaram durante as disciplinas.

Aos membros da equipe de desenvolvimento do PyPolyBuilder, pelo auxílio no desenvolvimento do projeto.

Aos membros do Laboratório de Redes Industriais e Sistemas de Automação - LARISA, por todo o apoio.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

A persistência é o caminho do êxito.

Charles Chaplin

RESUMO

COSTA, Elan Gonçalves. *Otimização de estrutura de dendrímeros.* 87 f. Dissertação (Mestrado em Engenharia Eletrônica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2021.

Neste trabalho, o foco dos estudos está nos dendrímeros conhecidos como PAMAM, cujas aplicações vão desde eluição de fármacos a encapsulação molecular e terapia gênica, e desde blocos de construção para nanoestruturas até micelas como agentes de descontaminação. Este trabalho desenvolve alguns métodos que substituem os utilizados originalmente no programa PyPolyBuilder, visando diminuir o tempo de execução e melhorar as moléculas geradas ao final do programa, tornando-as maiores e com um formato mais esférico. Para desenvolver os métodos, as simulações foram realizadas usando-se o PAMAM. Primeiramente, o método que gera a matriz-Z foi melhorado. Esse método era o gargalo computacional do programa e, por ser uma das primeiras etapas do mesmo, ocasionava grande dificuldade em se realizar testes nas etapas seguintes. Em sequência, foi desenvolvida uma nova função de aptidão para a otimização global. Ela possibilita um maior controle da forma do dendrímero que é gerado nessa etapa do programa. E, finalmente, com a intenção de melhorar os indivíduos gerados pela otimização global, alguns dos principais algoritmos de computação evolutiva foram testados e ajustados na geração zero do PAMAM, quais sejam: GA (*Genetic Algorithm*), DE (*Differential Evolution*), PSO (*Particle Swarm Optimization*) e CMA-ES (*Covariance Matrix Adaptation Evolution Strategy*). Cada um dos algoritmos tem sua própria variedade de métodos e parâmetros a serem ajustados. Para evitar o problema conhecido como "a maldição da dimensionalidade", uma metodologia aleatória foi utilizada nos ajustes, pois não demanda uma quantidade exageradamente grande de execuções e, ainda assim, garante a diversidade dos indivíduos no espaço de busca observado. Após os algoritmos terem seus parâmetros e métodos ajustados, os dez melhores são selecionados e simulados para as gerações um, dois, três, quatro, cinco, sete e dez do PAMAM, para então os melhores parâmetros e métodos serem definidos para cada algoritmo. Assim, os algoritmos são comparados entre si utilizando-se os melhores métodos e parâmetros escolhidos. Os algoritmos baseados no GA têm os piores resultados entre os observados. A CMA-ES apresenta os melhores re-

sultados para as gerações menores, mas apresenta erro de memória na geração dez. Com a CMA-ES sendo excluída da análise, a DE apresenta os melhores resultados para as gerações menores e o PSO apresenta os melhores para as gerações maiores do PAMAM.

Palavras-chave: Dendrímeros; PAMAM; Computação Evolutiva; GA; DE; PSO; CMA-ES.

ABSTRACT

In this work, the focus of the studies is on the dendrimers known as PAMAM, whose applications range from drug elution to molecular encapsulation and gene therapy, and from building blocks for nanostructures to micelles as decontamination agents. This work develops some methods that replace those originally used in the PyPolyBuilder program, aiming to decrease the execution time and improve the molecules generated at the end of the program, making them larger and with a more spherical shape. To develop the methods, the simulations were performed using PAMAM. First, the method that generates the Z-matrix has been improved. This method was the computational bottleneck of the program and because it was one of the early stages of the program, it caused great difficulty in carrying out tests in the following stages. Following, a new fitness function was developed for global optimization. It allows greater control of the shape of the dendrimer that is generated at this stage of the program. And finally, with the intention of improving the individuals generated by global optimization, some of the main evolutionary computation algorithms were tested and adjusted in the PAMAM zero generation, namely: Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Each of the algorithms has its own variety of methods and parameters to be adjusted. To avoid the problem known as " the curse of dimensionality ", a random methodology was used in the adjustments, as it does not demand an unreasonably large number of executions and, even so, it guarantees the diversity of individuals in the observed search space. After the algorithms have their parameters and methods adjusted, the top ten are selected and simulated for generations one, two, three, four, five, seven, and ten of PAMAM, so that the best parameters and methods are defined for each algorithm. Thus, the algorithms are compared with each other using the best methods and parameters chosen. GA-based algorithms have the worst results among those observed. The CMA-ES presents the best results for the smaller generations, but it presents memory error in the 10th generation. With the CMA-ES being excluded from the analysis, the DE presents the best results for the younger generations and the PSO presents the best results for the older generations of PAMAM.

Keywords: Dendrimers; PAMAM; Evolutionary Computing; GA; DE; PSO; CMA-ES.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

PAMAM	Dendrímeros de Poli Amidoamina
BB	Building Blocks
GA	Genetic Algorithm
MD	Molecular Dynamics
MC	Monte Carlo
DEAP	Distributed Evolutionary Algorithms in Python
PPI	Dendrímeros de Polipropileno Imina
EC	Evolutionary Computation
GP	Genetic Programming
DE	Differential Evolution
ES	Evolution Strategy
EP	Evolutionary Programming
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
PSO	Particle Swarm Optimization

SUMÁRIO

INTRODUÇÃO

Polímeros dendríticos (cascata), ou de arquitetura dendrítica, são uma das topologias mais difundidas, sendo observadas na natureza nas escalas de comprimento macro e microdimensional. Porém, quando se tratam de nanoescalas, essa arquitetura é pouco recorrente em estados naturais. Uma ampla gama de estruturas dendríticas foram relatadas utilizando-se numerosas estratégias sintéticas. Essa classe arquitetônica se divide em quatro subclasses dendríticas: polímeros hiper-ramificados aleatórios, polímeros dendrigráficos, dendrons e dendrímeros [1].

Todos os parâmetros críticos do projeto molecular de um dendrímero, como tamanho, forma, química de superfície e interior, flexibilidade e topologia, são controlados de maneira quase completa durante seu processo sintético de desenvolvimento [2]. Os dendrímeros têm diversas aplicações. Entre elas estão o aumento da solubilização, dissolução e biodisponibilidade de fármacos [3], considerações farmacocinéticas no projeto de nanomedicina [4], liberação direcionada de medicamentos [5], administração de fármacos anticâncer [6], quimioterapia direcionada ao câncer [7], projeto de pró-fármacos [8], entrega de genes [9], imunoterapia e terapia hormonal [10], diagnósticos médicos [11], além de aplicações diversas em cosméticos, veterinária e agricultura [12]

As principais técnicas de simulação de propriedades químicas são a de dinâmica molecular (MD, do inglês, *Molecular Dynamics*) e a de Monte Carlo (MC) [13]. Na simulação MC, o principal parâmetro observado é a energia da molécula. Já na simulação MD, o principal parâmetro é a força instantânea atuando no monômero [14]. As simulações utilizando o PyPolyBuilder têm como principal parâmetro para a otimização global as distâncias intramoleculares e, na otimização local, as forças intramoleculares. O PyPolyBuilder [15] utiliza para otimização global um algoritmo genético (GA, do inglês, *Genetic Algorithm*) [16]. Esse GA foi desenvolvido especificamente para o programa, sem a utilização de uma biblioteca pré-definida. Posteriormente é realizada uma otimização local baseada na descida do gradiente. O PyPolyBuilder será apresentado em detalhes mais à frente.

MD é uma ferramenta computacional poderosa usada em Química Medicinal para o planejamento racional de fármacos. MD é uma extensão da Mecânica Molecular, onde o comportamento dinâmico de um sistema molecular é simulado através da integração

numérica das equações de movimento. Esta técnica tem sido usada extensivamente para auxiliar e complementar o planejamento de novos ligantes de um alvo terapêutico, bem como estimar a sua potência [17, 18].

Durante uma simulação de MD são necessários os parâmetros de entrada, a arquitetura de topologia e a arquitetura de coordenadas (que determina a geometria da molécula). Os cálculos utilizados nos cálculos das arquiteturas são muito complexos, como apresentado em [17, 18]. O PyPolyBuilder simplifica esses cálculos, gerando esses arquivos que são utilizados na simulação. Por esse motivo, neste trabalho esse programa é utilizado e terá alguns de seus métodos melhorados. Na Figura 1 são apresentadas as entradas e saídas necessárias para a execução da simulação de MD. Nela são mostradas as entradas que são geradas pelo PyPolyBuilder.

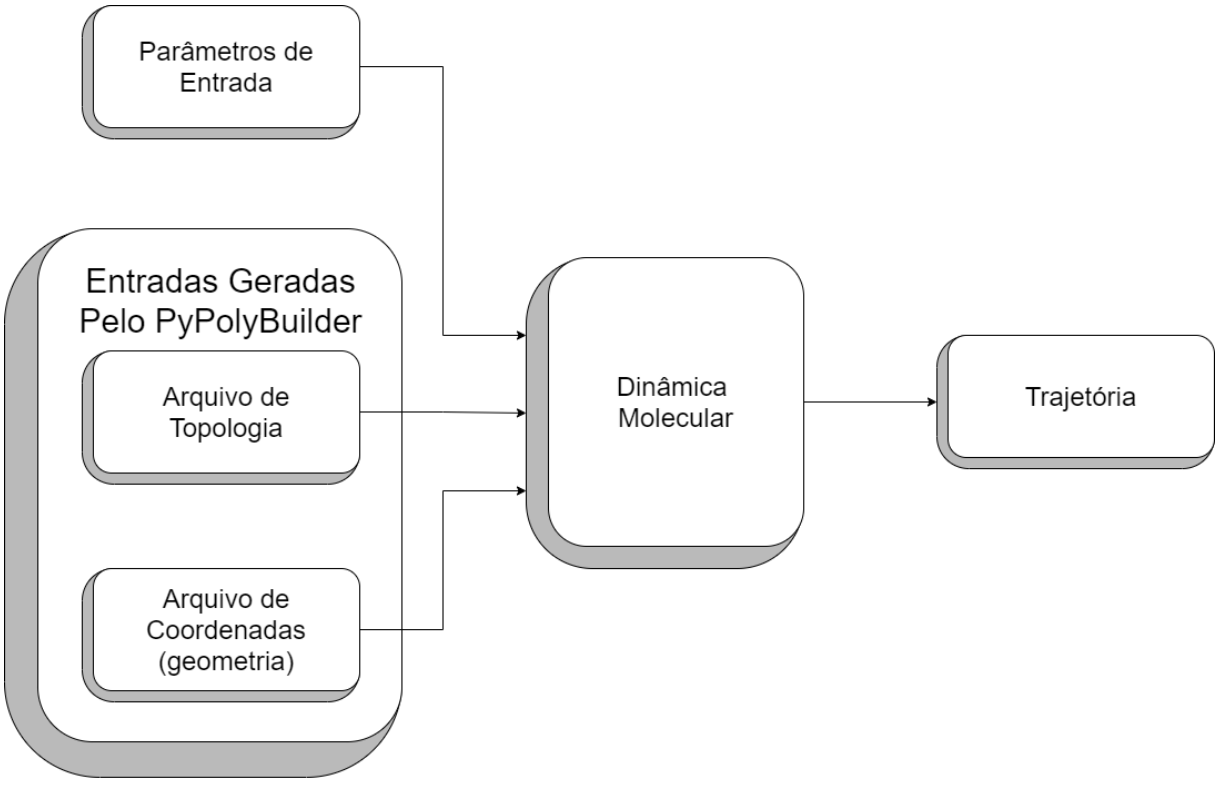

Figura 1 - Entradas e saídas, utilizadas e geradas pela dinâmica molecular, frisando as entradas que podem ser geradas pelo PyPolyBuilder.

Neste trabalho, o foco dos estudos são os dendrímeros de poli amidoamina (PAMAM) que têm aplicações que vão desde eluição de fármacos a encapsulação molecular e terapia gênica, e desde blocos de construção para nanoestruturas até micelas como agentes de descontaminação [19]. É necessário um estudo prévio abrangente de sua estrutura

e dinâmica, assim como sua forma e tamanho em função das gerações, distribuição de monômeros, distribuição de terminais e área de superfície acessível ao solvente para que os dendrímeros possam ser utilizados de maneira ótima em aplicações tão diversas [2].

Desde os primeiros estudos de modelagem, vários projetos de simulação teórica e computacional têm sido realizados sobre as propriedades estruturais do dendrímero quando submetido a diversas condições [2]. Muitos deles se concentram em modelos de dendrímeros simples para obter características qualitativas ou simulações de nível molecular para gerações inferiores. Tais estudos são capazes de gerar um maior conhecimento sobre algumas estruturas e dinâmicas do dendrímero. Porém, quando se trata de distribuição dos grupos terminais e sua localização na periferia da molécula, ainda existem controvérsias. Além disso, nem o nível atômico, nem as simulações do modelo de contas ainda investigaram a questão da geração limitante, que foi primeiramente levantada por de Gennes [2].

O PyPolyBuilder [15] é uma ferramenta implementada utilizando a linguagem de programação Python, que simula virtualmente a geração de polímeros lineares e ramificados, macromoléculas e estruturas supramoleculares formados por blocos construtores (BB, do inglês, *Building Blocks*). O programa disponibiliza duas opções: a preparação de polímeros radiais, como os dendrímeros e, no segundo modo, conecta os BBs com base em um arquivo de conectividade. Os BBs são escritos de acordo com a definição de uma topologia molecular no formato Gromacs (itp) ou GROMOS (mtb) [20]. O programa gera uma matriz-Z, cujas colunas descrevem as distâncias entre os átomos. Os valores são obtidos em relação às posições cartesianas dos átomos [21]. Uma vez que a matriz é gerada, o programa realiza uma otimização global para identificar a vizinhança onde se encontra a melhor topologia para a molécula. Após a otimização global, é realizada uma otimização local para obter o melhor indivíduo na vizinhança indicada na etapa anterior. No PyPolyBuilder [15] a criação da matriz-Z se torna o gargalo, pois a cada geração ocorre um aumento muito grande na demanda computacional.

A primeira contribuição desta dissertação é otimizar o método de criação da matriz-Z na versão original do PyPolyBuilder. O gargalo computacional na geração da matriz ocorria porque o algoritmo executava quatro laços, cada um interno ao anterior, assim, varrendo todas as possíveis combinações de quatro átomos da molécula para verificar suas ligações. O método proposto nessa dissertação e implementado no programa aproveita as

particularidades dos BBs utilizados para gerar a matriz-Z de forma mais eficiente.

Neste trabalho, uma vez que a geração da matriz-Z foi otimizada, são utilizados métodos estocásticos de otimização de maneira a se obter as melhores características para o dendrímero. O primeiro método de otimização proposto é um GA mais formal utilizando a biblioteca DEAP (do inglês, *Distributed Evolutionary Algorithms in Python*) [22] em Python. Além desse método, é proposta a utilização de outros algoritmos estocásticos: CMA-ES (do inglês, *Covariance Matrix Adaptation Evolution Strategy*), DE (do inglês, *Differential Evolution*) e PSO (do inglês, *Particle Swarm Optimization*). Os algoritmos foram selecionados por servirem como base para diversos dos métodos mais eficientes, quando se trata de otimização no domínio dos Reais, como pode ser observado nos resultados da competição *2015 IEEE CEC Competition on Niching Methods for Multimodal Optimization* [23]. Paralelamente a esse estudo, é proposta uma mudança na função de aptidão utilizada nos algoritmos para uma função que aumente a esfericidade da molécula ao invés da distância intramolecular.

Este documento está estruturado da seguinte forma: No Capítulo 1 são apresentadas as principais características dos dendrímeros e suas aplicações. Este Capítulo é dividido em dois tópicos: Conceitos Básicos e PAMAM. No primeiro são apresentadas as características gerais dos dendrímeros. No segundo, são apresentadas as características específicas do PAMAM, que é o dendrímero foco dos estudos neste trabalho. No Capítulo 2 são apresentadas as principais características dos algoritmos de computação evolutiva. Este Capítulo é dividido em cinco tópicos: Conceitos Básicos, GA, DE, PSO e CMA-ES. O primeiro apresenta as características básicas da computação evolutiva, enquanto os demais apresentam mais detalhadamente as características dos algoritmos que dão nome a esses tópicos. No Capítulo 3 é apresentada a metodologia utilizada neste trabalho. Este Capítulo é dividido em dois tópicos: PyPolyBuilder e Ajustes de parâmetros dos algoritmos de otimização. No primeiro são apresentadas as características do PyPolyBuilder que é o programa utilizado para gerar computacionalmente as moléculas. Já no segundo, são apresentados os métodos utilizados para escolher as melhores características dos algoritmos para o problema abordado. Por sua vez, no Capítulo 4 são apresentados os resultados obtidos neste trabalho. Este Capítulo é dividido em dois tópicos: Resultados Matriz-Z e Resultados da Otimização Global. No tópico Resultados Matriz-Z são apresentados os resultados que foram obtidos na otimização da matriz-Z. No tópico Resultados da Oti-

mização Global são apresentados os resultados obtidos por cada algoritmo e os resultados dos algoritmos são comparados para averiguar suas se vantagens e desvantagens. Nas conclusões os resultados obtidos no Capítulo 4 são analisados.

1 DENDRÍMEROS

As primeiras citações a dendrímeros foram realizadas em 1978, como moléculas hiper-ramificadas, sendo chamados de moléculas em cascata [24]. Essas moléculas hiper-ramificadas foram a base dos dendrímeros de polipropileno imina (PPI), desenvolvidos pelos grupos de Meijer e Mülhaupt no início dos anos 1990 [25] [26].

Em 1985, Tomalia et. al. exibem sua discussão sobre o conceito, definição, síntese e caracterização dos "polímeros *starburst*" e passaram a utilizar o termo dendrímeros para representá-los. Além disso, suas gerações de um a sete foram sintetizadas [27]. Nesse trabalho, pela primeira vez, o PAMAM foi apresentado juntamente com um grande número de suas aplicações [14]. Posteriormente foi realizada a síntese até a décima primeira geração por Maiti et. al., em 2004 [2].

Dendrímeros utilizam uma sequência de reações repetitivas em sua síntese. Esse procedimento garante que o dendrímero seja um polímero altamente monodisperso [1]. Dendrímeros, em contraste com os polímeros tradicionais, são uma topologia que se ramifica de maneira quase perfeita irradiando a partir de um núcleo (*core*). Essas vantagens incluem entrada celular rápida, absorção reduzida de macrófagos, capacidade de segmentação e passagem mais fácil através de barreiras biológicas por transcitose. Em comparação com os polímeros lineares, os dendrímeros são multivalentes devido à presença de altas multiplicidades de grupos terminais de superfície reativa, tornando-os portadores de drogas ideais com capacidades de carga útil de droga mais elevadas [28]. Dendrímeros podem ser caracterizados pelo número de gerações, pela quantidade de ramos partindo do núcleo e do comprimento entre os pontos de ramificação [29]. A Figura 2 mostra um dendrímero com duas gerações, quatro braços por ramo e comprimento entre os pontos de ramificação igual a dois.

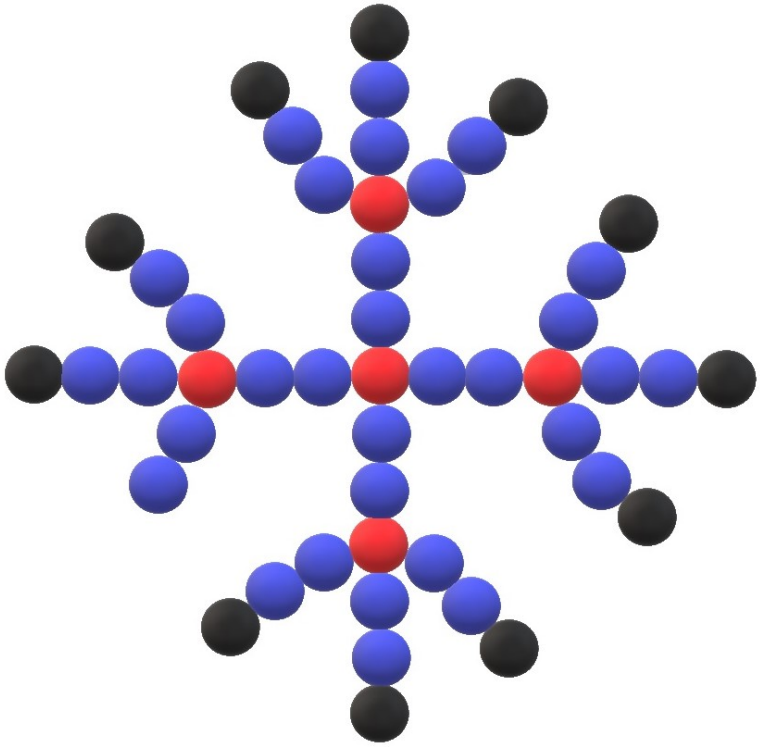

Figura 2 - Dendrímero com duas gerações, quatro ramos e comprimento entre os pontos de ramificação igual a dois. Os braços estão em azul, as ramificações em vermelho e os finais estão em preto.

1.1 **Conceitos Básicos**

Um dendrímero possui três estruturas básicas: um *core*, que pode ser composto de um único átomo ou um grupo atômico com pelo menos duas funções químicas idênticas; uma camada interior (que é referida nesse trabalho como *inter*), sendo que cada repetição do *inter* vai representar uma geração do dendrímero; e um grupo funcional terminal (que é referido neste trabalho como *term*), que geralmente é localizado na parte exterior da macromolécula e possui um papel fundamental em suas propriedades [1]. Um dendrímero carbosilano de segunda geração é apresentado, como exemplo, na Figura 3, onde dentro do círculo preto temos o *core*, dentro do círculo azul temos a primeira geração, dentro do verde a segunda geração e finalmente dentro do vermelho os terminais.

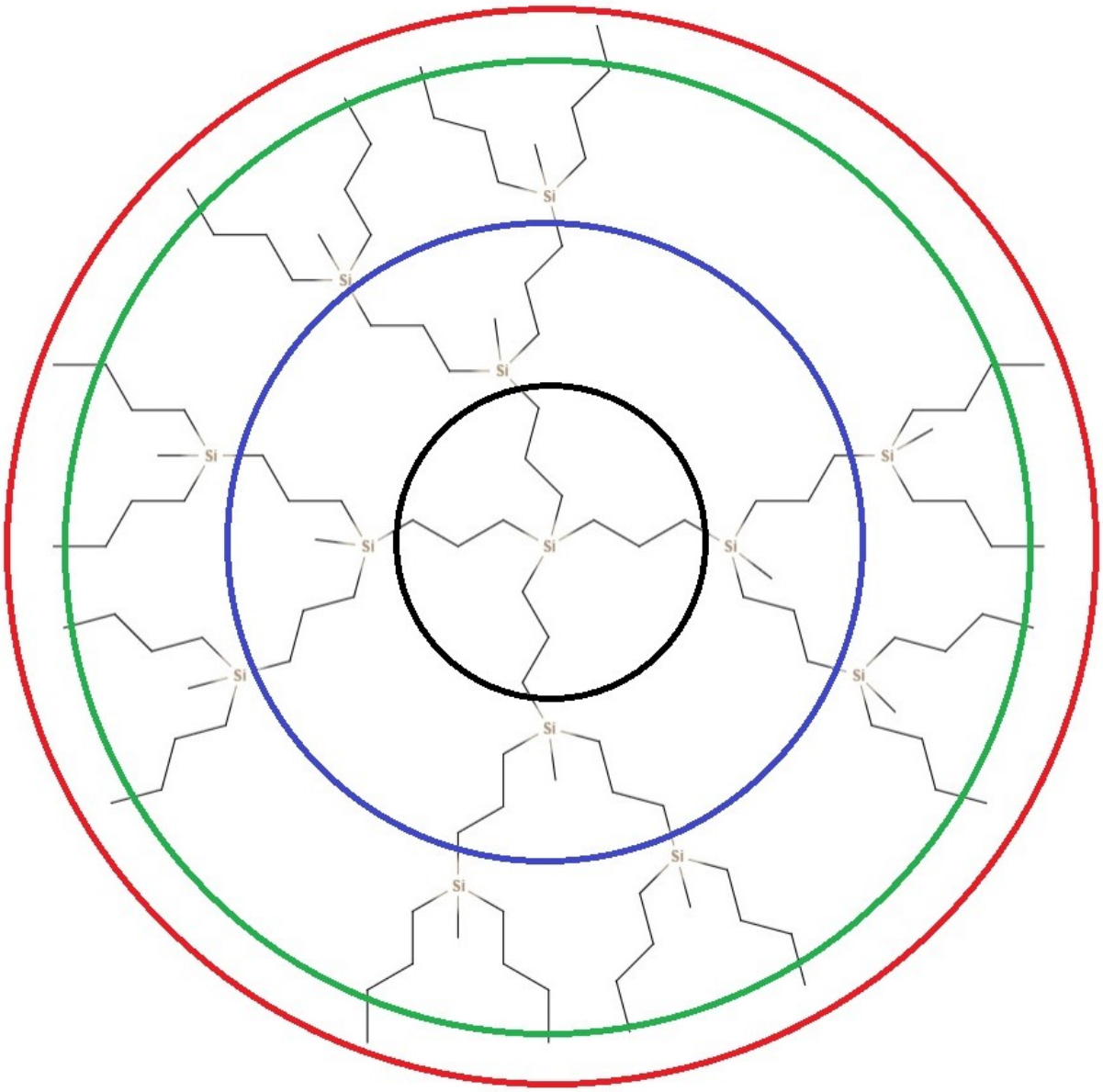

Figura 3 - Dendrímero carbosilano: dentro do círculo preto temos o core, dentro do círculo azul temos a primeira geração, dentro do verde a segunda geração e finalmente dentro do vermelho os terminais.

Existem diversas abordagens para sintetizar um dendrímero, onde as principais abordagens são os métodos convergente [30] e divergente [27]. No método divergente, o *core*, que é a primeira parte de um dendrímero, engloba várias espécies químicas [1]. Em seguida, o dendrímero possui várias gerações (camadas). A primeira geração é composta por diversos BBs do tipo *intern*, cada um deles conectado aos átomos com com elétrons livres do *core*; As gerações subsequentes são conectadas aos átomos com elétrons livres das

camadas anteriores [2]. Essas camadas geram um espaço flexível, criando vazios dentro dos BBs dendríticos repetitivos onde diversas moléculas menores podem ser encapsuladas [1].

A última camada forma uma superfície multivalente capaz de interagir com o ambiente externo, definindo assim, as características macroscópicas do dendrímero [1]. O *term* pode ser alterado pela adição de pequenas moléculas, incluindo íons, drogas e biomoléculas. Este processo pode alterar drasticamente os aspectos físico-químicos, reativos, dinâmicos e biológicos [31]. Essa camada é gerada ligando-se os BBs *term* aos átomos com elétrons livres na última geração da molécula [2]. Na Figura 4 são apresentados os BBs de um dendrímero carbosilano, onde se vê (a) o *core*, (b) um *inter* e (c) um *term*.

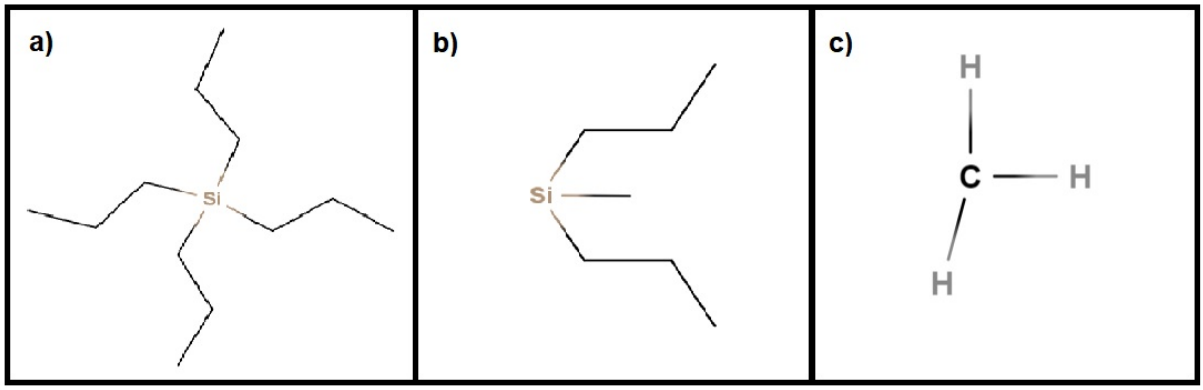

Figura 4 - BBs de um dendrímero carbosilano, quais sejam: (a) *core*, (b) *inter* e (c) *term*.

1.2 **PAMAM**

Uma metodologia divergente baseada em monômeros de acrilato foi desenvolvida nos laboratórios Dow durante o período de 1979 a 1985. Essa metodologia não sofreu com os baixos rendimentos, pureza ou purificação encontrados por Voegtle [32] em sua síntese em cascata e proporcionou a primeira família de dendrímeros bem caracterizados. Os dendrímeros PAMAM com pesos moleculares, que variam de várias centenas a mais de 1 milhão de daltons, foram preparados com altos rendimentos. Esta metodologia original teve tanto sucesso que hoje ainda constitui a rota comercial preferida para a família de dendrímeros Starburst®, de marca registrada [32]. Na Figura 5 é apresentado o PAMAM das gerações de zero a três, já na Figura 6 são apresentados os BBs de um PAMAM.

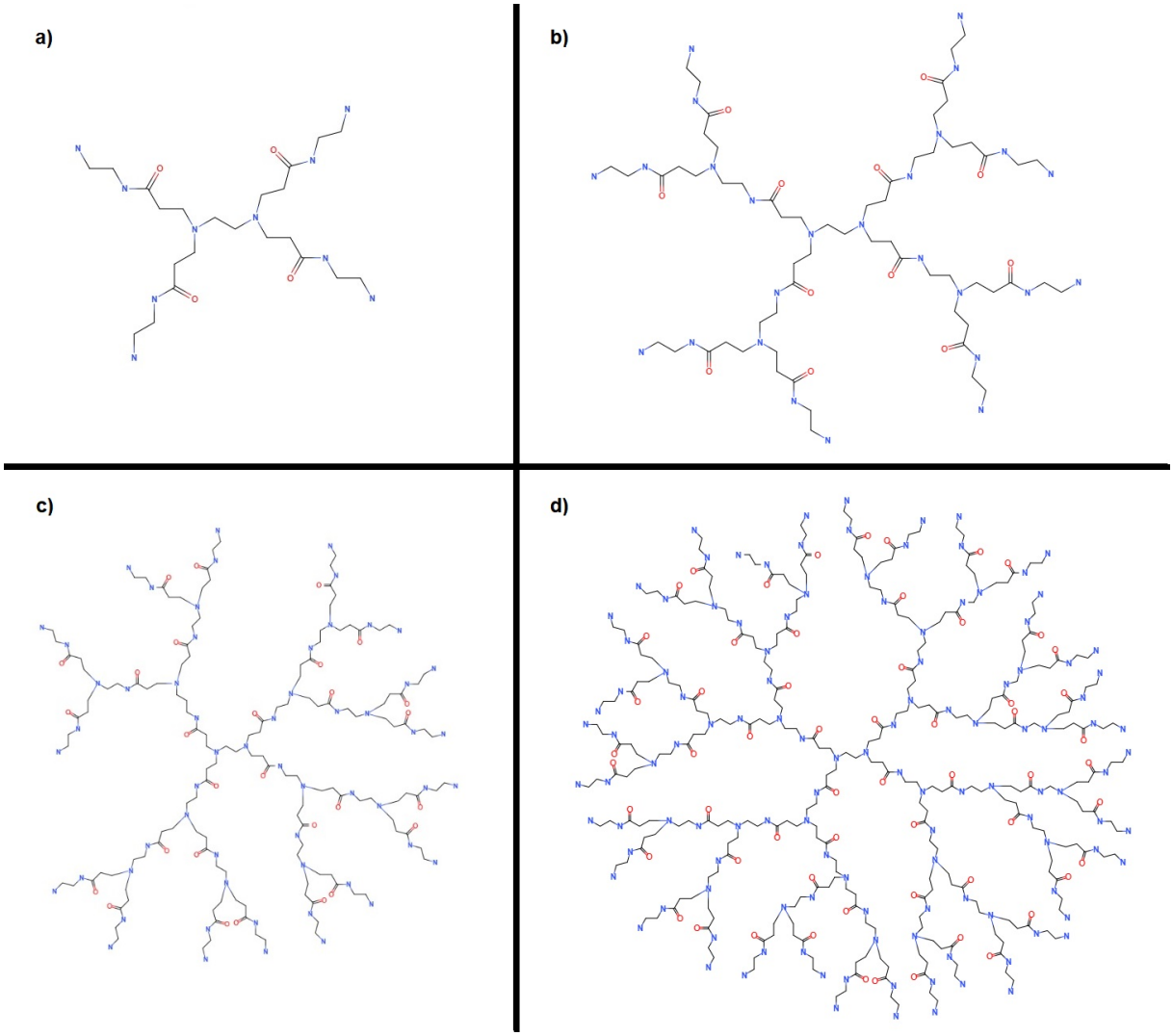

Figura 5 - Gerações de um PAMAM: em (a), é apresentado PAMAM de geração 0, em (b) o de geração 1, em (c) o de geração 2 e em (d) o de geração 3

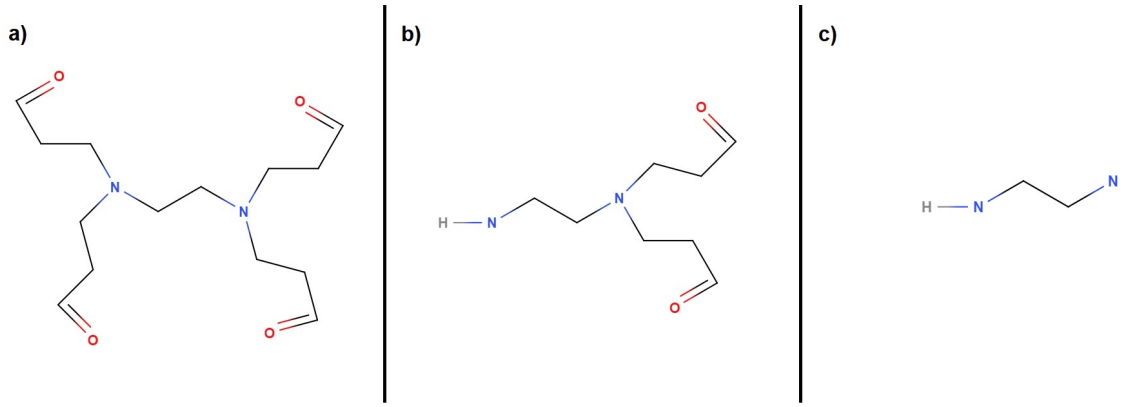

Figura 6 - BBs de um PAMAM: em (a), é apresentado o *core*, em (b) o *inter* e em (c) *term*.

A metodologia iterativa divergente envolvendo acrilonitrila usada pelo grupo Voegtle [24] inicialmente teve baixos rendimentos e dificuldades de isolamento de produto e não poderia ser usada para produzir moléculas grandes o suficiente para exibir as propriedades agora associadas ao termo dendrímero. Apenas uma década e meia depois desenvolveu-se uma modificação amplamente aprimorada da abordagem Voegtle para preparar verdadeiros dendrímeros de poli (propilenoimina). A rota desenvolvida pelo grupo DSM é particularmente notável porque também constitui uma rota comercial viável para esta família de dendrímeros de amina alifática [32].

O PAMAM é a família de dendrímeros mais referenciada e bem caracterizada da literatura [33]. O tamanho (ou seja, diâmetro) do PAMAM aumenta em aproximadamente 1 nm por geração e varia de 1,1 a 12,4 nm à medida que prolifera das gerações 1 a 10. A quantidade de repetições de unidades química idênticas no interior confere propriedades de nanocontêiner a partir da geração 4 [34]. O número exponencial de grupos terminais produzem apresentações estruturais 3D únicas de superfície, assim oferecendo extraordinárias vantagens funcionais hóspede-hospedeiro e de superfície interfacial [33]. Na Tabela 1 são apresentadas as quantidades de *inters*, *terms* e átomos. No gráfico apresentado na Figura 7 é mostrado o crescimento exponencial do número de átomos ao longo das gerações.

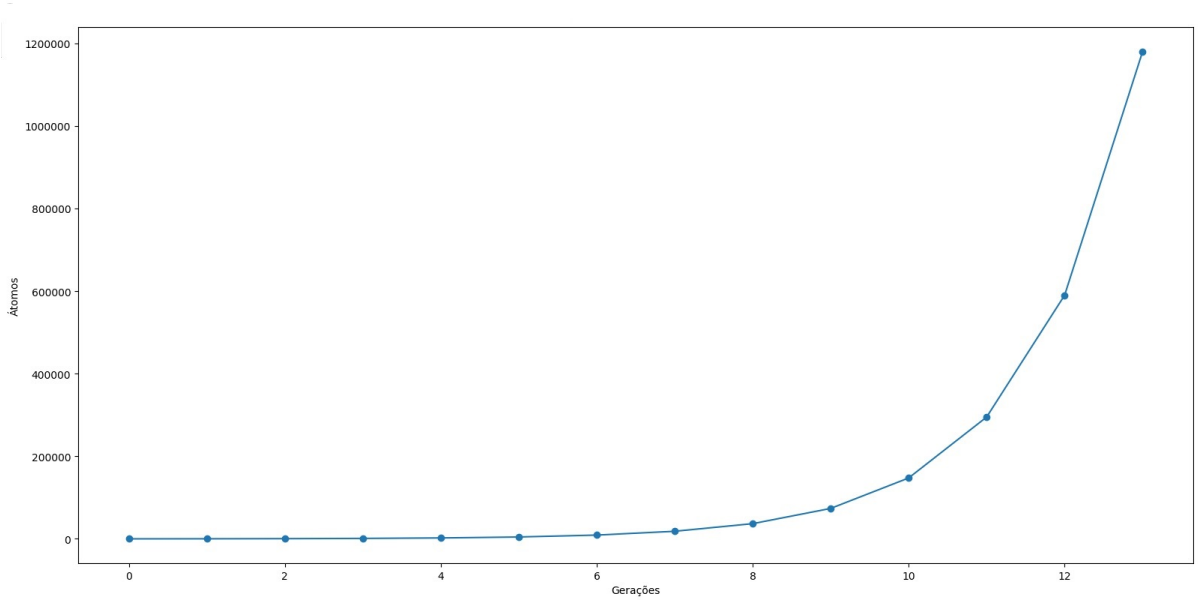

Figura 7 - Número de átomos x geração.

Tabela 1 - Número de átomos por geração.

Geração	Número de inters	Número de terms	Número de Átomos
0	0	4	84
1	4	8	228
2	12	16	516
3	28	32	1.092
4	60	64	2.244
5	124	128	4.548
6	252	256	9.156
7	508	512	18.372
8	1.020	1.024	36.804
9	2.044	2.048	73.668
10	4.092	4.096	147.396
11	8.188	8.192	294.852
12	16.380	16.384	589.764
13	32.764	32.768	1.179.588

Algumas das principais aplicações dos PAMAM são o aumento da solubilização, dissolução e biodisponibilidade de fármacos, em áreas como em tratamentos de disfunção erétil [35], de Alzheimer [36], de doenças oculares [37], entre outros [3]. Esses dendrímeros também são utilizados na entrega de agentes terapêuticos a locais alvo intracelulares para o controle da dor, inflamação e entrega oral, tópica, ocular e transdérmica. Essas aplicações podem ser observadas na Tabela 2, adaptada de [33]. Além disso, os PAMAMs são utilizados como sondas para ressonância magnética e imagens fluorescentes, como apresentado na Tabela 3, adaptada de [33].

Tabela 2 - Aplicações do PAMAM.

Aplicação	Fármaco/agente ativo	Dendrímero	Função	Referência
Gene delivery e agentes de tranfecção	DNA	PAMAM	Complexação	[38]
	DNA	PAMAM-arginina	Complexação	[39]
	DNA	G6 PAMAM	Complexação	[40]
	DNA	PAMAM	Ligação	[41]
	DNA/SiRNA	Priostar-PAMAM	Complexação	[41]
	Oligo-DNA	G4 PAMAM	Complexação	[42]
Anticorpos conjugados	60bca and J591, J591	G5 PAMAM	Transportador	[43]
	60bca and J591, J591	G5 PAMAM	Transportador	[44]
	Anticorpo monoclonal de ovelha	PAMAM	Transportador	[45]
	Anticorpo monoclonal de ovelha	PAMAM	Ligação (detecção de Antraz)	[46]
Agentes encapsulantes	Metotrexato	PAMAM	Lipossomas	[47]
Transdérmico	Indometacina	G4-PAMAM (OH & NH2)	Potenciadores de permeação	[48]
Ocular	Pilocarpina e Tropicamida	G1.5-4 PAMAM	Veículo	[49]
	G3.5 PAMAM-glucosamina	G3.5 PAMAM	Agente terapêutico (inibição de teciso cicatricial)	[50]
Oral	Propranolol	G3 PAMAM	Potenciador de solubilidade	[51]
	Naproxeno	G0 PAMAM	Potenciadores de permeação	[52]
	Niclosamida	G0-3PAMAM	Potenciador de solubilidade	[53]
	Sulfametoxazol	G3 PAMAM	Potenciador de solubilidade	[54]
	Furosemida	G0-3PAMAM	Potenciador de solubilidade	[55]
	Cetoprofeno	G5 PAMAM	Potenciador de solubilidade	[56]
	Piroxicam	G3-4PAMAM	Potenciador de solubilidade/complexação	[57]
Parenteral	N-Acetilcisteína	G4 PAMAM	Transportador	[58]
Liberação no cólon	Ácido 5-aminosalicílico	G3 PAMAM	Transportador	[59]
Géis tópicos	Nifedipina	G4 PAMAM	Potenciador de solubilidade e permeação	[60]
	5-Fluorouracil	G2-G6 PAMAM	Potenciadores de permeação	[61]
Injetáveis in vivo	Amina, superfície hidroxila	G5 PAMAM	Agente terapêutico (anti-inflamatório)	[62]

Tabela 3 - Dendrímeros como contraste para ressonância magnética.

Aplicação em Ressonância Magnética	Dendrímero	Estudos de biodistribuição em:	Referência
Câncer de mama	G6PAMAM-Cistamina	Camundongos sem pêlo	[63]
Câncer de mama	G6 PAMAM	Camundongos	[64]
Linfonodos sentinela (mamário)	G6 PAMAM	Camundongos	[65]
Linfonodos sentinela (mamário)	G8 PAMAM	Camundongos	[66]
Linfonodos sentinela (mamário)	G6 PAMAM	Camundongos	[67]
Rim	G4 PAMAM	Camundongos sem pêlo	[68]
Doença arterial coronariana	PAMAM	Humanos	[69]
Tumores ovarianos	G2PAMAM-Cistamina	Camundongos sem pêlo	[70]
Imagem óptica linfática e linfonodos sentinela	G6 PAMAM	Camundongos atímicos	[71]
Tumores	G3 PAMAM	Camundongos atímicos	[72]
Angiografia	G4 PAMAM	Ratos	[73]
Vasculatura intratumoral	G6 PAMAM	Camundongos sem pêlo	[74]
Micrometástases hepáticas	G4 PAMAM	Camundongos sem pêlo	[75]
Pool sanguíneo	G3-6 PAMAM	Camundongos sem pêlo	[76]

2 COMPUTAÇÃO EVOLUTIVA

A otimização é um processo que visa atingir um determinado objetivo utilizando os "recursos" disponíveis da melhor maneira possível. Nos algoritmos de otimização, um objetivo deve ser descrito matematicamente identificando-se suas variáveis e condições que devem ser satisfeitas [77]. Em geral, esses algoritmos são divididos em: determinísticos e estocásticos [78].

Um algoritmo determinístico apresenta sempre o mesmo resultado para o mesmo número de avaliações se o ponto de partida no espaço de busca e as condições de parada se mantiverem as mesmas, independente do momento em que é iniciado. Normalmente, esses algoritmos demandam limitações nos parâmetros, nas funções objetivo e nas restrições a serem otimizadas. Em certos casos, eles apresentam soluções matematicamente elegantes, todavia, podem necessitar de cálculos complexos podendo se tornar uma sobrecarga computacional. Eventualmente esses métodos são utilizados para otimização local [78].

As soluções de determinados tipos de problemas apresentam elevada dificuldade tanto no atendimento de requisitos de tratabilidade por ferramentas convencionais, quanto em uma formulação matemática abrangente. Nesses casos, pode-se utilizar algoritmos estocásticos que se caracterizam por serem, em geral, mais simples do ponto de vista matemático, capazes de obter soluções quase ótimas em espaços de busca amplos onde algoritmos determinísticos dificilmente conseguiriam obter bons resultados. São versáteis, não impondo muitas restrições sobre os problemas de otimização, além de apresentarem diversos resultados quase ótimos ocasionados por sua aleatoriedade, possibilitando ao usuário definir qual será utilizado [78, 79].

Estes tipos de problemas podem ser candidatos a uma abordagem baseada em computação evolutiva, uma vez que técnicas que garantem uma solução ótima não são realizáveis nestes casos [79]. A computação evolutiva começou a ser desenvolvida na década de 60 [80], gerando programas de computador utilizando aspectos da evolução natural como metáforas em sua concepção [81].

Os algoritmos evolutivos são aplicados principalmente a problemas de otimização, planejamento, projeto e gerenciamento. Esses algoritmos vem sendo utilizados principalmente nas áreas de engenharias eletrônica e elétrica, inteligência artificial, métodos teóricos da ciência da computação, aplicações interdisciplinares com ciência da com-

putação, sistemas de controle e automação, sistemas de informação e ciência do gerenciamento de pesquisa operacional [82].

A modelagem computacional de dendrímeros é um processo que demanda um grande poder computacional, ocasionado pela necessidade de se calcular o ângulo de torção referente a cada átomo. O número de átomos a cada geração do dendrímero cresce de maneira exponencial como mostrado no Capítulo 1. Além disso, não existe uma formulação matemática disponível para esse problema específico, sendo assim, para o problema abordado, são utilizados algoritmos estocásticos. Neste trabalho, para tornar possível a modelagem, são aplicados métodos de computação evolutiva.

2.1 **Conceitos Básicos**

Atualmente, toda a família de algoritmos de otimização evolutiva é conhecida como computação evolutiva (EC, do inglês, *evolutionary computation*). No domínio da computação evolutiva, podemos citar os seguintes algoritmos principais: o GA, a programação genética (GP, do inglês, *genetic programming*), a evolução diferencial (DE, do inglês, *differential evolution*), a estratégia evolutiva (ES, do inglês, *evolution strategy*) e a programação evolutiva (EP, do inglês, *evolutionary programming*) [82].

Cada uma dessas técnicas evolutivas tem muitas variações e estas são usadas em aplicações industriais diversas [82]. Na Figura 8, adaptada de [82], são apresentadas algumas abordagens das técnicas de otimização. Também na Figura 8, a sigla CMA-ES significa "estratégia de evolução da adaptação da matriz de covariância", do inglês *covariance matrix adaptation evolution strategy*.

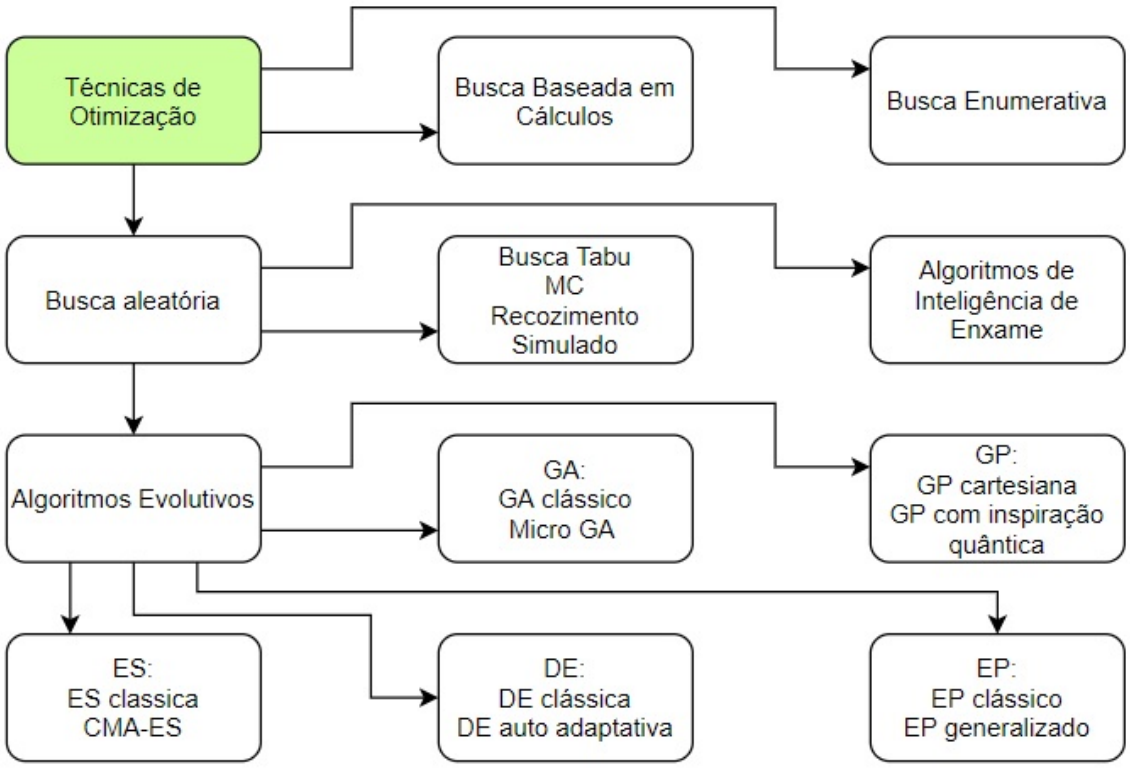

Figura 8 - Taxonomia das Técnicas de Otimização.

Com o aumento da complexidade da solução de problemas das mais diversas áreas, os algoritmos de otimização tem cada vez mais aplicações [83–86]. Na Figura 9, são apresentados os gráficos que representam o aumento da utilização de cada método entre os anos de 2000 e 2017. Os dados utilizados para a geração dos gráficos apresentados foram obtidos da base de dados *Web of Science*. O primeiro gráfico mostra o aumento das aparições do termo GA ao longo do tempo, o segundo, do termo GP, o terceiro, o DE, o quarto, a ES, o quinto, a EP e finalmente o último, o total dos 5 termos.

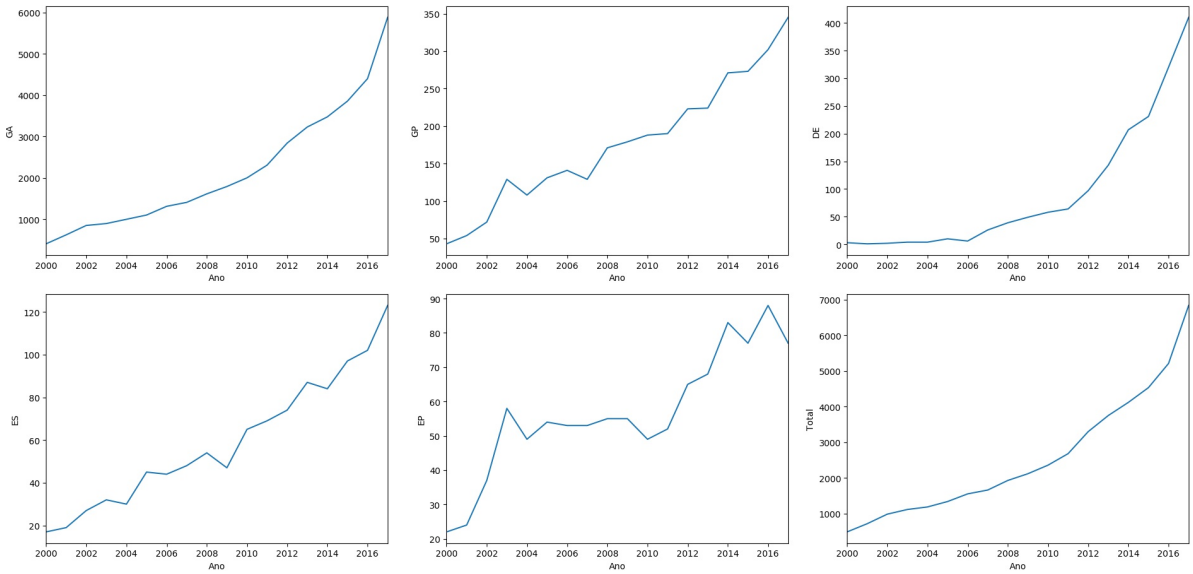

Figura 9 - Gráficos do aumento de aparições dos termos no *Web of Science* ao longo dos anos.

2.2 **Algoritmos Genéticos - GA**

GA foram criados por John Holland nos anos 60 e desenvolvidos por Holland, seus estudantes e seus colegas na universidade de Michigan entre os anos 60 e 70. Em contraste com ES e EP, o objetivo original de Holland não era solucionar problemas específicos, e sim, estudar formalmente o fenômeno da adaptação tal como ocorre na natureza com um grande número de "soluções" possíveis. Em biologia, o enorme conjunto de possibilidades é o conjunto de possíveis sequências genéticas e as "soluções" desejadas são organismos altamente adequados (organismos com alta capacidade de sobreviver e se reproduzir) [87]. Esse algoritmo simula a teoria de Charles Darwin na qual, na natureza, os indivíduos mais adaptados ao ambiente (evitando inimigos e localizando alimentos) se tornam predominantes e assim tem a maior probabilidade de sobreviver e transferir seus genes para as gerações subsequentes [88].

Alguns termos da terminologia biológica devem ser introduzidos no contexto do GA com o objetivo de fazer a analogia com a biologia natural. Todos os seres vivos são constituídos por células, que compartilham o mesmo conjunto de um ou mais cromossomos (sequencias de DNA). Um cromossomo pode, conceitualmente, ser dividido em genes, que são blocos funcionais de DNA, na qual cada gene é responsável por codificar determinada

característica do indivíduo [87]. Neste trabalho, no contexto do GA cada cromossomo é referido como um indivíduo, e cada indivíduo faz parte de uma população, que se modifica a cada geração. Além disso, neste trabalho, todas as populações tem o mesmo número de indivíduos. Outro elemento essencial para o GA é a aptidão, que na evolução natural, é a capacidade do indivíduo de sobreviver e se reproduzir. Já na evolução artificial, a aptidão é medida por uma função que deve ser maximizada ou minimizada, de acordo com o problema abordado.

Na reprodução sexuada, a cada geração, dois indivíduos pais misturam seus genes para gerar indivíduos filhos. Esse conceito é conhecido como cruzamento. Além disso, um indivíduo filho pode sofrer uma mutação, ou seja, um gene vindo dos pais pode sofrer uma modificação [87]. Na próxima geração, os indivíduos filhos passam a ser os indivíduos pais que produzem a geração subsequente. No GA, os indivíduos pais tem uma maior chance de serem selecionados de acordo com a sua aptidão. Na Figura 10, é mostrado como funciona o cruzamento e a mutação natural. Inicialmente são mostrados os indivíduos pais, em sequência estão os indivíduos pais com as marcações onde ocorrerá o cruzamento, depois podem ser vistos os indivíduos filhos que recebem partes dos pais e finalmente um indivíduo sofre mutação que é representado na cor azul. Enquanto a Figura 11 mostra como funciona o cruzamento e a mutação artificial para indivíduos binários. As etapas mostradas na figura são equivalentes as mostradas na Figura 10.

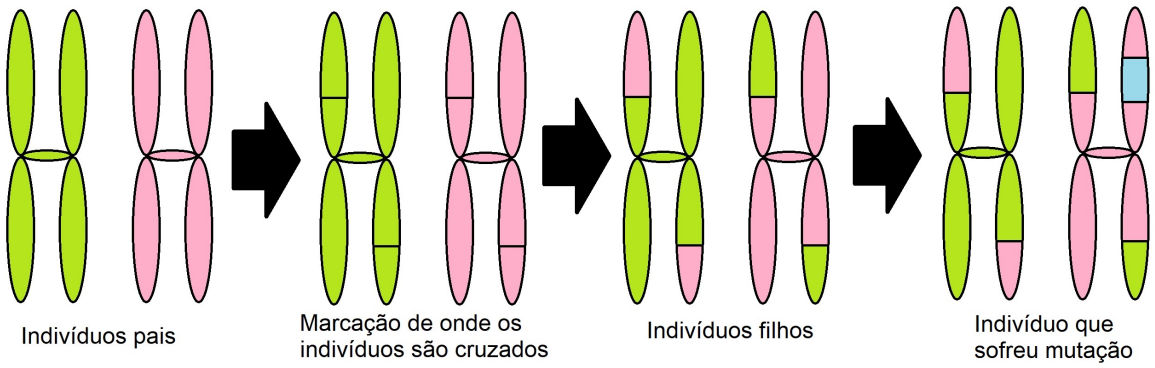

Figura 10 - Cruzamento e mutação natural.

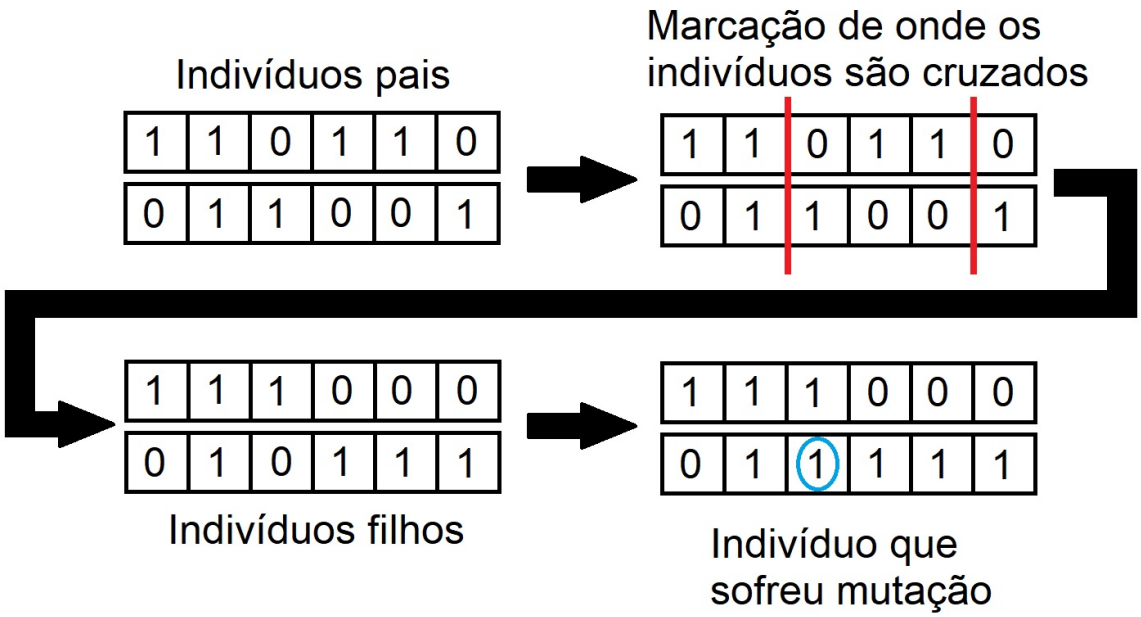

Figura 11 - Cruzamento e mutação artificial.

Com isso, são formalizados os quatro elementos essenciais para um algoritmo ser considerado um GA. São eles: uma população de indivíduos (cromossomos), os indivíduos que são selecionados para produzir a próxima geração, escolhidos com base na sua aptidão, uma nova geração criada pelo cruzamento dos indivíduos da geração anterior e a chance de ocorrerem mutações aleatórias nos indivíduos a cada geração [87].

2.2.1 Métodos do GA

No GA, temos duas variantes, uma no domínio binário e outra no contínuo (ou inteiro). No binário, cada gene do cromossomo poderá receber os valores zero ou um. Já no domínio contínuo os genes podem receber qualquer valor entre dois valores (um mínimo e outro máximo) especificados para o problema [88]. Neste trabalho, o problema abordado está no domínio dos contínuos, por isso as ferramentas serão apresentadas focando nesse domínio.

O primeiro passo para um GA é a inicialização de sua população. Para essa rotina, é utilizado um laço com a duração igual ao tamanho da população. Dentro desse laço, a cada repetição, será executado outro laço com a duração igual ao número de genes em um indivíduo. Nesse laço mais interno, é gerado um valor aleatório entre os limites inferior e superior especificados para o problema. Este procedimento tem como objetivo fazer

com que os indivíduos sejam uniformemente distribuídos pelos valores possíveis para as variáveis [88]. A título de exemplo simples, a Figura 12 mostra uma população de 200 indivíduos gerados aleatoriamente. Os indivíduos tem dois genes (chamados X e Y) e sua aptidão é a soma dos valores de seus genes. Os limites inferior e superior são 0 e 1 respectivamente. Quanto mais próximo do azul escuro é a cor do indivíduo, maior a sua aptidão.

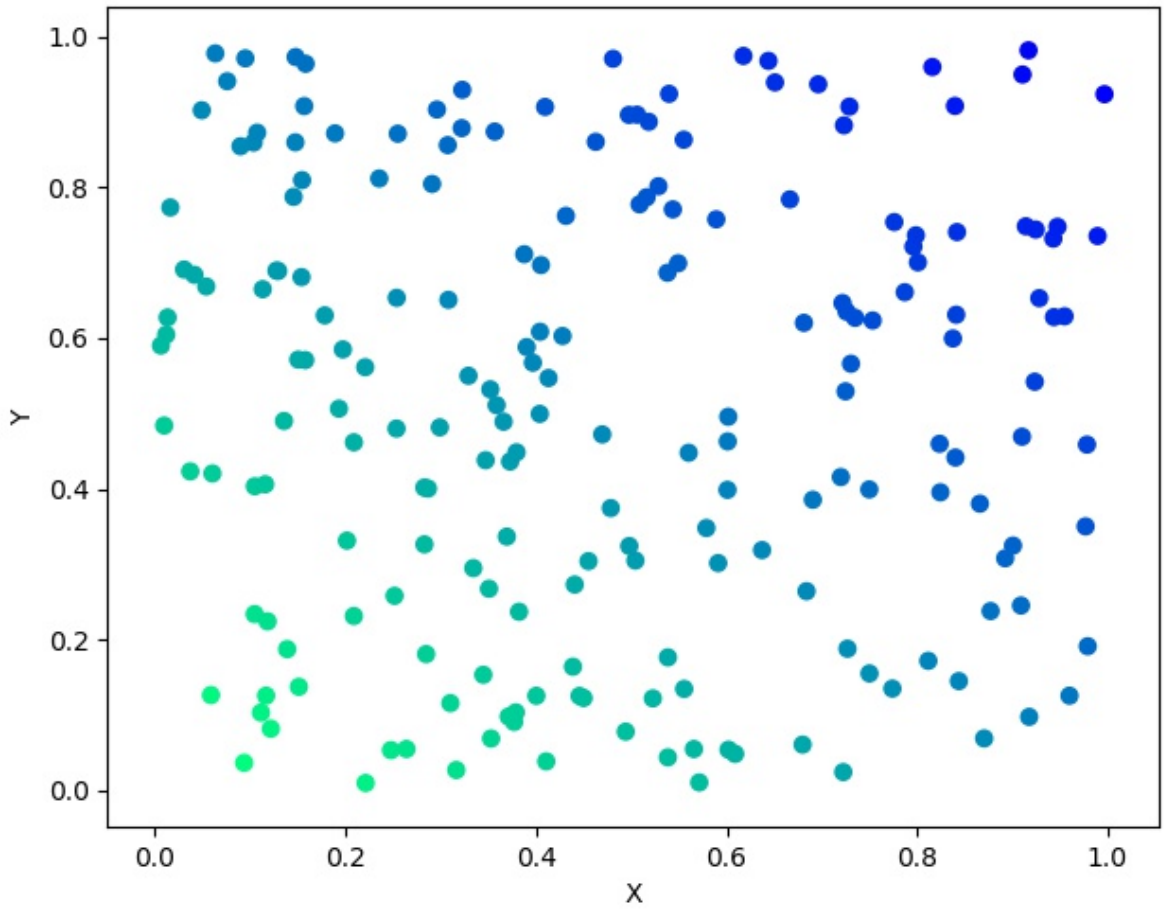

Figura 12 - População gerada aleatoriamente.

O operador de seleção simula a seleção natural. Na seleção natural, as chances de sobrevivência aumentam proporcionalmente à aptidão. Ser selecionado equivale a passar seus genes para a próxima geração. No GA, esse processo é simulado por uma roleta. A roleta normaliza a aptidão dos indivíduos e realiza um sorteio no qual a chance de um indivíduo ser selecionado é proporcional à sua aptidão. Como o GA simula a seleção natural, indivíduos com aptidão baixa não são excluídos, porém tem uma menor chance de

passar seus genes para a próxima geração [88]. Na Figura 13, é exemplificada uma roleta com 20 indivíduos e as larguras das fatias representam a probabilidade dos indivíduos serem selecionados para a próxima geração.

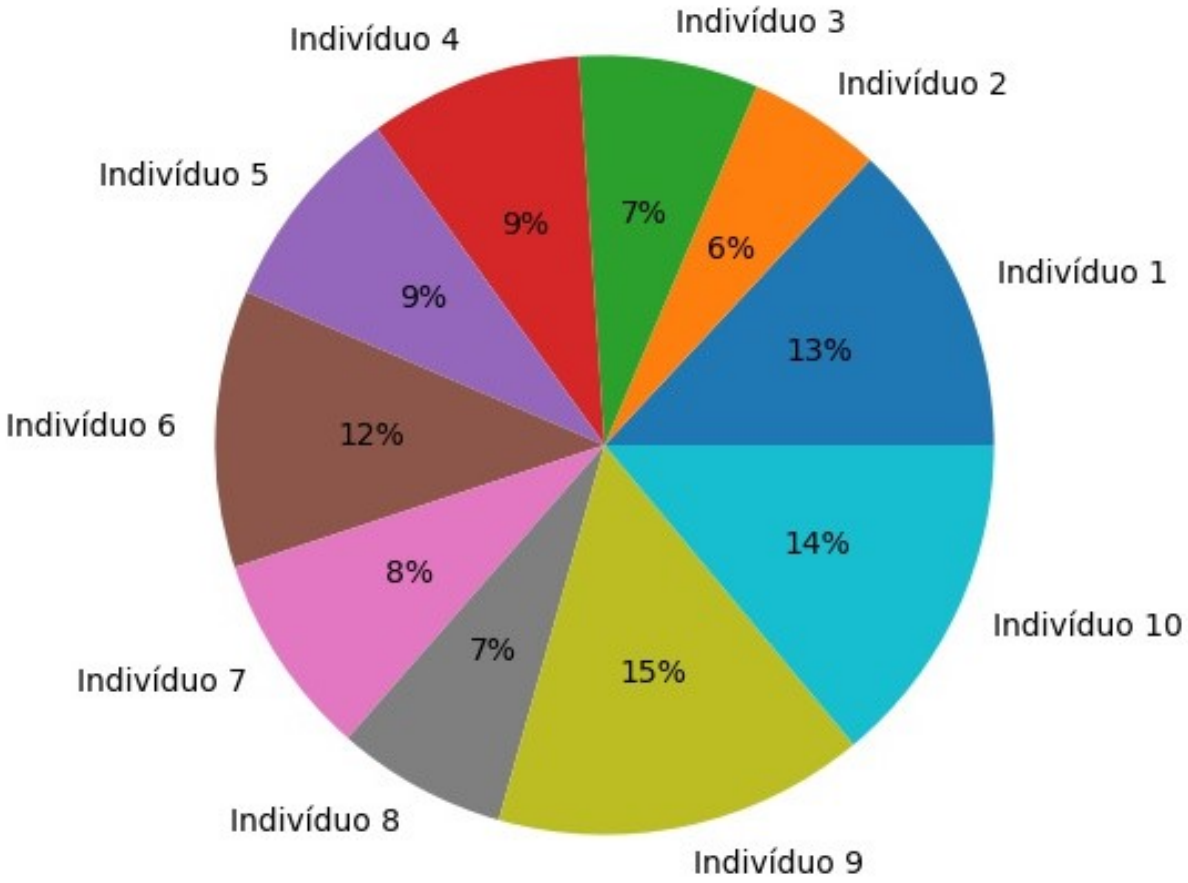

Figura 13 - Exemplo de roleta.

Um conceito importante nesse contexto é a pressão seletiva, que indica o quanto o indivíduo é beneficiado pela sua aptidão. Se a pressão seletiva for muito baixa, o algoritmo irá ter mais dificuldade para convergir, tornando a execução do GA mais longa que o necessário para encontrar a melhor solução. Porém, se a pressão seletiva for muito alta, o algoritmo pode convergir prematuramente e ficar preso a uma solução sub-ótima [89].

Além da seleção por roleta, o GA possui diversos outros tipos de seleção, como: seleção por torneio [89], seleção de Boltzman [90], seleção por aptidão uniforme [91], seleção de estado estacionário [92], entre outras. Neste trabalho, além da seleção por

roleta, utilizamos a seleção por torneio, que consiste em selecionar uma quantidade n de indivíduos e apenas aquele com melhor aptidão será escolhido para fazer o cruzamento. O número n de indivíduos é chamado de tamanho de torneio; por exemplo, se n for igual a dois, temos um torneio de dois indivíduos. A escolha do tamanho de torneio controla a pressão seletiva do algoritmo. Quanto maior o tamanho de torneio, maior é a pressão seletiva [89]. Na Figura 14, é apresentado um diagrama que representa o funcionamento de uma seleção por torneio, com tamanho de torneio igual a três. Inicialmente é mostrada a população inicial, na sequência são mostrados seis indivíduos divididos em dois grupos de três, onde ocorrem os torneios, e então são mostrados os indivíduos selecionados nos torneios para sofrerem cruzamento. Finalmente, é mostrado o indivíduo gerado na população da próxima geração.

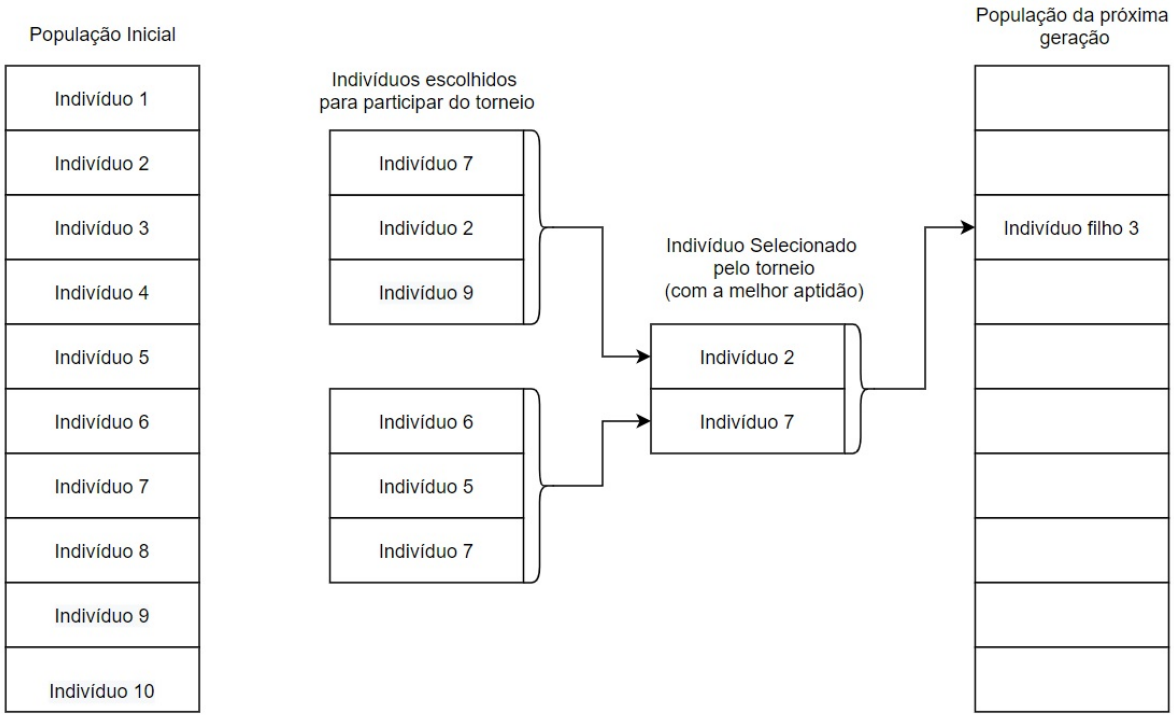

Figura 14 - Diagrama da seleção por torneio.

Outro operador essencial é o cruzamento, que simula a reprodução sexuada. Nele, os genes dos indivíduos pais são misturados para gerar dois indivíduos filhos. Os cruzamentos artificiais mais comuns são os de um e dois pontos [88]. Na Figura 15, é mostrado o cruzamento de um e dois pontos. Na Figura 15, a cor azul e a cor vermelha representam de qual pai o cromossomo se origina e cada gene contém um número real.

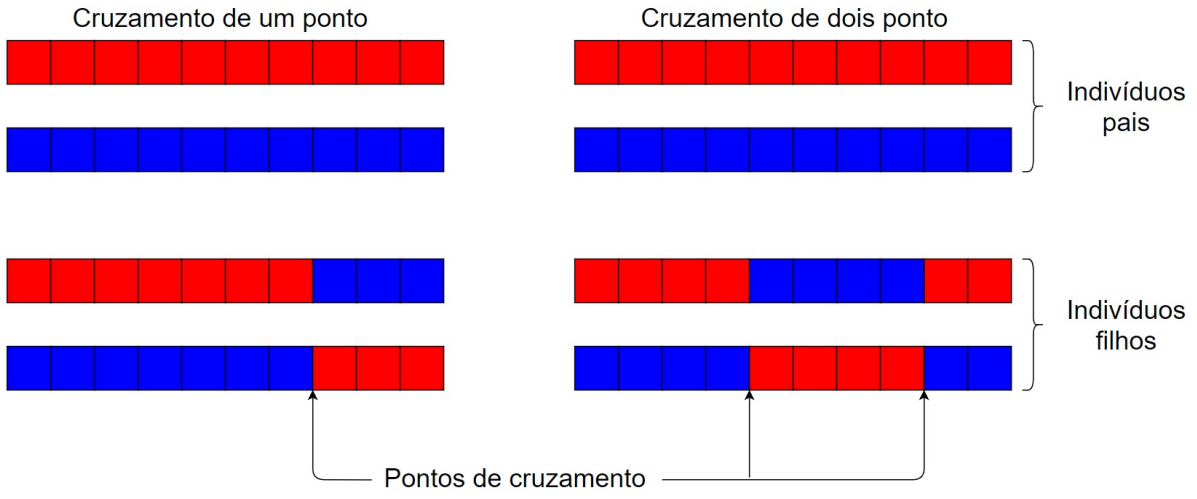

Figura 15 - Cruzamento de um ou dois pontos.

Um conceito importante é a probabilidade de cruzamento. Essa probabilidade será um número entre zero e um. A cada cruzamento, é escolhido um número entre zero e um de maneira pseudoaleatória e se o número for menor que a probabilidade de cruzamento, o indivíduo filho será transmitido para a próxima geração; caso contrário, o indivíduo pai continuará na próxima geração. Além desses cruzamentos, existem vários outros tipos, como por exemplo: cruzamento uniforme [93], cruzamento meio uniforme [94], cruzamento com três pais [95], cruzamento mascarado [96], entre outros. Neste trabalho, além dos cruzamentos por um e dois pontos, vamos estudar o cruzamento uniforme e o cruzamento "*blend*"(em português, "cruzamento por combinação"), que estão implementados na biblioteca Python DEAP [22].

No cruzamento uniforme, cada gene dos pais pode ser trocado individualmente. Um parâmetro é definido, a priori, com um valor entre zero e um. Durante a execução do cruzamento, para cada gene de um indivíduo pai, é gerado um valor pseudoaleatório entre zero e um. Se esse valor for menor que o definido para o parâmetro, os genes dos pais são trocados [97]. Na Figura 16, é apresentado um exemplo de como o cruzamento uniforme funciona. No exemplo na Figura 16, o parâmetro escolhido foi 0,5, as cores representam de qual indivíduo pai o cromossomo se origina e a barra branca com números gerados aleatoriamente representa uma máscara que define se os cromossomos dos pais são trocados ou não. Na Figura 16, cada gene representa um número real.

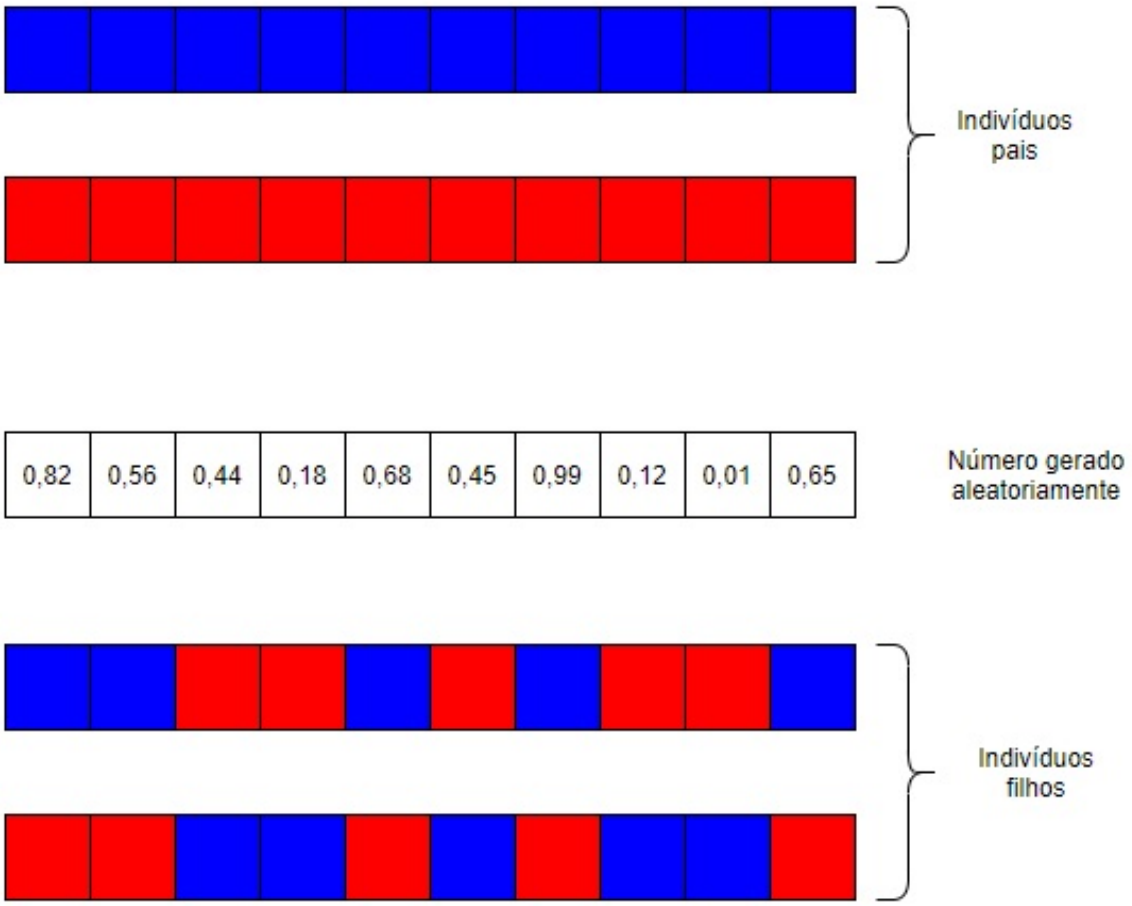

Figura 16 - Exemplo de cruzamento uniforme.

O cruzamento por combinação pode ser utilizado em problemas no domínio dos contínuos. Nele, os genes dos pais são misturados para gerar os indivíduos filhos. Um parâmetro é definido com o objetivo de controlar a proporção dos genes de cada pai nos filhos [98]. Nesse método cada gene dos indivíduos filhos recebe um percentual dos genes correspondentes de cada um dos pais. Na Figura 17, é apresentado um exemplo de como o cruzamento por combinação funciona. No exemplo, o parâmetro foi definido de maneira que os genes do filho 1 recebem 75% dos genes do pai 1 e 25% dos genes do pai 2, enquanto o filho 2 recebe 25% do pai 1 e 75% do pai 2. Como os indivíduos filhos são obtidos ponderando os indivíduos pais eles são mais próximos de um dos pais. Esse comportamento pode ser observado pelas cores dos indivíduos pois cada filho tem a cor mais parecida com a de um dos pais. Na Figura 17, cada gene contém um número real.

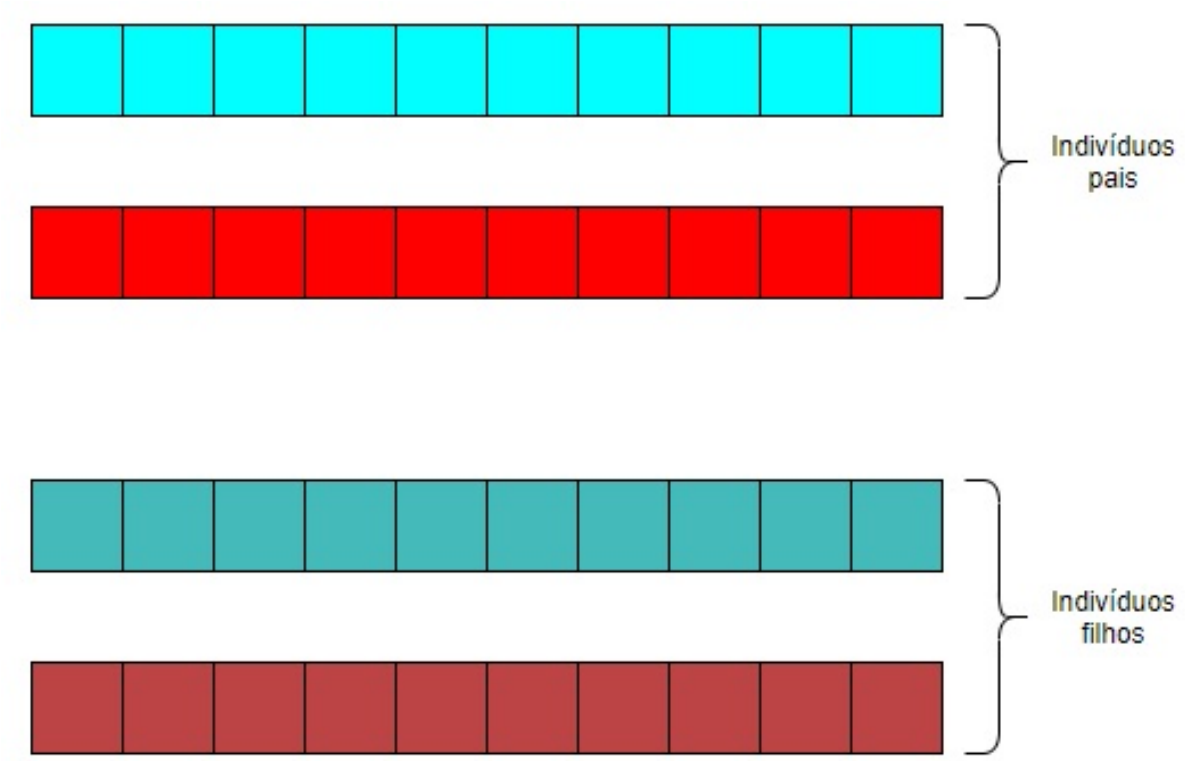

Figura 17 - Exemplo de cruzamento por combinação.

Mais um operador essencial para o GA é a mutação. Esse operador gera uma maior diversidade nos indivíduos, podendo assim, percorrer áreas do espaço de busca que não seriam observadas apenas com o cruzamento que não gera novos genes [88].

A mutação causa mudanças aleatórias nos genes. O parâmetro chamado "probabilidade de mutação" é um valor entre zero e um, normalmente um valor muito baixo. Após o cruzamento, é escolhido um número entre zero e um de maneira pseudoaleatória e, se o número for menor que a probabilidade de mutação, os genes do indivíduo filho podem ser modificados de maneira pseudoaleatória [88]. Alguns dos tipos de mutação são: mutação Gaussiana [99], mutação uniforme [100] e mutação por mudança de índice [101].

A mutação Gaussiana é realizada gerando um ruído Gaussiano ao redor do gene com um desvio padrão proporcional ao valor máximo que pode ser assumido pelo gene [99]. A proporção é escolhida de acordo com o parâmetro definido *a priori* no algoritmo.

A mutação uniforme é realizada gerando um valor aleatório entre os valores máximo e mínimo possíveis para o gene [100].

Na mutação por mudança de índice, dois genes do indivíduo trocam de posição dentro do cromossomo [101]. Na Figura 18, é mostrado como são realizadas as mutações

Gaussiana, uniforme e por mudança de índice. As cores azul escuro e vermelho representam os cromossomos vindo dos indivíduos pais. Na mutação uniforme, os genes que sofrem mutação mudam para cores bem diferentes das originais pois mudam de maneira totalmente aleatória. Na mutação Gaussiana, os genes que sofrem mutação continuam mais parecidos com os originais que na mutação uniforme pois a mutação ocorre nas proximidades do gene pai. Na mutação por mudança de índices, pelas cores, é possível observar que os cromossomos do indivíduo mudam de posição. Na Figura 18, cada gene contém um número real.

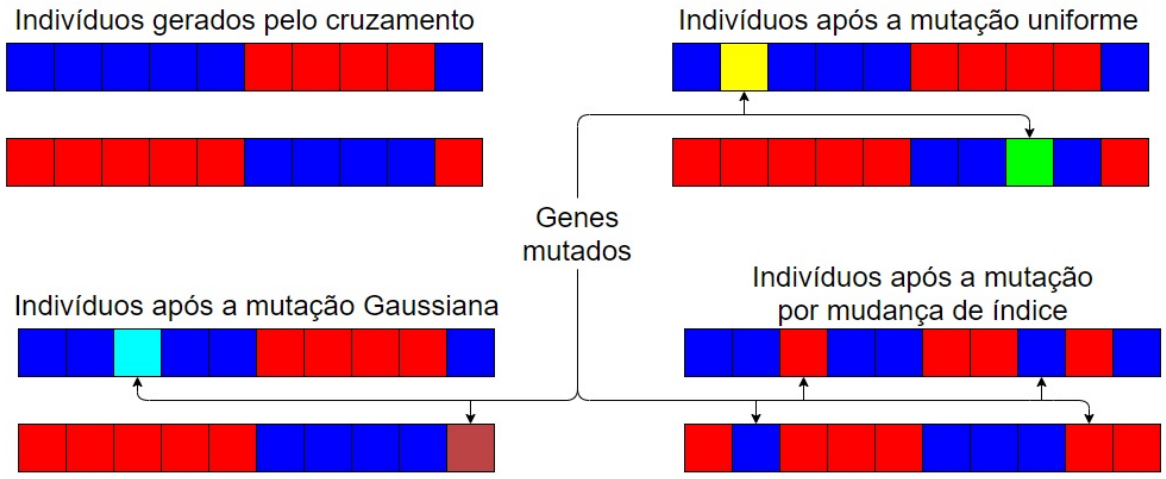

Figura 18 - Representação das mutações.

Neste trabalho, a Seção sobre o GA é consideravelmente maior que as seções relacionadas aos outros algoritmos. Esse fato se dá por diversos fatores: primeiro, esse algoritmo é melhor documentado, por haver um número maior de estudos sobre ele, conforme pode ser observado na Figura 9; segundo, os métodos do GA são mais diversificados entre si, assim sendo necessárias mais explicações sobre eles. E, por último, alguns métodos utilizados nos outros algoritmos são muito parecidos com os do GA e, por isso, são explicados de maneira mais sucinta nas próximas seções.

2.3 **Evolução Diferencial - DE**

A DE é um método de busca que gera uma população de maneira pseudoaleatória de indivíduos abrangendo todo o espaço de parâmetros e, posteriormente, essa população

inicial é avaliada. A população que passa de uma geração para a seguinte é chamada de população agente [102].

Na DE, a cada geração, é criada uma nova população pseudoaleatoriamente. Então, cada indivíduo "i" da população gerada pseudoaleatoriamente é mesclada a alguns indivíduos escolhidos pseudoaleatoriamente da população agente [102, 103]. Neste trabalho, a mescla é chamada de cruzamento. Na Tabela 4, são mostrados os tipos de cruzamentos utilizados neste trabalho, adaptados de [103]. Na Tabela 4, $\vec{x_1}$, $\vec{x_2}$, $\vec{x_3}$, $\vec{x_4}$ e $\vec{x_5}$ são indivíduos selecionados aleatoriamente, já $\vec{x_i}$ é o indivíduo atual da população e o $\vec{x_b}$ é o melhor indivíduo já observado. Enquanto isso, "F" e λ são valores entre zero e um que ponderam as diferenças apresentadas.

Tabela 4 - Tipos de cruzamento utilizados neste trabalho..

Tipo de Cruzamento	Nome do Cruzamento	Fórmula do Cruzamento
aleatório 1	rand/1	$\vec{x_1} + F(\vec{x_2} - \vec{x_3})$
aleatório 2	rand/2	$\vec{x_1} + F(\vec{x_2} + \vec{x_3} - \vec{x_4} - \vec{x_5})$
com o melhor indivíduo 1	best/1	$\vec{x_b} + F(\vec{x_2} - \vec{x_3})$
com o melhor indivíduo 2	best/2	$\vec{x_b} + F(\vec{x_2} + \vec{x_3} - \vec{x_4} - \vec{x_5})$
aleatório com o melhor	rand-to-best/1	$\vec{x_1} + \lambda(\vec{x_b} - \vec{x_1}) + F(\vec{x_2} - \vec{x_3})$
atual com um aleatório	current-to-rand/1	$\vec{x_i} + \lambda(\vec{x_1} - \vec{x_i}) + F(\vec{x_2} - \vec{x_3})$
atual com o melhor	current-to-best/1	$\vec{x_i} + \lambda(\vec{x_b} - \vec{x_i}) + F(\vec{x_1} - \vec{x_2})$

Após os indivíduos serem mesclados, cada indivíduo "i" mesclado é avaliado e comparado ao indivíduo "i" da população agente e, caso seja melhor, esse novo indivíduo substitui o da população agente [102, 103]. Na Figura 19, é mostrado como é realizado o cruzamento e depois a seleção na DE. Na Figura 19 uma população é gerada mesclando os indivíduos da população agente e o melhor indivíduo, dependendo do tipo de cruzamento selecionado, para em sequência gerar uma população selecionada. A população selecionada é gerada adicionando um indivíduo "i" escolhido comparando os indivíduos "i" da população agente e da população gerada.

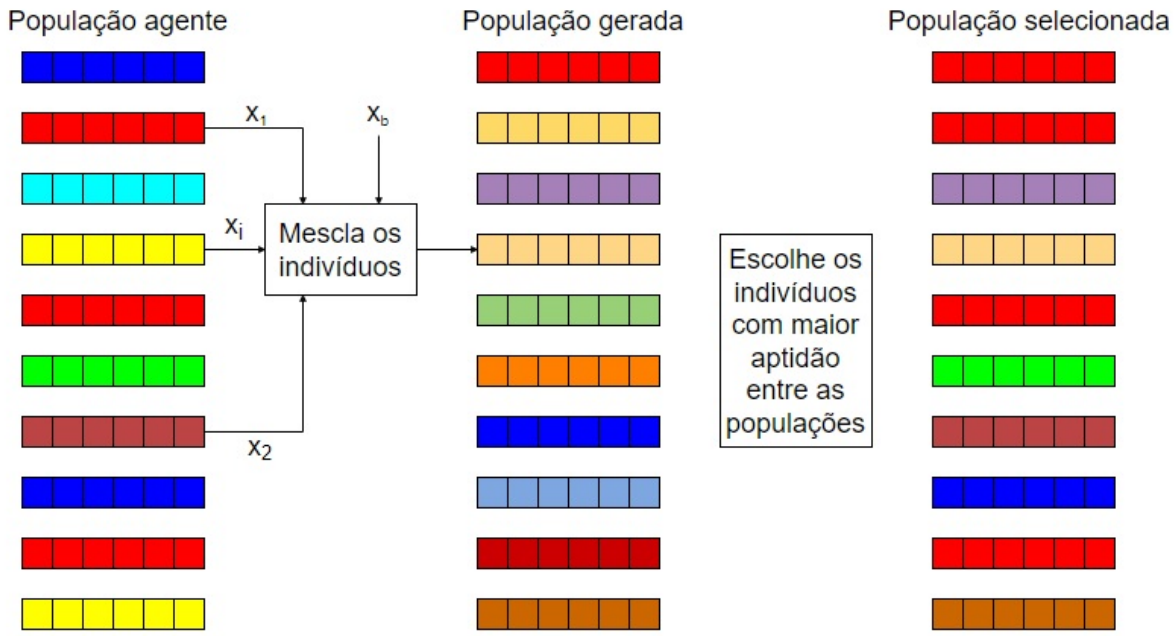

Figura 19 - Cruzamento e evolução da DE.

2.4 **Algoritmos de Inteligência de Enxame - PSO**

O PSO é um algoritmo que se baseia no comportamento de uma revoada de pássaros, cada indivíduo seguindo as ações do melhor do grupo, mas se baseando também em suas próprias experiências [104]. Assim, o PSO simula o comportamento de um grupo agindo de maneira a aumentar a capacidade cognitiva de cada indivíduo. Esse processo se dá baseado em suas experiências e nas experiências dos outros indivíduos do grupo, compartilhando o conhecimento obtido [105].

No PSO, cada indivíduo é chamado de partícula, onde cada partícula é uma solução possível para o problema a ser otimizado e o grupo de todas as partículas é chamado de enxame. Cada partícula inicia o movimento em uma posição e com uma velocidade pseudoaleatória. O processo de atualização da posição e velocidade é baseado na sua experiencia pessoal e na melhor experiência pessoal do grupo [104, 105].

O PSO tem cinco princípios básicos. O primeiro é a proximidade: o enxame deve ser capaz de realizar cálculos de espaço e velocidade. O segundo é a qualidade: o enxame deve ser capaz de reagir ao ambiente. O terceiro é a resposta diversa: o enxame não deve manter todas as partículas num mesmo espaço. O quarto é a estabilidade: o enxame deve mudar o seu comportamento de acordo com as mudanças do ambiente. O quinto é

a adaptabilidade: o enxame deve ser capaz de mudar o modo de comportamento quando vale o preço computacional [104].

Na Equação 1 é apresentada a maneira na qual a velocidade de uma partícula é atualizada [106–109].

$$\overrightarrow{v}_i(t+1) = U_1 \times rnd() \times (\overrightarrow{p}_g - \overrightarrow{x}_i(t)) + U_2 \times rnd() \times (\overrightarrow{p}_i - \overrightarrow{x}_i(t)) + \omega[\overrightarrow{v}_i(t)] \quad (1)$$

Onde:

- $\vec{v_i}(t+1)$ é a velocidade atualizada;
- $\vec{v_i}(t)$ é velocidade atual;
- $\vec{x_i}(t)$ é posição atual;
- U_1 é o coeficiente para o controle da componente social;
- U_2 é o coeficiente para o controle da componente cognitiva;
- $rnd()$ é um número aleatório entre zero e um;
- $\vec{p_g} - \vec{x_i}(t)$ é o termo de aceleração pela distância em relação a melhor posição já visitada por qualquer partícula;
- $\vec{p_i} - \vec{x_i}(t)$é o termo de aceleração pela distância em relação a melhor posição já visitada pela própria partícula;
- ω é o parâmetro de peso de inércia.

Na Equação 2 é apresentada a maneira na qual a posição de uma partícula é atualizada [106–109].

$$\overrightarrow{x}_{\mathrm{i}}(t+1) = \overrightarrow{x}_{\mathrm{i}}(t) + \overrightarrow{v}_{\mathrm{i}}(t+1) \quad (2)$$

Onde:

- $\vec{x_i}(t+1)$ é a posição atualizada;
- $\vec{x_i}(t)$ é posição atual;

- $\vec{v_i}(t+1)$ é a velocidade atualizada.

Na Figura 20, é apresentado graficamente como a posição de uma partícula é atualizada, onde, nesse exemplo U_1, U_2, ω e $rnd()$ são iguais a um. Na Figura 20 (a), é apresentado um enxame com partículas posicionadas aleatoriamente. A partícula em vermelho é aquela com a melhor aptidão já observada e a partícula em verde é a que vai ter a posição atualizada. Os vetores apresentados são os utilizados para atualizar a posição da partícula. Na Figura 20 (b), é mostrado no vetor em azul a velocidade atualizada. Na Figura 20 (c), é mostrada a posição atualizada da partícula observada, que está em verde.

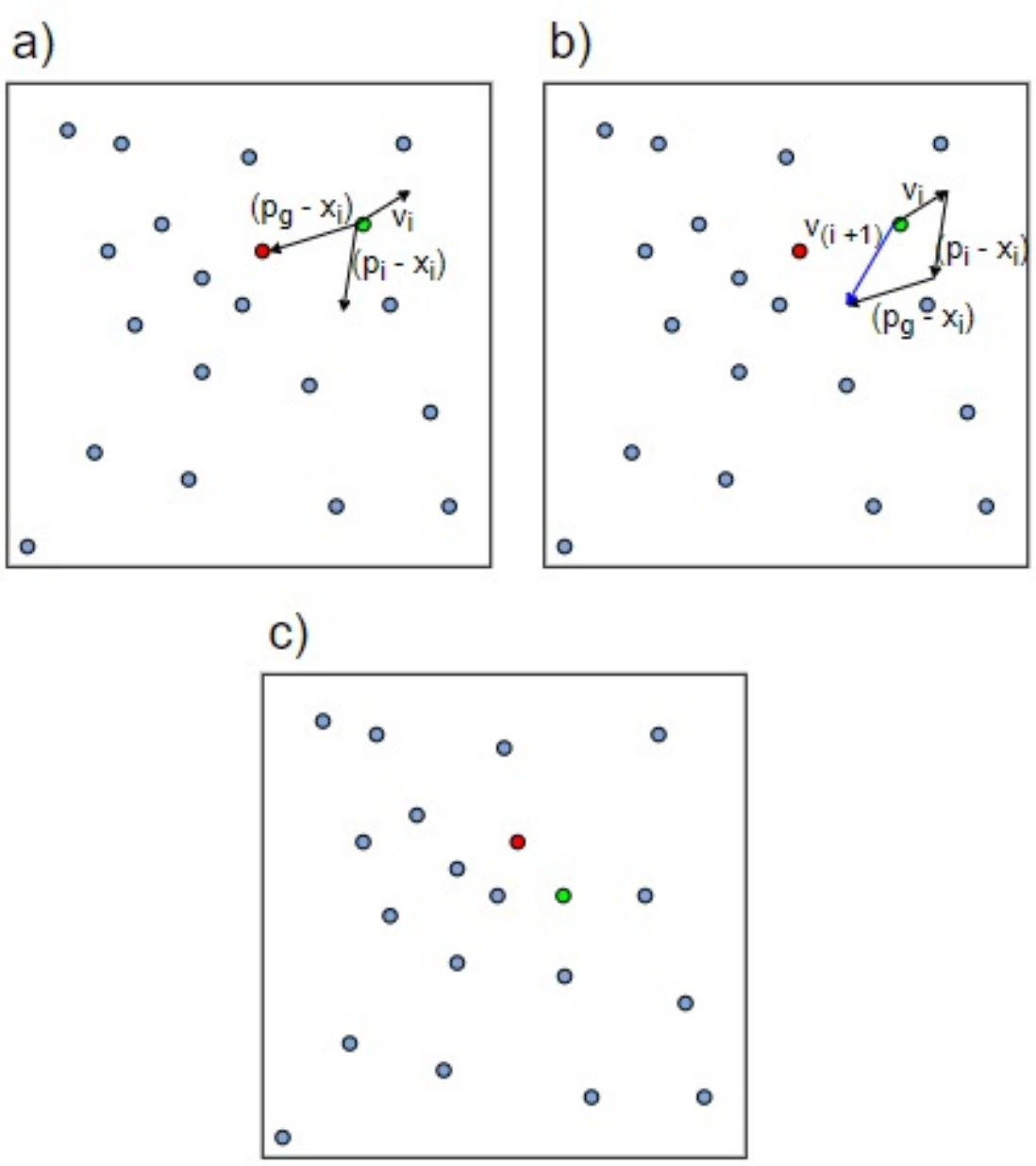

Figura 20 - Atualização da posição de uma partícula no enxame.

2.5 Estratégia de Evolução da Adaptação da Matriz de Covariância - CMA-ES

A CMA-ES foi desenvolvida por Hansen [110] e é um otimizador "caixa-preta" para otimização global, utilizado no domínio dos números contínuos, principalmente para problemas não-lineares e não-convexos [111–113]. O método tem como base dois conceitos importantes. São eles: a direção em que uma modificação ocorre e a quantidade de

informação representada por essa modificação. Esses conceitos também são conhecidos como gradiente e tamanho de passo [114].

Além destes conceitos, o algoritmo utiliza dois princípios importantes. O primeiro é o princípio da máxima verossimilhança que calcula a média da distribuição, de maneira que os passos anteriores bem sucedidos sejam maximizados. Já o segundo é o caminho evolutivo, que armazena as informações de correlação entre gerações consecutivas [114]. Quando a direção de uma modificação se repete em duas gerações consecutivas, o passo da evolução é aumentado. Esse princípio ajuda a evitar mínimos locais além de acelerar a busca pela convergência do algoritmo [115]. Na Figura 21, é mostrado um exemplo de uma CMA-ES ao longo de nove gerações, com uma população de dez indivíduos aplicados ao *benchmark* Rastringin. Na Figura 21, quanto mais próximo ao azul escuro, maior é a aptidão do elemento naquela geração e a elipse vermelha mostra a área onde os indivíduos são gerados.

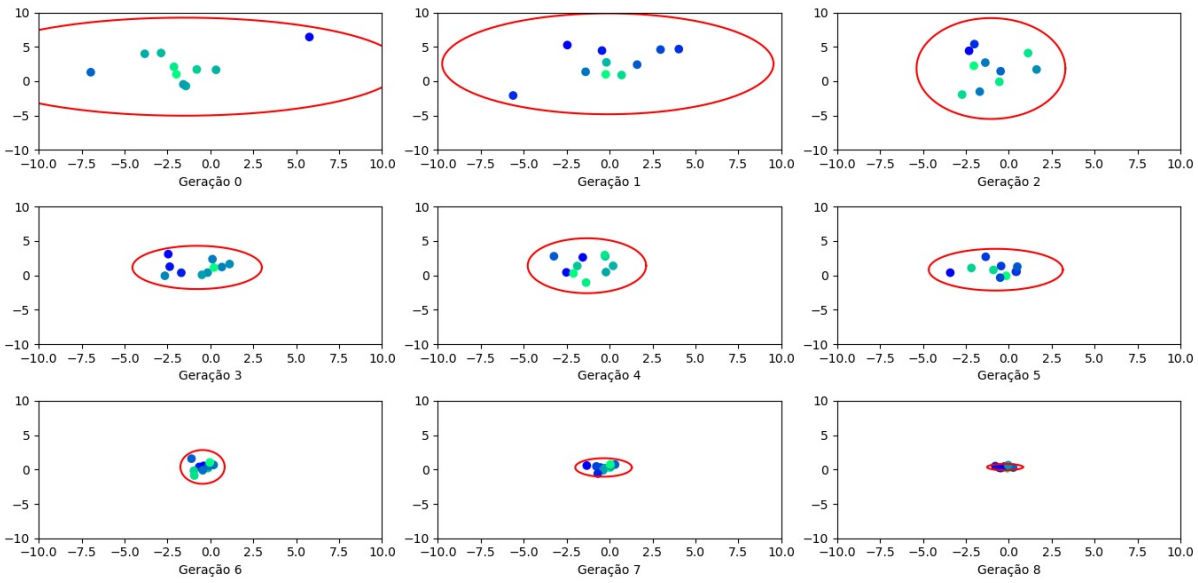

Figura 21 - Evolução de uma população ao longo das gerações utilizando a CMA-ES.

A CMA-ES se baseia em uma distribuição normal de amostras em um vetor de média $\vec{m}$, também chamado de centroide, uma matriz de covariância C de tamanho $n \times n$ e um tamanho de passo σ [115]. Os indivíduos gerados como soluções para o problema são classificados por sua aptidão. Em sequência os parâmetros são adaptados iterativamente e por último uma próxima geração é criada [116].

A cada geração o método atualiza os parâmetros da CMA-ES. O primeiro passo

para a atualização é o ranqueamento dos indivíduos da população de acordo com a sua aptidão. Posteriormente, o primeiro parâmetro ajustado é o centroide, que terá sua posição adaptada de acordo com as posições dos μ melhores indivíduos, onde μ é um número entre um e o tamanho total da população, onde tipicamente é escolhido um valor baixo. Na Figura 22, é mostrado como ocorre a atualização da posição do centroide de uma população com 200 indivíduos e um μ igual a 10. Na Figura 22 (a), é mostrado como a aptidão se comporta ao longo do espaço de busca. Quanto mais escuro, maior é a aptidão no ponto. Na Figura 22 (b), é mostrada uma população gerada aleatoriamente e o "x" vermelho representa a posição do centroide. Na Figura 22 (c), são mostrados os μ indivíduos com melhor aptidão da população. Finalmente, na Figura 22 (d), é mostrado o novo centroide baseado nos μ melhores indivíduos.

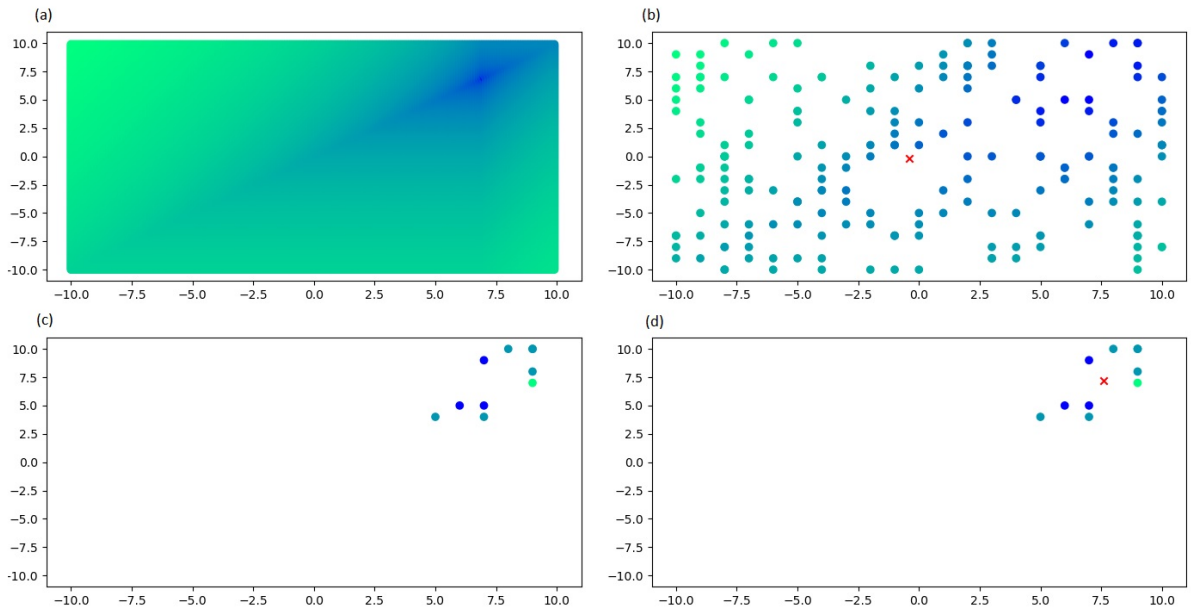

Figura 22 - Atualização da posição do centroide.

Depois é iniciada a atualização do caminho evolutivo. O primeiro parâmetro atualizado é o vetor trajetória $\vec{p_\sigma}$, onde, para isso, é utilizada a Equação 3 [110, 114, 116, 117].

$$\overrightarrow{p_\sigma}(g+1) = (1-c_\sigma)\overrightarrow{p_\sigma}(g) + \sqrt{c\sigma(2-c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{-\frac{1}{2}}(g)\frac{\overrightarrow{m}(g+1) - \overrightarrow{m}(g)}{\sigma(g)} \tag{3}$$

Onde:

- $\vec{p_\sigma}(g+1)$ é o vetor trajetória atualizado;
- $\vec{p_\sigma}(g)$ é o vetor trajetória atual;
- c_σ é a taxa de aprendizado do tamanho do passo;
- μ_{eff} é a variância efetiva da seleção, calculado de acordo com a Equação 4;

$$\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} \omega_i^2} \tag{4}$$

- $\mathbf{C}$ é a matriz de covariância;
- $\vec{m}(g+1)$ é o centroide da próxima geração;
- $\vec{m}(g)$ é o centroide da geração atual;
- ω_i designa os pesos aplicados na média ponderada usada para calcular o centroide.

Em seguida o valor do tamanho do passo é atualizado. Para isso é utilizada a Equação 5 [110, 114, 116, 117].

$$\sigma(g+1) = \sigma(g) \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\overrightarrow{p}_\sigma(g+1)||}{E||\mathcal{N}(0,I)||} - 1\right)\right) \tag{5}$$

Onde:

- d_σ é um parâmetro de suavização. Seu cálculo é apresentado na Equação 6.

$$d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) \tag{6}$$

- $E||\mathcal{N}(0,I)||$ é a expectativa normal de um vetor aleatório $||\mathcal{N}(0,I)||$. Seu cálculo é apresentado na Equação 7.

$$E||\mathcal{N}(0,I)||) = \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right) \tag{7}$$

A função de *Heaviside* evita o crescimento exagerado do caminho evolutivo quando $\overrightarrow{p_\sigma}$ for muito grande. A Equação 8 apresenta a função de *Heaviside* [110, 114, 116, 117].

$$\overrightarrow{h_\sigma}(g+1) = \begin{cases} 0, & \text{se } \frac{||\overrightarrow{p_\sigma}(g+1)||}{\sqrt{1-(1-c_\sigma)^2}} < \left(1{,}4 + \frac{2}{n+1}E||\mathcal{N}(0,I)||\right) \\ 1, & \text{se não} \end{cases} \tag{8}$$

Então, para se atualizar o caminho evolutivo é utilizada a Equação 9.

$$\overrightarrow{p_c}(g+1) = (1-c_c)\overrightarrow{p_c}(g) + h_\sigma\sqrt{c_c(2-c_c)\mu_{\text{eff}}}\frac{\overrightarrow{m}(g+1) - \overrightarrow{m}(g)}{\sigma(g)} \tag{9}$$

Onde:

- $\vec{p_c}(g+1)$ é o caminho evolutivo atualizado.
- $\vec{p_c}(g)$ é o caminho evolutivo que será atualizado.
- c_c é a taxa de aprendizado da matriz de covariância.

Na sequência a matriz de covariância é atualizada de acordo com a Equação 10.

$$\begin{aligned} \mathbf{C}(g+1) &= (1-c_1-c_\mu)\mathbf{C}(g) + c_1(\overrightarrow{p_c}(g)\overrightarrow{p_c^T}(g) \\ &+ \delta(h_\sigma(g+1))\mathbf{C}(g)) + c_\mu\sum_{i=1}^{\mu}\omega_i x_{i:\lambda}(g+1)x_{i:\lambda}^T(g+1) \end{aligned} \tag{10}$$

Onde:

- c_1 é a taxa de aprendizado média.
- c_μ é a taxa de aprendizado da população.
- $\delta(h_\sigma(g+1))$ é um parâmetro controlado pela Equação 11.

$$\delta(h_\sigma(g+1)) = (1-h_\sigma(g+1))c_c(2-c_c) \tag{11}$$

- $x_{i:\lambda}$ é o i-ésimo indivíduo de uma população de tamanho λ

Finalmente, para criar os indivíduos que compõem a população da próxima geração é utilizada a Equação 12.

$$\overrightarrow{x} \simeq \mathcal{N}(\overrightarrow{m},\sigma^2\mathbf{C}) \tag{12}$$

3 METODOLOGIA

Neste trabalho o PyPolyBuilder [15] é o programa utilizado para gerar os dendrímeros. O PyPolyBuilder e suas principais características são abordados na Seção 3.1. As primeiras contribuições deste trabalho são voltadas a otimização da formação da matriz-Z utilizada pelo PyPolyBuilder. Esta etapa era o gargalo computacional do programa e após um longo estudo das características do programa foram propostos novos métodos que acelerassem esse processo.

Posteriormente foram propostas algumas mudanças no intervalo dos ângulos utilizados na otimização global do PyPolyBuilder. Além disso foi proposta uma nova função objetivo que possibilita controlar melhor as características do dendrímero.

Finalmente, alguns dos principais algoritmos de otimização são aplicados com a intenção de obter o melhor dendrímero possível. Os algoritmos utilizados são: GA, DE, PSO e CMA-ES. Cada um dos algoritmos tem seus métodos e parâmetros ajustados para a geração zero do PAMAM. A geração zero foi escolhida para o ajuste dos métodos e parâmetros pois é a que demanda o menor custo computacional. Na sequência as dez melhores seleções de parâmetros e métodos, de cada algoritmo, são testados para as gerações de zero a três para se decidir qual a melhor seleção de parâmetros para o algoritmo. Então, as seleções escolhidas são simuladas para as gerações de zero a cinco, sete e dez para verificar o melhor algoritmo a ser aplicado neste problema.

3.1 PyPolyBuilder

O PyPolyBuilder é uma ferramenta implementada utilizando a linguagem de programação Python, que tem como objetivo construir polímeros radiais e dendrímeros de maneira ágil e possibilitando o controle de suas características. Para isso, o PyPolyBuilder se divide em quatro etapas principais: a primeira é a formação da topologia da molécula, a segunda é a geração de uma matriz-z, a terceira uma otimização global e, finalmente, uma última etapa onde é realizada uma otimização local. Os conceitos de otimização global e otimização local são abordados de maneira mais detalhada no Capítulo 2. As Seções 3.1.2 e 3.1.3 são mais detalhadas pois são onde ocorreram as maiores contribuições deste trabalho.

3.1.1 Topologia da molécula

Para a criação da topologia da molécula, inicialmente, os BBs fornecidos pelo usuário são lidos pelo PyPolyBuilder. Em seguida, os BBs são conectados utilizando-se a síntese divergente, respeitando o número de gerações fornecido pelo usuário. Posteriormente, são criadas as listas de átomo, ligações, ângulos e diedros.

3.1.2 Matriz-Z

A matriz-Z é uma matriz formada a partir das posições cartesianas dos átomos de uma molécula e armazena as posições dos átomos em relação aos demais. Na primeira coluna, é armazenado o tipo do átomo que está sendo observado. Na segunda, é armazenado o índice do átomo ao qual o observado está ligado. Na terceira, é armazenada a distância entre os átomos ligados. Na quarta, é armazenado o índice do átomo com o qual o observado e o que ele está ligado formam um ângulo. Na quinta, é armazenado o valor do ângulo formado. Na sexta, é armazenado o índice do átomo que forma um diedro com os três átomos citados anteriormente. Na sétima, é armazenado um ângulo de torção entre os átomos que formam o diedro [21]. Nesse trabalho, as distâncias das ligações e os valores dos ângulos são fixos. Além disso, algumas torções são conhecidas como impróprias e também tem um valor fixo. Sendo assim, os valores a serem otimizados nas próximas etapas são as torções que não são impróprias.

Na Tabela 5 é apresentada a matriz-Z de uma molécula de Flúor-Cloro-Bromo-Metano, que é representada na Figura 23. Esta molécula é utilizada para exemplificar como é gerada uma matriz-Z.

Na segunda linha da Tabela 5 apenas a primeira coluna é preenchida com o primeiro átomo da molécula. Já na terceira linha, as três primeiras colunas são preenchidas, sendo a primeira coluna preenchida com o segundo átomo da molécula, a segunda coluna preenchida com o índice do átomo ao qual ele é ligado e a terceira coluna com o valor da ligação. Na quarta linha, as três primeiras colunas são preenchidas da mesma maneira que na terceira linha. Já suas quarta e quinta colunas são preenchidas com o índice do átomo com o qual o terceiro átomo forma um ângulo e o valor do ângulo, respectivamente. A quinta linha tem suas cinco primeiras colunas preenchidas da mesma maneira que a quarta linha e, nas duas últimas, com o índice do átomo com o qual forma a torção e o

valor da torção. E, finalmente, a última linha é preenchida de maneira análoga a quinta.

Tabela 5 - Matriz-Z do Flúor-Cloro-Bromo-Metano.

Átomo	Ligação	Valor da Ligação	Ângulo	Valor do Ângulo	Torção	Valor da Torção
C						
H	1	1,0				
F	1	1,3	2	109,5		
Cl	1	1,7	2	109,5	3	120
Br	1	1,9	2	109,5	3	240

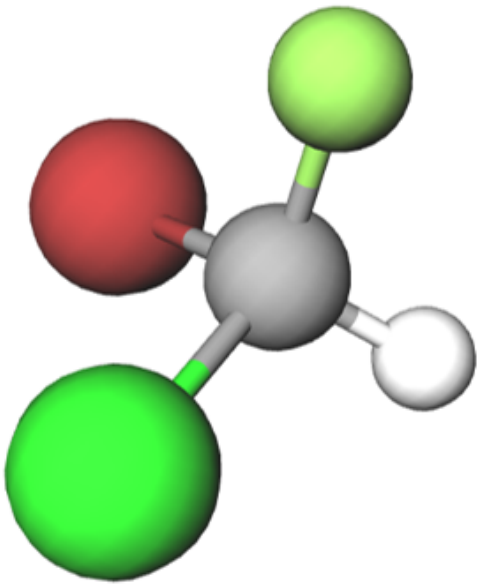

Figura 23 - Molécula de Flúor-Cloro-Bromo-Metano utilizada para exemplificar a maneira como a Matriz-Z é gerada.

A matriz-Z originalmente estava implementada de maneira que era necessária a realização de quatro laços, cada um interno ao anterior, para ser preenchida. O primeiro laço varria todos os átomos para completar a primeira coluna da matriz. Quando o primeiro laço do método estava observando o i-ésimo átomo da molécula, ele buscava do primeiro átomo ao átomo com índice "i-1" testando se cada um deles estava ligado ao i-ésimo átomo. Uma vez que um átomo que formasse uma ligação com o i-ésimo fosse encontrado, se iniciava um segundo laços, novamente buscando do primeiro átomo

àquele com índice “i-1”, e testando se cada átomo formava um ângulo com os dois que formavam uma ligação. E finalmente era realizado um último laço, procurando um átomo que formasse uma torção com os três que formavam um ângulo.

Como contribuição dessa dissertação, para otimizar a geração da matriz-Z, foi proposta outra abordagem; nela busca-se o átomo observado na lista de ligações para saber a qual átomo o observado está ligado e então preencher as colunas dois e três com o índice do átomo ligado e a distância da ligação. Depois, na lista de ligações, é procurado o átomo ligado ao observado para encontrar o átomo que forma um ângulo com os ligados. Uma vez que o átomo que forma o ângulo é localizado, a quarta e a quinta colunas são preenchidas com o índice do átomo que forma o ângulo e o valor do ângulo. E, finalmente, é buscado na lista de ligações o átomo ao qual aquele que forma o ângulo está ligado e, quando localizado, a sexta e a sétima colunas são preenchidas com o índice do átomo e o valor da torção. Como esse método realiza a busca na lista de ligações e não na lista de átomos isso reduz o tamanho do laço tornando o método mais suficiente. Como a lista de ligações já é preparada na criação da topologia da molécula o método não exige uma demanda computacional para isso e aproveita melhor os recursos produzidos na etapa anterior.

Posteriormente, e conjuntamente com os outros membros do grupo de pesquisa e desenvolvimento do PyPolyBuilder, foi desenvolvido um novo método, baseado no elaborado inicialmente nesta dissertação, onde o conceito de se utilizar a lista de ligações foi expandido para também utilizar as listas de ângulos e de torções. Sendo assim, as três primeiras colunas são preenchidas da mesma maneira que no método anterior. Porém, para preencher as colunas quatro e cinco da matriz, o átomo que forma o ângulo é procurado na lista de ângulos e o átomo que forma a torção é localizado na lista de diedros para preencher a sexta e a sétima coluna. Sendo assim, os recursos gerados na etapa anterior são ainda melhor utilizados. As comparações dos métodos apresentados são mostradas no Capítulo 4.

3.1.3 Otimização Global

Uma vez que a matriz-Z está completa, é inicializada uma otimização global para se encontrar as melhores torções para a molécula. Inicialmente o programa tinha como objetivo aumentar o raio da molécula. Para aumentar o raio da molécula são utilizadas

duas equações, onde a equação 13 serve para calcular a posição do centro da molécula. As posições cartesianas dos átomos são calculadas e armazenadas em 3 vetores, uma para a posição no eixo “x”, outro para a posição no eixo “y” e um último para a posição no eixo “z”. Então é calculada a soma dos valores de cada vetor, que é dividida pela quantidade de átomos na molécula, gerando assim um vetor com três posições que representam a posição cartesiana do centro da molécula.

$$centro = \frac{\sum_{i=1}^{atom}[x_i,y_i,z_i]}{atom} \tag{13}$$

Onde:

- *centro* é o vetor que representa a posição do centro da molécula.
- i é o índice do indivíduo.
- *atom* é a quantidade de átomos na molécula.
- x_i é a posição do indivíduo “i” no eixo “x”.
- y_i é a posição do indivíduo “i” no eixo “y”.
- z_i é a posição do indivíduo “i” no eixo “z”.

Uma vez que o centro do átomo é calculado pela equação 13, a equação 14 é utilizada para calcular o raio médio da molécula. Nela é calculada a distância euclidiana entre os átomos e o centro da molécula e depois esse valor é dividido pelo número de átomos para calcular o raio médio da molécula.

$$raio = \frac{\sum_{i=1}^{atom}\sqrt{(x_i - centro_x)^2 + (y_i - centro_y)^2 + (z_i - centro_z)^2}}{atom} \tag{14}$$

Onde:

- *raio* é o raio médio da molécula.
- $centro_x$ é a componente em “x” do centro da molécula.
- $centro_y$ é a componente em “y” do centro da molécula.
- $centro_z$ é a componente em “z” do centro da molécula.

Observou-se que os indivíduos gerados apresentavam um formato aproximadamente linear. Segundo o especialista, que faz parte do grupo de estudo do PyPolyBuilder, o esperado era que a molécula ao final da evolução fosse aproximadamente esférica. Nesse trabalho foi observado que esse comportamento decorre da função de aptidão utilizada. Então, foi proposta uma nova função de aptidão que é representada pela equação 15. Nessa função, são buscados indivíduos com as maiores variações de tamanho nos três eixos.

$$funcaoObjetivo = ((x_{max} - x_{min}) + (y_{max} - y_{min}) + (z_{max} - z_{min})) \tag{15}$$

Onde:

- $funcaoObjetivo$ é a função objetivo do algoritmo.
- x_{max} é o maior valor de "x" em algum átomo da molécula.
- x_{min} é o menor valor de "x" em algum átomo da molécula.
- y_{max} é o maior valor de "y" em algum átomo da molécula.
- y_{min} é o menor valor de "y" em algum átomo da molécula.
- z_{max} é o maior valor de "z" em algum átomo da molécula.
- z_{min} é o menor valor de "z" em algum átomo da molécula.

Essa função de aptidão gera indivíduos mais esféricos que a função do raio médio. Porém, ela também não garante que todos os ramos tenham um comportamento esférico, apenas os ramos com as maiores ou menores coordenadas. Então, posteriormente, conjuntamente com os outros membros do grupo de pesquisa e desenvolvimento do PyPolyBuilder, foi desenvolvido o método que calcula a esfericidade. Ele calcula o centro da molécula usando a equação 13. Em seguida, calcula-se a esfericidade pela equação 16. E, finalmente, utiliza a equação 17, para calcular a função de aptidão do método levando em consideração tanto o tamanho da molécula quanto a sua esfericidade.

$$esfericidade = 10\frac{\sqrt[3]{rX \times rY \times rZ}}{\max\left(\frac{rX}{Wx}, \frac{rY}{Wy}, \frac{rZ}{Wz}\right)} \tag{16}$$

Onde:

- rX é o desvio padrão entre os valores de “x”.
- rY é o desvio padrão entre os valores de “y”.
- rZ é o desvio padrão entre os valores de “z”.
- W é um vetor que controla a forma da molécula ponderando os eixos. Neste trabalho o vetor $W = [1; 1; 1]$, essa seleção força a molécula a ter uma forma aproximadamente esférica. Caso o vetor fosse configurado com dois valores pequenos e um maior, por exemplo $W = [0{,}1; 0{,}1; 2{,}8]$, a molécula teria um formato aproximadamente linear e caso fosse configurado com um valor menor e dois maiores, por exemplo $W = [0{,}2; 1{,}4; 1{,}4]$ a molécula teria um formato de disco.

$$funcObj = \frac{\sum_{i=1}^{atom} \sqrt{(x_i - centro_x)^2 + (y_i - centro_y)^2 + (z_i - centro_z)^2}}{atom}\ esfericidade \tag{17}$$

Os algoritmos utilizados para a otimização global nesse trabalho são: GA, DE, CMA-ES e PSO. Esses algoritmos são apresentados de maneira detalhada no Capítulo 2 e na Seção 3.2 é mostrado como esses algoritmos são ajustados para o problema.

3.1.4 Otimização Local

Após a conclusão da otimização global, que usa a esfericidade e o raio da molécula como função objetivo, é iniciada a otimização local, que tem como objetivo controlar as características da molécula de acordo com a sua energia. O tipo de otimização utilizada é a descida do gradiente. Como a descida do gradiente se baseia em derivadas para reduzir a complexidade do problema, o algoritmo atua sobre a força que é a derivada da energia. Assim, é evitado um aumento da complexidade computacional por não haver a necessidade de se realizar diversas derivadas durante a execução. Uma vez que essa otimização termina, são gerados os arquivos que posteriormente podem ser lidos pelo GROMOS ou pelo GROMACS.

3.2 Ajustes de parâmetros dos algoritmos de otimização

Após a otimização da matriz-Z, é iniciada a etapa de ajuste dos algoritmos utilizados na otimização global. O primeiro algoritmo sugerido foi um GA utilizando a biblioteca DEAP. O algoritmo utiliza dois tipos de seleção: por Roleta e por Torneio. A seleção por Torneio tem um parâmetro que precisa ser ajustado. Além disso, o algoritmo utiliza quatro tipos de cruzamento: de Um Ponto, de Dois Pontos, Uniforme e por Combinação. Os métodos Uniforme e por Combinação têm um parâmetro cada a ser ajustado. E, finalmente, o algoritmo utiliza três tipos de mutação: Gaussiana, Uniforme e por Mudança de Índice. A mutação Gaussiana precisa ajustar dois parâmetros e a mutação Uniforme, um parâmetro. Além desses, para todos os métodos, o algoritmo utiliza uma taxa de cruzamento e uma taxa de mutação que determinam a probabilidade de um par de indivíduos realizar o cruzamento e dos indivíduos filhos sofrerem mutação.

Com isso, temos que selecionar três métodos dentre nove, um para seleção, outro para o cruzamento e outro para a mutação. Além disso, é necessário ajustar diversos parâmetros, com sua quantidade variando de acordo com os métodos selecionados, podendo variar de dois a seis dependendo dos métodos utilizados. Cada um dos outros algoritmos selecionados tem seus próprios métodos e parâmetros a serem ajustados.

Como os algoritmos são estocásticos, vão ocorrer resultados diferentes para uma mesma seleção de métodos e parâmetros. Para garantir que seja obtida a melhor solução para cada algoritmo, é calculada a média entre cem simulações com a mesma seleção de métodos e parâmetros. Esse é considerado o valor de aptidão do algoritmo para a seleção de parâmetros. Essas simulações são realizadas na geração zero por motivo de simplicidade.

Um dos principais problemas em computação evolutiva é o ajuste de parâmetros, uma vez que esses algoritmos tem uma grande variedade de métodos e parâmetros relacionados à sua execução. Existem diversas abordagens para o ajuste dos mesmos, que vão desde realizar todas as combinações de parâmetros possíveis, até tentativas de seleções de parâmetros aleatoriamente. Passando, inclusive, por algoritmos evolutivos para localizar a melhor seleção de parâmetros [118].

É importante ressaltar que a escolha desses parâmetros determina a qualidade da solução obtida pelo algoritmo [119]. Neste trabalho, os parâmetros relativos à quantidade de gerações utilizadas pelo algoritmo e ao tamanho da população não são otimizados, pois

são fornecidos pelo PyPolyBuilder. Então, para uma comparação mais justa, esses valores são mantidos fixos para todos os algoritmos.

Em [118], foram testadas diversas maneiras para obter a melhor seleção de parâmetros para um algoritmo. Com isso, observou-se que a busca aleatória é uma boa maneira para ajustá-los, apresentando um bom equilíbrio entre complexidade computacional e diversidade nos espaços de busca. Neste trabalho, a busca aleatória foi a maneira utilizada para selecionar os parâmetros utilizados por cada um dos algoritmos.

Antes dos parâmetros dos algoritmos serem ajustados foi analisado o comportamento esperado das torções para gerar uma molécula com as características esperadas, ou seja, o maior e mais esférica possível. Nessa analise supôs-se que as torções deveriam estar no semi-plano formado para torções entre 90 e 270 graus pois ângulos fora desse intervalo facilitariam a ocorrência de enovelamento na molécula. O efeito dessa mudança no intervalo de ângulos na aptidão foi simulado 100 vezes para o algoritmo utilizado originalmente no PyPolyBuilder nas gerações de zero a três e é apresentado na Seção 4.2.

Para se selecionar os melhores parâmetros e métodos para cada algoritmo, inicialmente são gerados parâmetros de maneira pseudoaleatória para cada método ou para cada casamento de métodos. Essa seleção de métodos e parâmetros é simulada 100 vezes para a geração zero do dendrímero, donde a média das aptidões nas 100 simulações é considerada a aptidão para a seleção. No primeiro momento, os parâmetros podem se encontrar em qualquer ponto do espaço de busca e, portanto, são realizadas diversas simulações iniciais visando a reduzi-lo.

Quando se fala sobre casamento de métodos, refere-se a algoritmos que utilizam mais de um método, como, por exemplo, o GA, que precisa de um método para a seleção, outro para o cruzamento e mais um para a mutação. Nesse caso a mesma seleção de parâmetros será testada para todas as possíveis combinações de métodos.

Após diversas simulações, os resultados são observados. Essas diversas simulações são chamadas neste trabalho de baterias de simulações. A primeira maneira de se observar os dados é em relação aos métodos, verificando-se quantas vezes cada método obtém a melhor aptidão para uma determinada seleção de parâmetros. Se algum método obtém os melhores resultados com uma frequência muito baixa, ele não é utilizado nas próximas baterias de simulações. Além dessa, outra maneira de observar os dados é comparando os parâmetros com as aptidões. O objetivo dessa observação é diminuir o domínio de busca

de cada parâmetro. Para isso, os intervalos do domínio onde não ocorrem resultados bons são excluídos. Para alguns parâmetros não é possível observar num primeiro momento um intervalo do domínio com resultados piores ao ponto de poder ser excluído. Para poder observar, nesses casos, é gerado um gráfico observando apenas um percentual dos melhores indivíduos. Caso alguma parte do domínio não apareça nessa observação, ele pode ser excluído pois não apresenta bons resultados. Após essas observações, o procedimento é repetido excluindo-se os métodos que não obtiveram bons resultados e restringindo-se os domínios dos parâmetros que são possíveis. As baterias de simulação têm tamanhos bem diversificados para os algoritmos e isso se dá por dois motivos principais: o primeiro deles é que cada algoritmo tem uma quantidade diferente de métodos para se testar os parâmetros. Então, o tamanho da bateria de simulações é variado para se tentar manter a mesma ordem de grandeza das simulações feitas em uma bateria de outro algoritmo; o outro motivo é que cada bateria é realizada até que seja possível se restringir algum dos parâmetros ou excluir algum dos métodos.

Esse procedimento é repetido até que as aptidões obtidas em uma bateria de simulação tenham valores próximos. Quando esse objetivo é atingido, as dez melhores seleções de métodos e parâmetros, para um determinado algoritmo, são simuladas 100 vezes para as gerações de zero a três de forma a se determinar a melhor seleção para o algoritmo. Finalmente, após esse procedimento ser realizado para todos os algoritmos, a melhor seleção de cada algoritmo é simulada para as gerações de zero a cinco, sete e dez. Os resultados obtidos são comparados para se verificar as vantagens de cada algoritmo. Esses resultados obtidos são apresentados na Seção 4.2.

Como citado anteriormente, cada algoritmo tem seus métodos e parâmetros a serem ajustados. O GA tem dois métodos para seleção, sendo eles a Roleta e o Torneio, onde este último apresenta um parâmetro a ser ajustado, o qual define o número de indivíduos que participam de cada torneio. Além disso, o GA tem quatro métodos para o cruzamento, sendo eles o cruzamento de Dois Pontos, o cruzamento de Um Ponto, o cruzamento Uniforme e o cruzamento Por Combinação. Os cruzamentos de Um Ponto e de Dois Pontos não possuem parâmetros internos, já o cruzamento Uniforme e o Por Combinação possuem um parâmetro interno cada. Esse parâmetro é aqui chamado genericamente de taxa de cruzamento interno e, no cruzamento uniforme, representa o valor que é utilizado para se definir se os genes dos indivíduos pais vão ser trocados. Já no cruzamento Por

Combinação, esse parâmetro representa o percentual que os indivíduos filhos vão receber dos indivíduos pais. Finalmente, o GA possui três tipos de mutação: Gaussiana, Uniforme e Por Mudança de Índice. As mutações Uniforme e Por Mudança de Índice tem um parâmetro interno cada e, para esta última mutação, o parâmetro controla se o gene muda de posição ou não. Já na Uniforme o parâmetro controla a chance do gene ser modificado. Já a mutação Gaussiana tem dois parâmetros para serem atualizados. Um controla a probabilidade do gene ser mutado enquanto o outro controla o desvio padrão da mutação. Os parâmetros utilizados nas mutações Por Mudança de Índice, na Uniforme, e o primeiro utilizado na Gaussiana são chamados genericamente de probabilidade interna de mutação. Por sua vez, o segundo parâmetro utilizado na mutação Gaussiana é chamado de σ. Além desses parâmetros, o algoritmo possui dois outros: a probabilidade de cruzamento e a probabilidade de mutação. Esses parâmetros controlam a probabilidade de um indivíduo sofrer cruzamento e mutação, respectivamente.

A DE tem sete métodos e todos eles funcionam como cruzamento. As equações que representam esses métodos são mostradas na Tabela 4. Todos os métodos da DE ponderam os indivíduos pais para gerar os indivíduos filhos. Os métodos Aleatório 1 e Aleatório 2, bem como os Com o Melhor Indivíduo 1 e o Com o Melhor Indivíduo 2 só precisam ajustar um parâmetro, sendo ele, o F. Já os métodos aleatório com o melhor, atual com um aleatório e atual com o melhor precisam ajustar dois parâmetros, sendo eles o F e o λ. Estes parâmetros variam entre zero e um e ponderam como os indivíduos pais geram os filhos. A maneira na qual essa ponderação acontece para cada método é mostrada na Tabela 4. Além disso, todos os métodos possuem um parâmetro chamado Taxa de Cruzamento, que controla a chance do indivíduo pai, que está sendo observado, ser selecionado para realizar o cruzamento.

Neste trabalho, o PSO dispensa a necessidade de se ajustar os métodos, pois possui apenas um. Este algoritmo utiliza seis parâmetros: P_{min} que representa a posição mínima de uma partícula e está entre 90 e 180, P_{max} que representa a posição máxima de uma partícula e está entre 180 e 270, S_{min} que representa a velocidade mínima de uma partícula e está entre 90 e 180, S_{max} que representa a velocidade máxima de uma partícula e está entre 180 e 270; $Phi1$ controla o coeficiente para o controle da componente social e está entre um e o comprimento da partícula, enquanto que $Phi2$ controla o coeficiente para o controle da componente cognitiva e está entre um e o comprimento da partícula (o

comprimento da partícula é equivalente ao número de genes em um cromossomo no GA).

A CMA-ES é um algoritmo que utiliza três métodos, sendo eles: Igual, Linear e Superlinear. Além disso, o algoritmo otimiza seis parâmetros, sendo eles: centroide, que é o ponto onde a evolução é iniciada, como o intervalo de ângulos para os demais algoritmos é entre 90 e 270 esse intervalo é utilizado para o centroide; σ, que é o desvio padrão inicial, o intervalo selecionado para o σ é de um a 180; μ, que é o número de indivíduos pais que são mantidos para a próxima geração, o seu intervalo de valores é entre 1 e o tamanho da população; *cs*, que é constante de acumulação para o tamanho do passo e o seu intervalo de valores é entre zero e um; *damps*, que é o amortecimento para passo e o seu intervalo de busca é de 0 a 5; e *ccum* que é a constante de acumulação para matriz de covariância e seu intervalo de busca é entre zero e 0,8. O *cs* é um valor baseado na variância efetiva da seleção e no tamanho do indivíduo, o *damps* é calculado em relação a constante de acumulação para matriz de covariância e o *ccum* é um valor referente ao tamanho do indivíduo. Os intervalos são determinados em relação aos pontos críticos possíveis para a variância efetiva da seleção e o tamanho do indivíduo.

Uma vez que os algoritmos estejam com seus métodos e parâmetros ajustados, o teste-t é utilizado para verificar a probabilidade da diferença entre os algoritmos ocorrer de maneira aleatória. O teste-t é utilizado para avaliar se há diferença de significância estatística entre as médias de duas amostras. Para verificar a possibilidade da utilização do teste-t, é realizado anteriormente o teste de Levene. O teste-t utilizado neste trabalho está implementado na biblioteca Python chamada “SciPy”. Para se realizar o teste, os algoritmos são executados 100 vezes e as listas de aptidões geradas para cada um são comparadas duas a duas.

4 RESULTADOS

Neste capítulo, são mostrados os resultados obtidos neste trabalho. Inicialmente, são apresentados os resultados obtidos na matriz-Z e, posteriormente, são apresentados os resultados obtidos pelos métodos de otimização global.

4.1 Resultados Matriz-Z

Durante a realização deste projeto, foram desenvolvidos dois métodos alternativos, um que se baseia na lista de ligações para formar a matriz-Z e outro que utiliza as listas de ligações, de ângulos e de torções. Na Tabela 6, são apresentados os tempos, em segundos, que cada um dos métodos leva para gerar a matriz-Z para uma determinada geração. Estes tempos foram obtidos pela média de 100 vezes que a matriz-Z foi gerada. A primeira coluna mostra a geração do dendrímero para a qual a matriz-Z é criada. As demais colunas mostram os tempos que cada um dos três métodos levou para criar as matrizes, sendo eles, o original, com uma lista e com várias listas.

Tabela 6 - Tempo para os diferentes métodos gerarem a matriz-Z para as gerações de zero a cinco, sete e dez.

	Original	Uma lista	Várias Listas
Geração 0	0,224	0,005	0,012
Geração 1	4,667	0,032	0,052
Geração 2	60,785	0,174	0,139
Geração 3	520,375	0,743	0,510
Geração 4	6.634,392	2,974	1,865
Geração 5	-	12,476	7,640
Geração 7	-	206,82	128,00
Geração 10	-	13.865,04	8.101,30

O método original só foi simulado até a quarta geração, pois os seus resultados são visivelmente piores que os outros métodos. Como o método com várias listas é o mais rápido e não tem nenhuma restrição, ele é considerado o melhor método observado nesse trabalho. Esses métodos são detalhados na Seção Matriz-Z (3.1.2).

4.2 **Resultados da Otimização Global**

A primeira mudança sugerida foi na definição do domínio dos ângulos de torção. Originalmente, o domínio era entre zero e 360 graus. Como um dos objetivos do programa é aumentar o raio da molécula, supôs-se que os melhores ângulos estariam nos arredores de 180 graus. Para manter os ângulos mais próximos de 180 graus sem perder diversidade na população, propôs-se utilizar um domínio entre 90 e 270 graus. Para testar essa hipótese, o GA original foi testado cem vezes para ambos os domínios e para as gerações de zero a três.

Na Tabela 7, são mostrados os resultados destas simulações. Na primeira coluna, estão as gerações que estão sendo simuladas. Na segunda, é mostrado o valor da aptidão para o domínio de zero a 360. Na terceira, é mostrado o valor da aptidão para o e intervalo entre 90 e 270. Na última, é mostrado o aumento percentual do raio em função do intervalo. Como o aumento percentual cresce a cada geração, para os próximos algoritmos e para a avaliação do GA original, foi utilizado o domínio de noventa a duzentos e setenta.

Tabela 7 - Comparação das aptidões dos indivíduos gerados para os diferentes domínios.

	0 a 360	90 a 270	Aumento percentual
geração 0	5,33	6,26	17%
geração 1	7,76	11,12	43%
geração 2	9,95	15,90	60%
geração 3	11,56	20,69	79%

4.2.1 GA

Como citado na Seção 3.2, o GA tem dois métodos para a seleção, quatro tipos de cruzamento e três tipos de mutação. Para otimizar os parâmetros e ajustar os melhores métodos, a primeira solução utilizada foi selecionar os parâmetros de maneira aleatória e, para cada seleção dos mesmos, utilizar todas as combinações possíveis de métodos para calcular as aptidões. Na primeira bateria de simulações, foram testadas 160 seleções aleatórias de parâmetros. Na Tabela 8, é mostrada a quantidade de vezes que um determinado casamento de métodos teve a melhor aptidão para os parâmetros escolhidos de maneira aleatória - lembrando que a aptidão é obtida pela média de cem simulações com os mesmos parâmetros e métodos. Na primeira coluna da Tabela 8, são mostradas as

seleções utilizadas na combinação de métodos; na segunda, são mostrados os cruzamentos utilizados; na terceira é mostrada a quantidade de vezes que a mutação Gaussiana obtém a melhor aptidão para uma determinada seleção de parâmetros; na quarta é mostrada a quantidade de vezes que a mutação uniforme obteve a melhor aptidão para uma determinada seleção de parâmetros; e na última, é mostrada a quantidade de vezes que a mutação por mudança de índices obteve a melhor aptidão para uma determinada seleção de parâmetros.

Tabela 8 - Quantidade de vezes em que as melhores aptidões são escolhidas para as determinadas combinações de métodos do GA.

Seleção	Cruzamento	Mutação Gaussiana	Mutação Uniforme	Mutação Por Índices
Roleta	Um Ponto	37	5	6
Roleta	Dois Pontos	33	5	3
Roleta	Uniforme	27	2	2
Roleta	Por Combinação	9	2	1
Torneio	Um Ponto	3	1	1
Torneio	Dois Pontos	11	0	2
Torneio	Uniforme	3	1	1
Torneio	Por Combinação	4	0	1

Após essa primeira bateria de simulações, observou-se que em aproximadamente 80% das simulações, a melhor combinação de métodos ocorre quando a mutação selecionada é a Gaussiana. Por esse motivo, nas próximas baterias de simulações, só será utilizada a mutação Gaussiana.

A Tabela 9 representa os resultados da segunda bateria de simulações. Nela, são mostradas as melhores aptidões encontradas em 360 seleções de parâmetros. Esta quantidade de seleções de parâmetros foi escolhida pois a partir desse ponto, e através de experimentos preliminares, foi possível observar um padrão de comportamento quando relacionados os métodos utilizados e as aptidões obtidas. Na primeira coluna, são mostrados os tipos de seleção utilizados, na segunda são mostradas as aptidões para o cruzamento por Um Ponto, na terceira as aptidões para o cruzamento por Dois Pontos, na quarta as aptidões para o cruzamento Uniforme e na quinta as aptidões para o cruzamento Por Combinação.

Tabela 9 - Quantidade de melhores aptidões para as combinações de métodos, com a mutação Gaussiana selecionada.

	Um Ponto	Dois Pontos	Uniforme	Por Combinação
Roleta	98	105	70	30
Torneio	12	19	11	15

Na Tabela 9, observou-se que aproximadamente 84% das melhores aptidões ocorrem quando a seleção escolhida é a seleção por Roleta. Por isso, nas próximas baterias, é a seleção utilizada. Também foi possível observar que o cruzamento Por Combinação é o tipo selecionado apenas em aproximadamente 12% das seleções; porém, como mostrado na Figura 24, observou-se que, quando todas as amostras obtidas como resultados para uma seleção de parâmetros o cruzamento por Combinação é o selecionado em apenas 12,5% das seleções. Porém quando apenas as melhores aptidões são consideradas, a porcentagem de aparições do cruzamento Por Combinação aumenta. Supõe-se que esse comportamento ocorra porque o cruzamento Por Combinação e Uniforme dependem do ajuste de um parâmetro enquanto os cruzamentos De Um Ponto e Dois Pontos não dependem de nenhum parâmetro. Por isso, na próxima bateria de simulações, são mantidos os quatro tipos de cruzamento, a seleção por Roleta e a mutação Gaussiana.

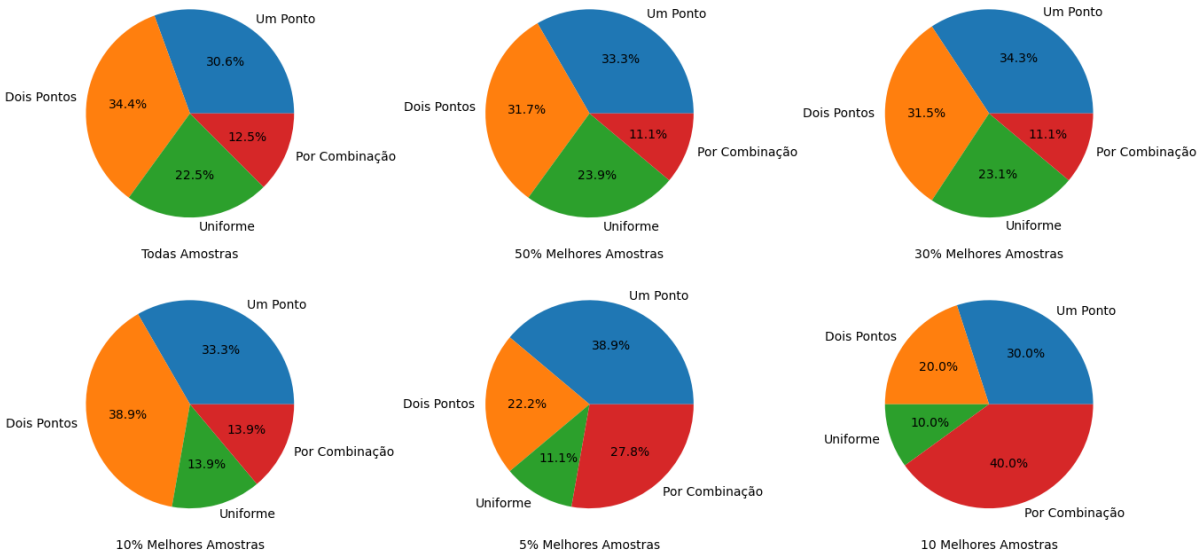

Figura 24 - Porcentagens dos tipos de cruzamentos do GA nas melhores aptidões com as seleções por Roleta e por Torneio. Cada gráfico representa um percentual das amostras obtidas como melhores para uma determinada seleção de parâmetros.

Com os métodos selecionados, os parâmetros começam a ser ajustados. Uma terceira bateria de simulações é então realizada com 500 seleções de parâmetros aleatórias sendo realizadas e testadas com todos os métodos de cruzamento. Na Figura 25, é mostrado o resultado da terceira bateria de simulações. Nela, cada ponto representa a aptidão de uma simulação: quanto mais próximo ao azul escuro é a cor do indivíduo, melhor é sua aptidão. Na Figura 25, observou-se que o único parâmetro que poderia ter seu domínio reduzido era a taxa de cruzamento interno, que deve ter seu valor entre 0,2 e 0,4.

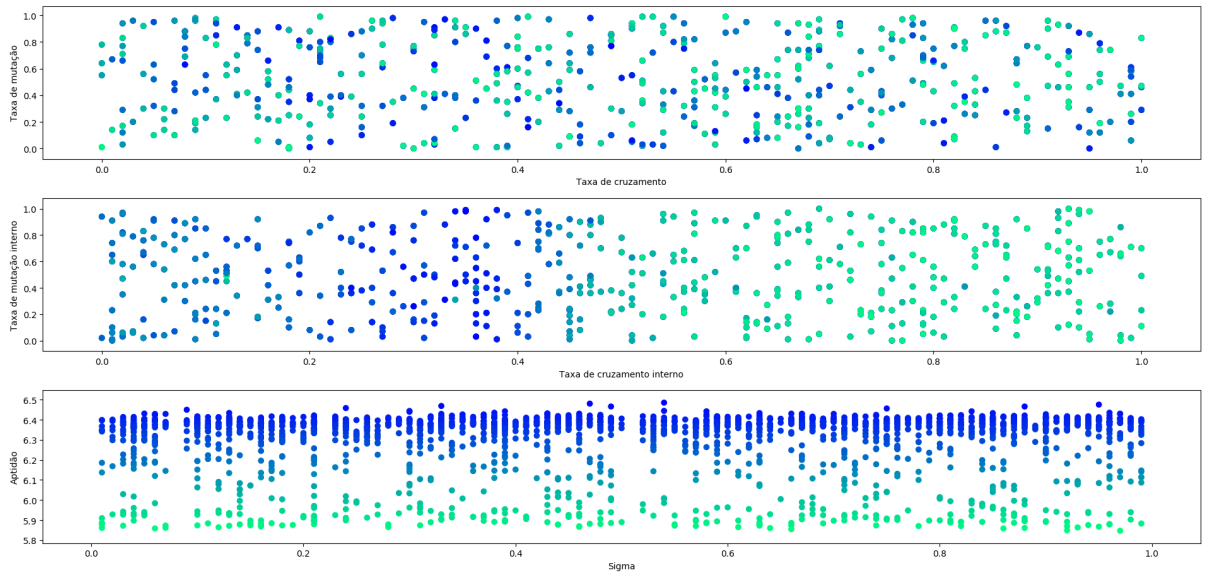

Figura 25 - Distribuição das aptidões de acordo com os parâmetros no GA.

Para se avaliar se algum dos tipos de cruzamento podem ser desconsiderados na próxima bateria de simulações, observou-se a quantidade de vezes que as melhores aptidões ocorrem para cada um dos cruzamentos utilizados. Esses dados podem ser observados na Figura 26. Pode-se observar um comportamento parecido com o mostrado na Figura 24, com as diferenças mais significativas sendo o cruzamento Uniforme aparecendo com uma frequência menor e o cruzamento de Dois Pontos não aparecendo no gráfico com os vinte melhores indivíduos.

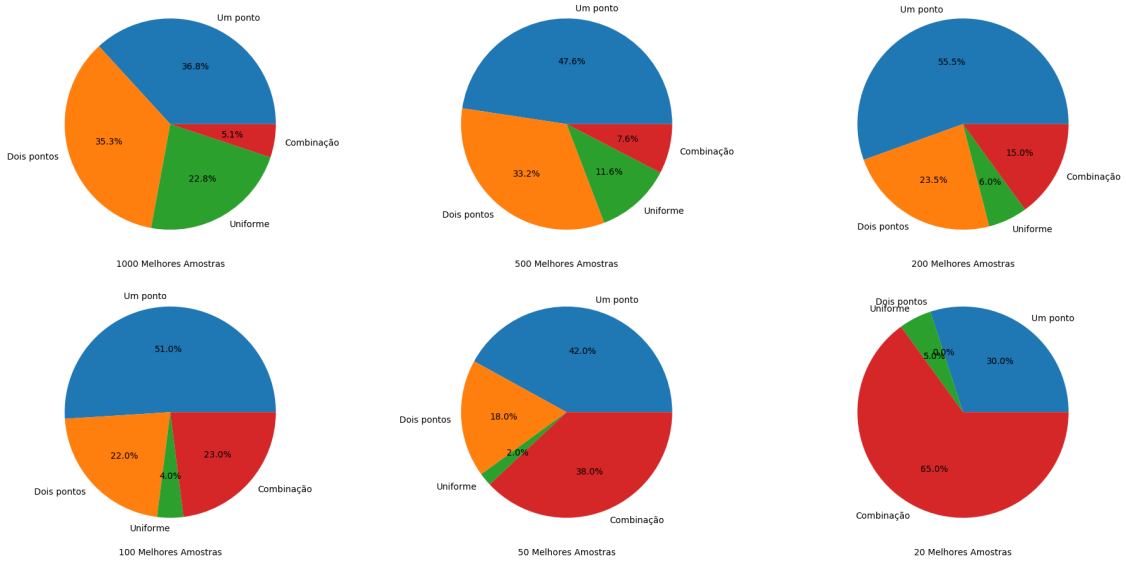

Figura 26 - Porcentagens dos tipos de cruzamentos no GA nas melhores aptidões.

Com as taxas de cruzamento interno limitadas entre 0,2 e 0,4, é realizada uma quarta bateria de simulações. Nessa bateria, são realizadas 500 seleções de parâmetros, para cada um dos quatro métodos com os parâmetros ajustados. Os resultados da quarta bateria de simulações são mostrados na Figura 27.

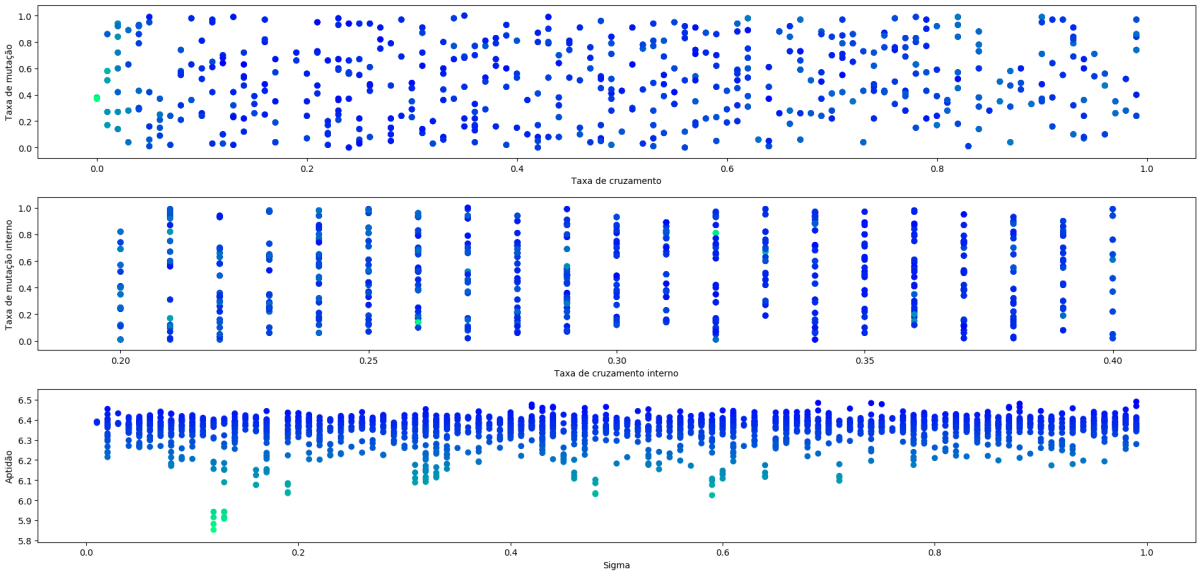

Figura 27 - Distribuição das aptidões em função dos parâmetros do GA com a taxa de cruzamento interno entre 0,2 e 0,4.

Com a distribuição apresentada na Figura 27, concluiu-se que o espaço de busca

para os parâmetros é o escolhido. O ajuste comparando as 10 melhores seleções obtidas na geração zero para as gerações de zero a três não foi realizado pois o algoritmo demandou grande quantidade de simulações quando comparado aos outros algoritmos. Essa maior quantidade ocorre pois o algoritmo utiliza um casamento de métodos e assim cada seleção de parâmetros precisa ser testada muitas vezes ao longo dos ajustes. Sendo assim, o indivíduo com a melhor aptidão utiliza a seleção por Roleta, a mutação Gaussiana e o cruzamento Por Combinação. Os parâmetros escolhidos foram: taxa de cruzamento 0,23, para determinar a probabilidade dos indivíduos selecionados realizarem o cruzamento; taxa de mutação 0,39, para determinar a probabilidade dos indivíduos sofrerem mutação; taxa de cruzamento interna é 0,3, para determinar a proporção que os indivíduos filhos recebem de cada pai durante o cruzamento por combinação; taxa de mutação interna é 0,26, para uma vez que um indivíduo é selecionado para sofrer mutação determinar a chance de cada um de seus genes serem mutados; e o sigma 0,54, utilizado na mutação Gaussiana para determinar o espaço ao redor do gene original onde o mutado pode se encontrar. O indivíduo que utiliza essa seleção de parâmetros e métodos tem, em média, uma aptidão de 6,4851.

4.2.2 DE

Neste trabalho, a DE utiliza sete tipos de cruzamento: dois cruzamentos Aleatórios e dois cruzamentos Com o Melhor indivíduos, ambos com um parâmetro F, um cruzamento Aleatório Com o Melhor indivíduo, um do indivíduo Atual Com Um Aleatório e um do Atual Com o Melhor indivíduo. Esses três últimos cruzamentos utilizam dois parâmetros, um F e um λ. Além destes parâmetros, uma taxa de cruzamento também é utilizada. A taxa de cruzamento determina a probabilidade do indivíduo selecionado sofrer cruzamento enquanto o F e o λ representam valores que vão ponderar os valores dos indivíduos escolhidos para a seleção, como apresentado na Tabela 4 que mostra como são realizados os cruzamentos na DE.

Uma primeira bateria de simulações é realizada gerando 128 seleções de parâmetros aleatórias e utilizando essas seleções nos sete cruzamentos. Na Tabela 10, é mostrada a quantidade de vezes que cada um dos cruzamentos é o melhor para uma determinada seleção de parâmetros.

Tabela 10 - Quantidade de melhores aptidões para cada método.

Cruzamento	Quantidade de melhores
Aleatório 1	0
Aleatório 2	0
Melhor 1	31
Melhor 2	1
Aleatório com o melhor	57
Atual com um Aleatório	7
Atual com o melhor	29

Como pode ser observado na Tabela 10, os cruzamentos Aleatórios, o cruzamento Melhor 2 e o cruzamento Atual com um Aleatório são os melhores para uma das seleções de parâmetros muito poucas vezes. Por esse motivo não serão utilizados nas próximas baterias de simulações.

Na segunda bateria de simulações, novamente, os parâmetros são escolhidos aleatoriamente - dessa vez, 715 vezes e testados nos três cruzamentos restantes, tendo assim um total de 2.145 amostras. Na Figura 28, são mostradas quantas vezes os cruzamentos aparecem entre as melhores aptidões.

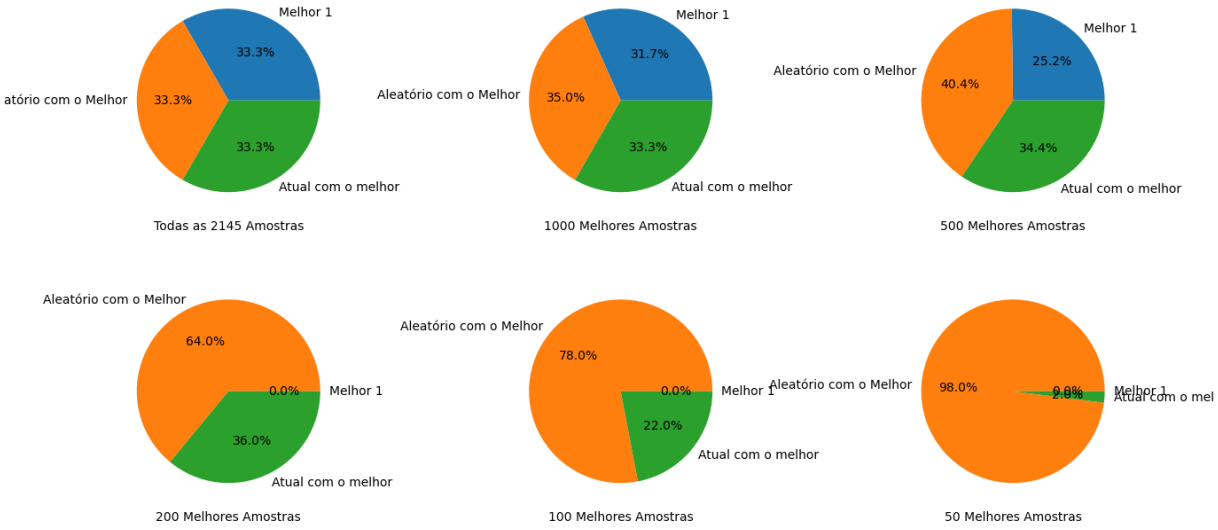

Figura 28 - Porcentagens das melhores aptidões nos tipos de cruzamentos para DE.

Na Figura 28, é possível observar que os melhores indivíduos sempre ocorrem

quando o cruzamento Aleatório Com o Melhor é selecionado. Por isso, na próxima bateria de simulações será o único tipo cruzamento utilizado. Além disso, na Figura 29, é apresentada a aptidão em função de cada um dos parâmetros.

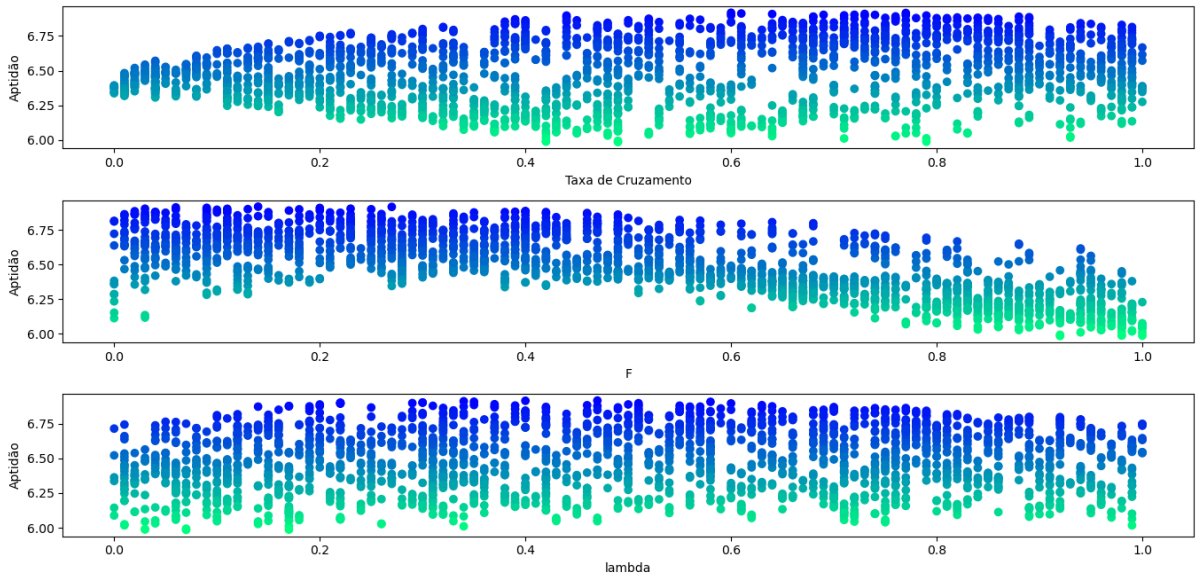

Figura 29 - Variação das aptidões em função dos parâmetros para DE.

Na Figura 29, observou-se que o domínio na próxima bateria de simulações pode ser reduzido para a taxa de cruzamento entre 0,6 e 0,9, F entre 0 e 0,3 e λ entre 0,1 e 0,6. A terceira bateria de simulações tem mil seleções de parâmetros e seus resultados são apresentados na Figura 30. Nela, pode-se observar que o espaço de busca selecionado abrange os melhores indivíduos sem perder diversidade.

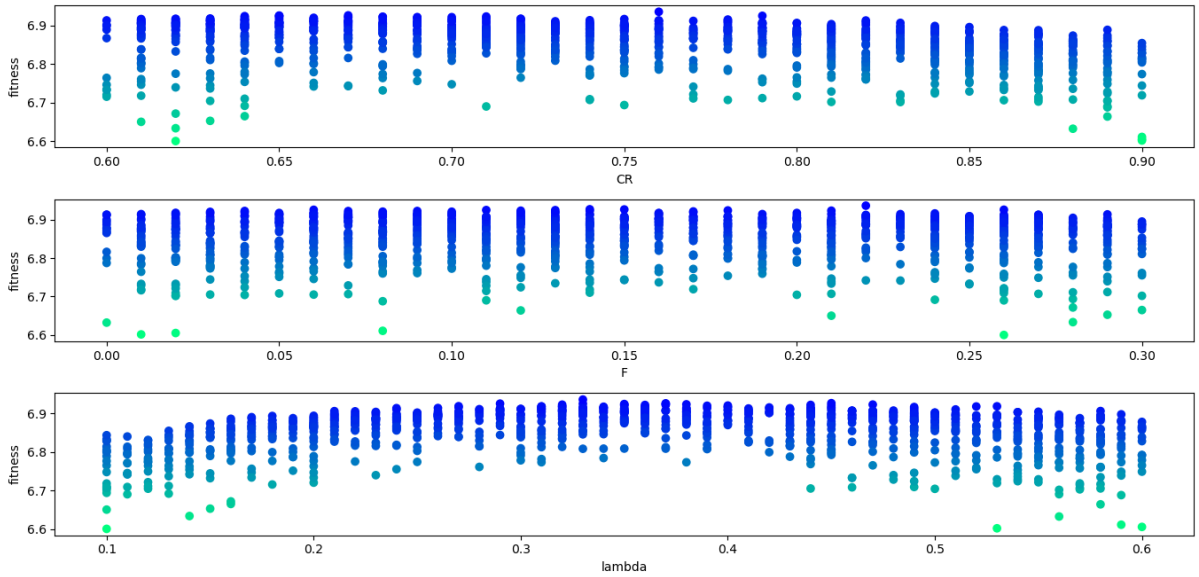

Figura 30 - Variação das aptidões em função dos parâmetros, com o espaço de busca reduzido para DE.

As dez seleções de parâmetros com as melhores aptidões, obtidas nas baterias de simulações, foram simulados para as gerações de zero a três do dendrímero. Os resultados obtidos são apresentados na Tabela 11. Com isso, o método utilizado para o cruzamento é o Aleatório com o Melhor e os parâmetros escolhidos foram: para a taxa de cruzamento, 0,64; para F, 0,11; e para λ, 0,37.

Tabela 11 - Resultados das dez melhores seleções de parâmetros para as gerações de zero a três.

Taxa de cruzamento	F	λ	Geração 0	Geração 1	Geração 2	Geração 3
0,76	0,22	0,33	6,910	12,601	18,716	25,025
0,67	0,14	0,45	6,911	12,585	18,759	25,058
0,65	0,15	0,37	6,919	12,606	18,778	25,053
0,79	0,26	0,29	6,902	12,547	18,716	24,841
0,65	0,06	0,33	6,920	12,607	18,833	25,159
0,64	0,11	0,37	6,926	12,632	18,826	25,229
0,65	0,13	0,35	6,920	12,597	18,725	25,082
0,71	0,18	0,38	6,918	12,606	18,765	25,131
0,67	0,21	0,47	6,886	12,566	18,697	24,959
0,68	0,12	0,35	6,907	12,592	18,809	25,101

4.2.3 PSO

Nessa Seção são mostrados os resultados obtidos para o PSO. Neste trabalho o PSO utiliza apenas um método e seis parâmetros, sendo eles: P_{min}, a posição mínima de uma partícula; P_{max}, a posição máxima da partícula; S_{min}, a velocidade mínima da partícula; S_{max}, a velocidade máxima da partícula; $Phi1$, coeficiente para o controle da componente social; $Phi2$, coeficiente para o controle da componente cognitiva. Os resultados obtidos nas 760 simulações da primeira bateria são apresentados na Figura 31.

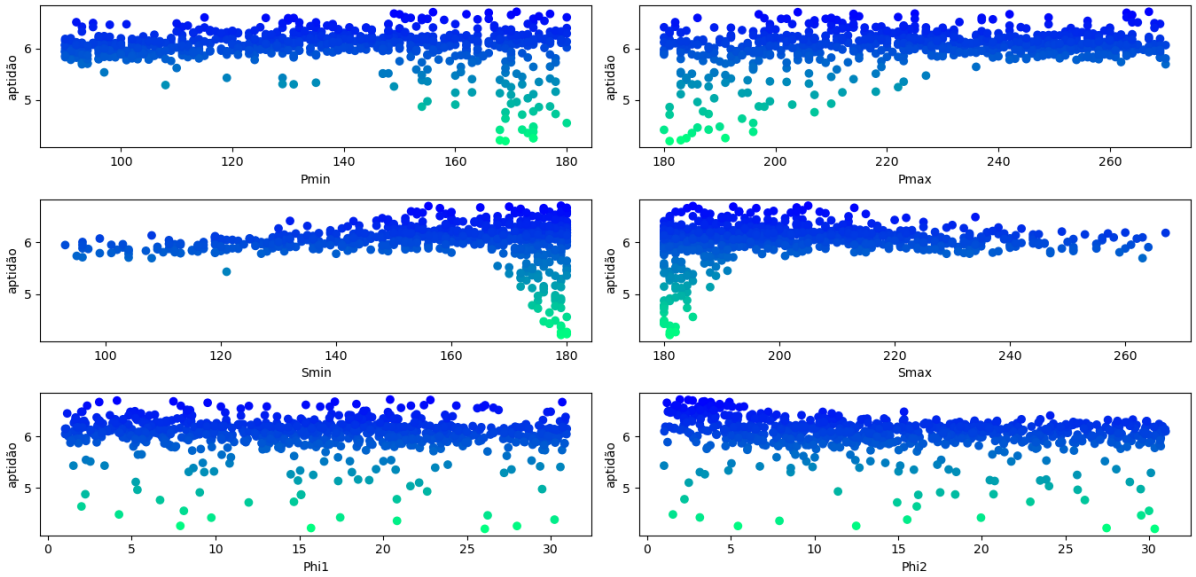

Figura 31 - Aptidão em relação aos parâmetros do PSO.

Na Figura 31, observou-se que os domínios de alguns parâmetros podem ser reduzidos. O domínio do parâmetro P_{min} é reduzido para o intervalo de 140 a 180, o S_{min} para o intervalo de 150 a 180, o S_{max} para o intervalo de 180 a 220 e o $Phi2$ de um a dez. Estes intervalos de parâmetros foram escolhidos por serem onde estão as melhores partículas mesmo que as partículas com as piores aptidões também estejam nesse intervalo. Então é realizada uma segunda bateria com quinhentas simulações. Os resultados dessa bateria são apresentados na Figura 32.

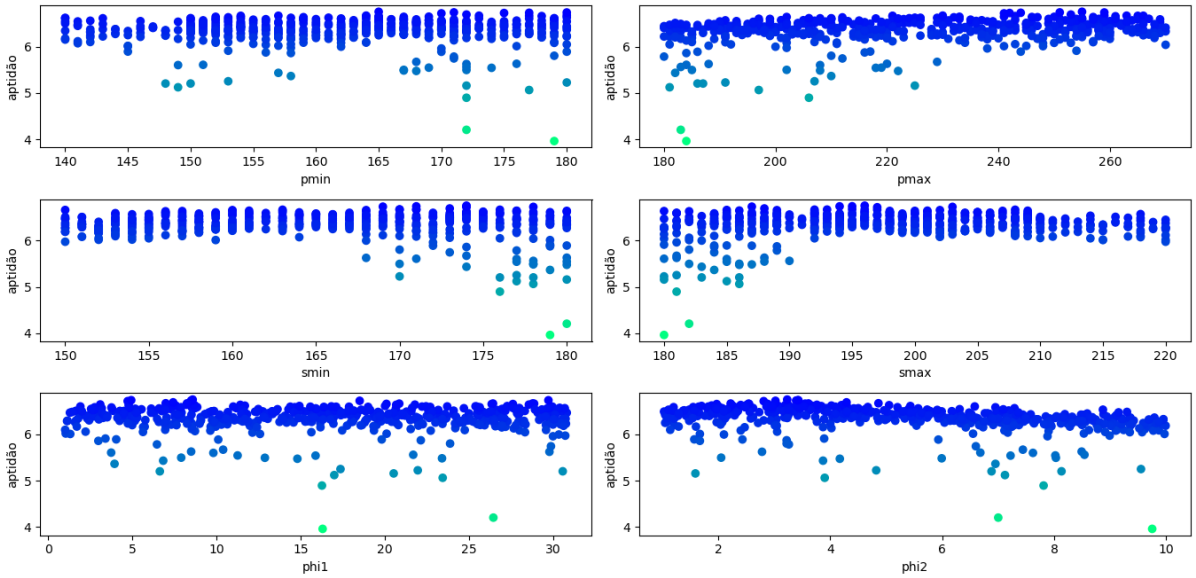

Figura 32 - Aptidão em relação aos parâmetros reduzidos do PSO.

Com os dados na Figura 32, não se pôde reduzir o espaço de busca, então, foram observados parâmetros dos cinquenta indivíduos com melhores aptidões na Figura 33.

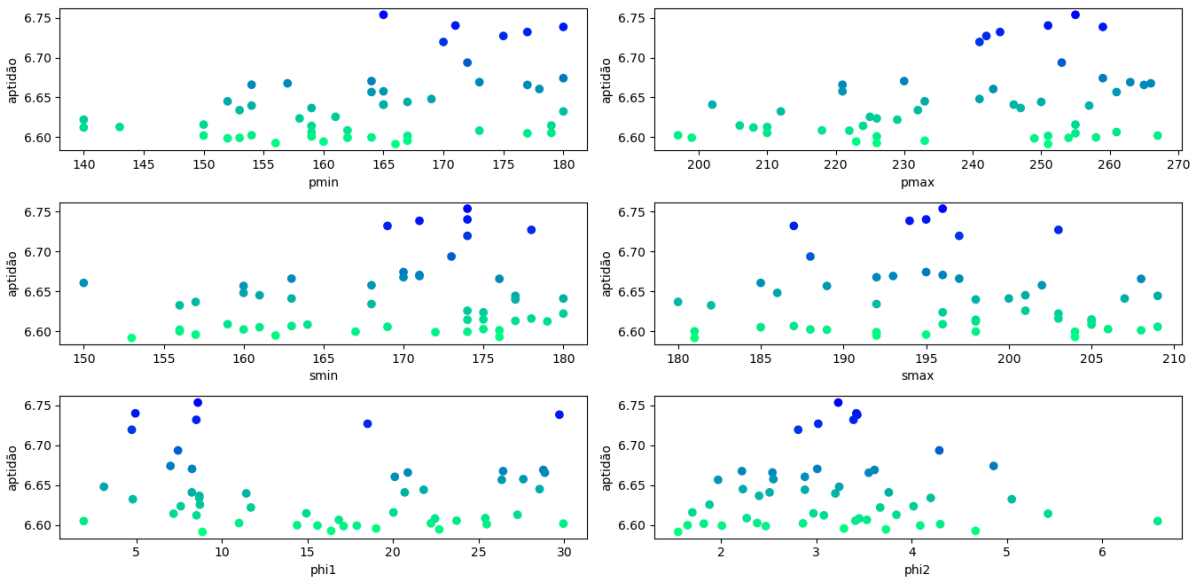

Figura 33 - Aptidão em relação aos parâmetros dos 50 melhores indivíduos obtidos pelo PSO.

Com a Figura 33, o domínio do parâmetro P_{min} é reduzido para o intervalo de 165 a 180, o P_{max} é reduzido para o intervalo de 240 a 270, o S_{min} para o intervalo de 165 a

180, o S_{max} para o intervalo de 185 a 205 e o $Phi2$ de dois a quatro. Então é realizada uma terceira bateria com mil simulações. Os resultados dessa bateria são apresentados na Figura 34.

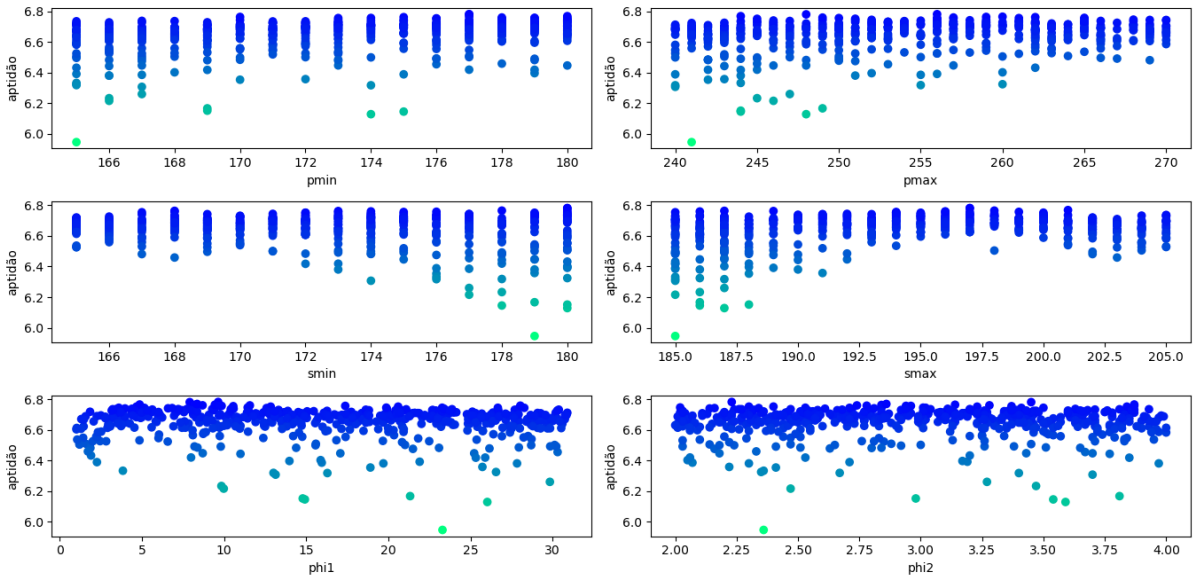

Figura 34 - Aptidão em relação aos parâmetros novamente reduzidos do PSO.

$Phi1$ e $Phi2$ são tamanhos de passo em relação ao tamanho do indivíduo. Nesse ponto duas abordagens eram possíveis, uma utilizando os valores absolutos de $Phi1$ e $Phi2$ e outra utilizando seus valores percentuais em relação ao tamanho dos indivíduos. A Tabela 12 apresenta os resultados obtidos para as gerações de zero a dois para os valores percentuais e absolutos de $Phi1$ e $Phi2$. Nela é possível observar que as melhores aptidões para uma seleção de parâmetros em uma determinada geração vai sempre ocorrer para os valores absolutos de $Phi1$ e $Phi2$. Sendo assim, os valores absolutos são utilizados para as simulações das dez melhores seleções de parâmetros.

Tabela 12 - Aptidões para as gerações de zero a dois do PSO comparando os valores absolutos e percentuais de $Phi1$ e $Phi2$.

Geração 0		Geração 1		Geração 2	
Absoluto	Percentual	Absoluto	Percentual	Absoluto	Percentual
6,767	6,749	12,393	11,922	18,492	17,192
6,750	6,771	12,451	11,687	18,695	17,251
6,771	6,784	12,414	11,608	18,383	17,160
6,763	6,754	12,381	11,690	18,460	17,093
6,768	6,771	12,373	11,738	18,616	17,136
6,743	6,749	12,275	11,572	18,179	16,887
6,765	6,765	12,261	11,649	18,139	16,978
6,742	6,709	12,293	11,607	18,230	16,928
6,748	6,755	12,423	11,724	18,613	17,174
6,751	6,734	12,388	11,702	18,443	17,115

As dez seleções de parâmetros com as melhores aptidões, obtidas nas baterias de simulações, foram simuladas para as gerações de zero a três do dendrímero. Os resultados obtidos são apresentados na Tabela 13. Com isso, os parâmetros escolhidos foram: Para o P_{min}, 177; para o P_{max}, 248; para o S_{min}, 180; para o S_{max}, 197; para o $Phi1$, 7,91; e para o $Phi2$, 3,45.

Tabela 13 - Aptidões para as gerações de zero a três do PSO com as dez melhores seleções parâmetros encontradas.

Pmin	Pmax	Smin	Smax	Phi1	Phi2	Geração 0	Geração 1	Geração 2	Geração 3
177	256	180	197	9,64	2,23	6,767	12,393	18,492	24,828
177	248	180	197	7,91	3,45	6,750	12,451	18,695	25,175
176	256	173	191	7,24	3,71	6,771	12,414	18,383	24,785
174	247	180	199	7,64	3,17	6,763	12,381	18,460	25,002
180	244	180	201	8,36	2,86	6,768	12,373	18,616	25,012
170	259	178	198	11,66	2,72	6,743	12,275	18,179	24,436
177	260	178	198	4,86	3,87	6,765	12,261	18,139	24,307
170	259	168	187	9,44	3,14	6,742	12,293	18,230	24,347
178	249	177	195	12,58	3,06	6,748	12,423	18,613	24,969
175	262	173	189	8,20	2,84	6,751	12,388	18,443	24,755

4.2.4 CMA-ES

A CMA-ES é um algoritmo que utiliza três métodos, sendo eles: Igual, Linear e Superlinear. Além disso, o algoritmo utiliza seis parâmetros que precisam ser ajustados, sendo eles: centroide, desvio padrão inicial da distribuição (σ), número de indivíduos mantidos como filhos para a próxima geração (μ), constante de acumulação para o tamanho do passo (cs), amortecimento para passo ($damps$) e constante de acumulação para matriz de covariância ($ccum$).

Antes de se iniciar as simulações, é realizado um estudo do espaço de busca para avaliar cada parâmetro, como apresentado no Capítulo 3. Nesse estudo, observou-se que o centroide deve ser mantido entre 90 e 270, o σ deve estar entre 1 e 180, o μ deve estar entre 1 e o tamanho da população, o cs deve estar entre zero e um, o $damps$ deve estar entre zero e cinco e o $ccum$ deve estar entre zero e 0,8.

A primeira bateria de simulações gera quatrocentas seleções de parâmetros, que são testadas para os três métodos. Na Figura 35, são mostradas as aptidões em função dos parâmetros considerando todos os métodos utilizados. Já na Figura 36, são mostradas as aptidões em função dos parâmetros e separadas pelos métodos. A primeira linha de gráficos mostra o resultado para o método Superlinear. A segunda linha mostra o método

Linear e a terceira, o Igual.

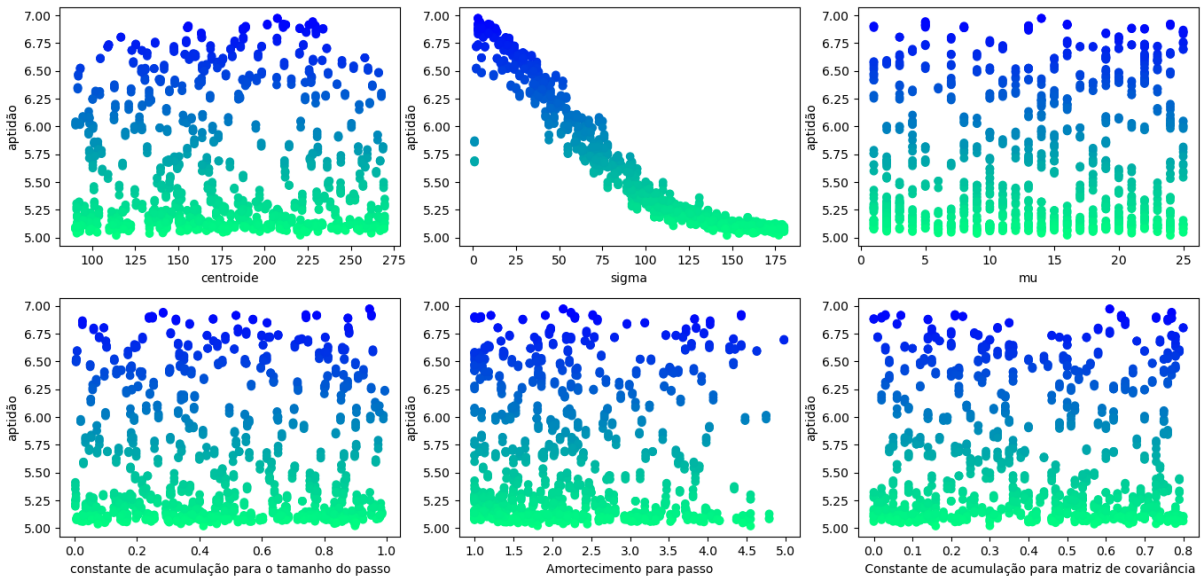

Figura 35 - Aptidão em função dos parâmetros da CMA-ES, considerando todos os métodos.

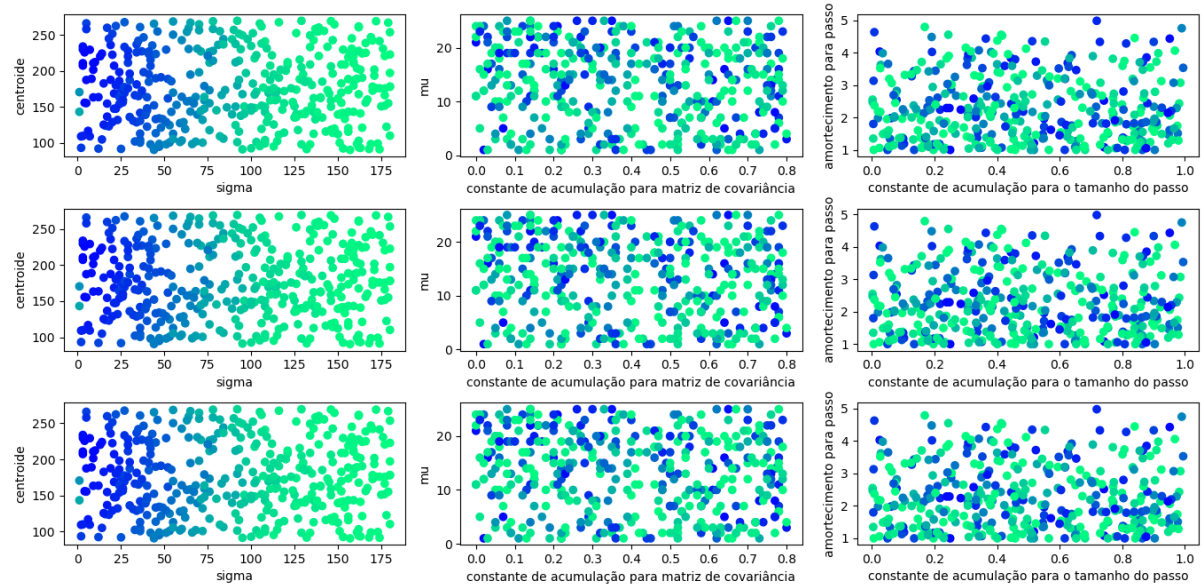

Figura 36 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos.

Na Figura 35 e na Figura 36, observou-se que o sigma é o parâmetro que mais afeta a aptidão. Para confirmar se é possível filtrar mais os domínio dos outros parâmetros, foram gerados a Figura 37 e a Figura 38 que apresentam os 10% melhores elementos e 5%

melhores elementos respectivamente.

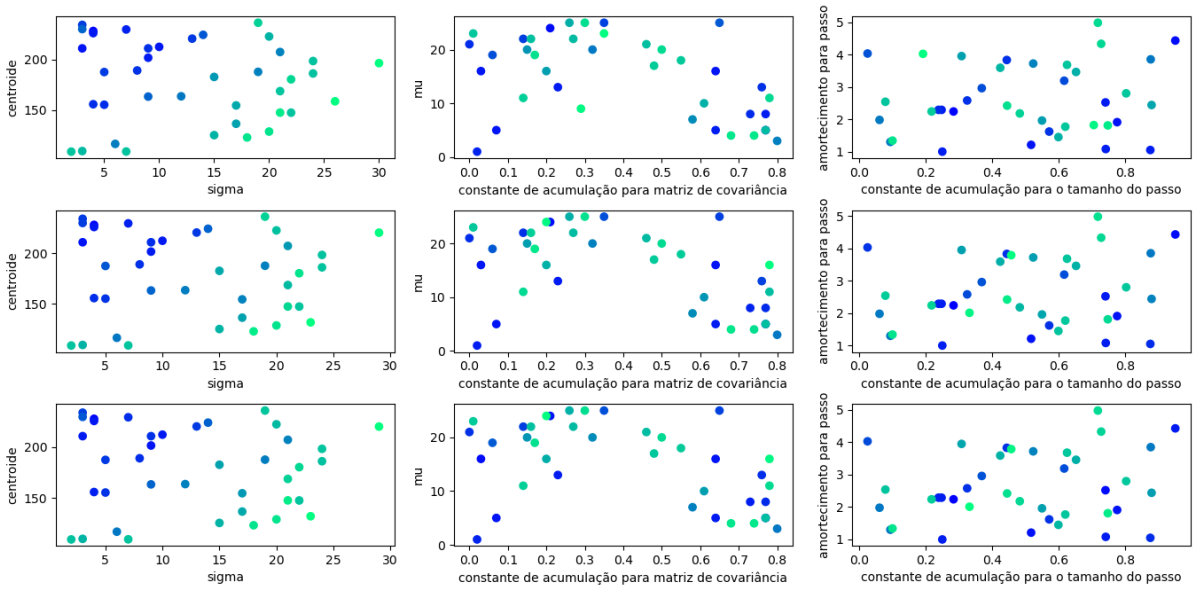

Figura 37 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos para os 10% melhores indivíduos.

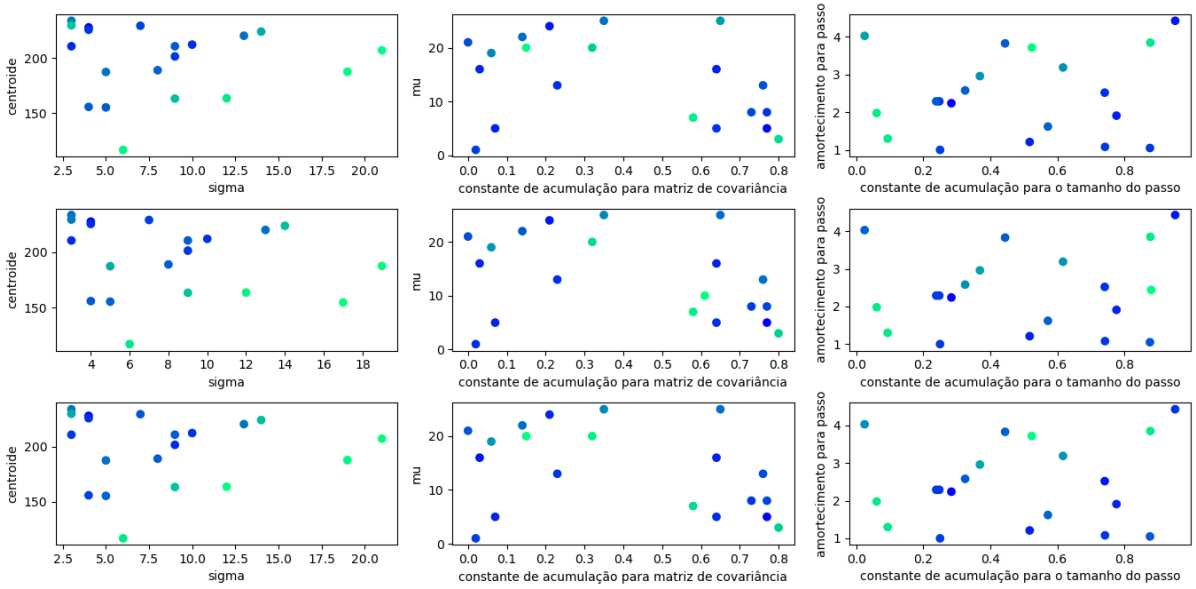

Figura 38 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos para os 5% melhores indivíduos.

Nas Figuras 35 a 38, observou-se que o parâmetro que tem o efeito mais relevante na aptidão é o σ, que deve ter seu intervalo limitado entre um e quinze. Já para os outros

parâmetros, os intervalos devem ser mantidos os mesmos. Além disso, não foi observada nenhuma diferença significativa entre os resultados obtidos pelos métodos. A média dos desvios padrão das aptidões, nas 400 seleções de parâmetros, entre os resultados obtidos pelos três métodos, é de 0,017.

Então foi inicializada uma segunda bateria de simulações com os domínios citados. As Figuras 37, 38, 39 e 40 são obtidas da mesma maneira que as Figuras 33, 34, 35 e 36, respectivamente.

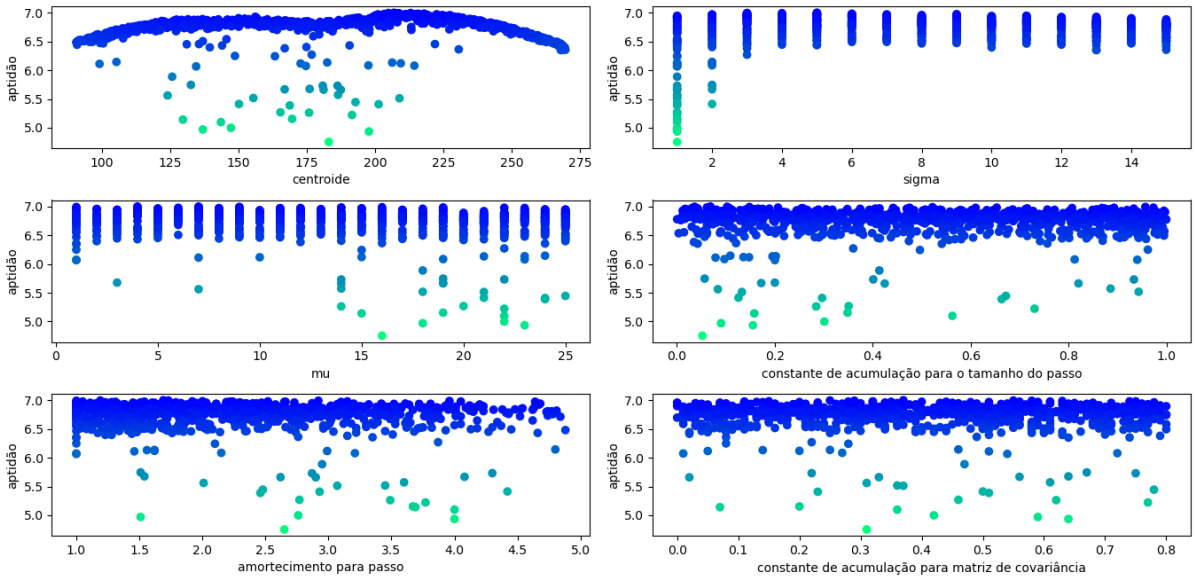

Figura 39 - Aptidão em função dos parâmetros da CMA-ES com o espaço de busca dos parâmetros reduzido.

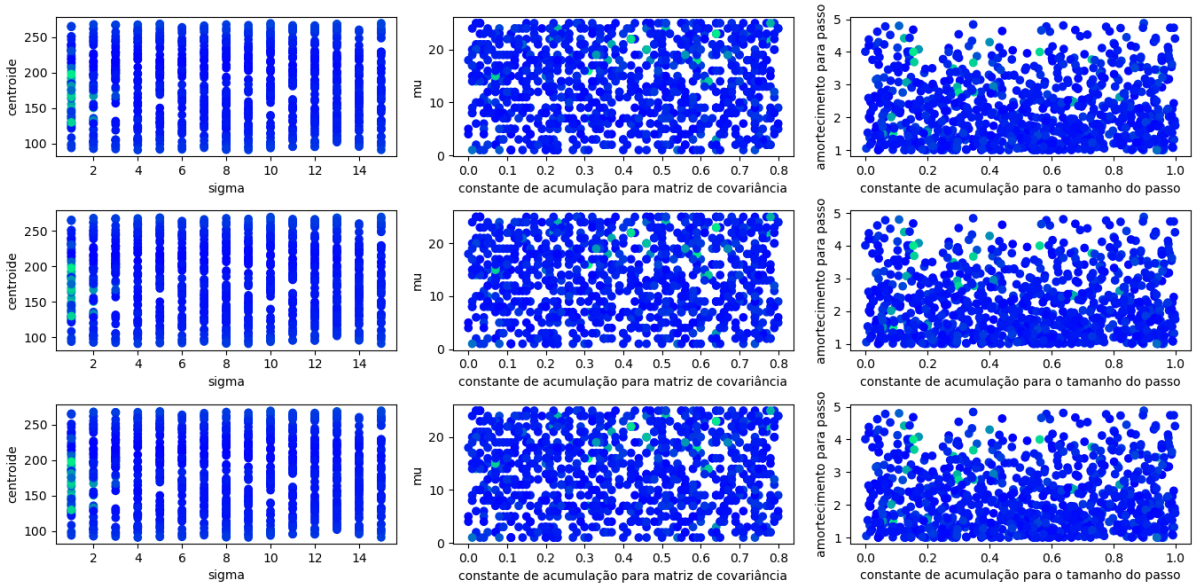

Figura 40 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos com o espaço de busca reduzido.

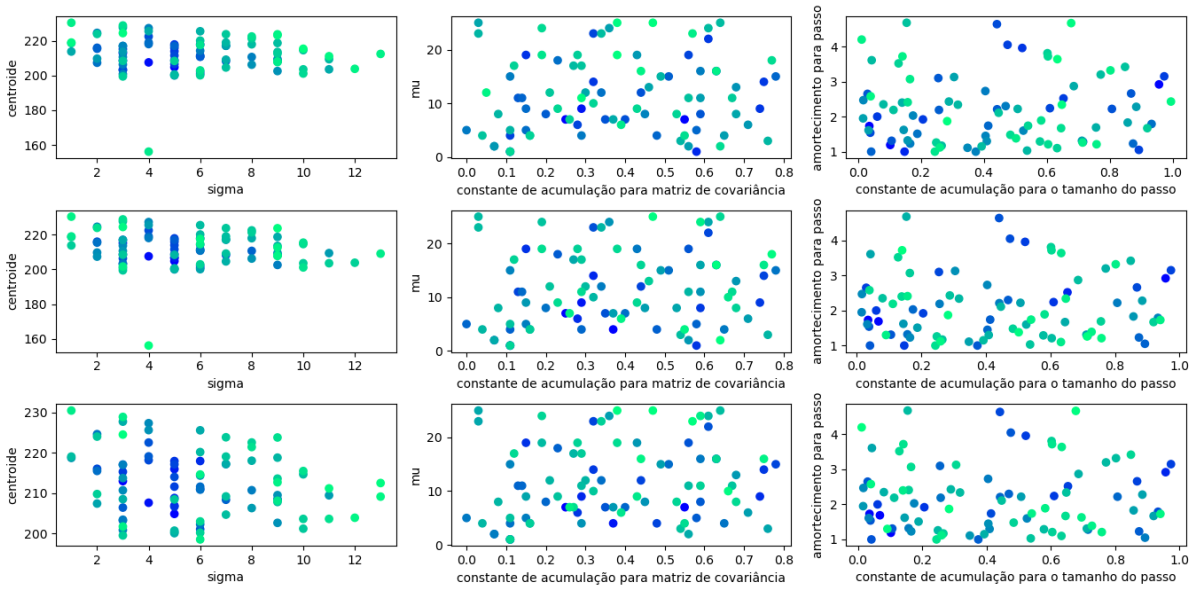

Figura 41 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos para os 10% melhores indivíduos com o espaço de busca reduzido.

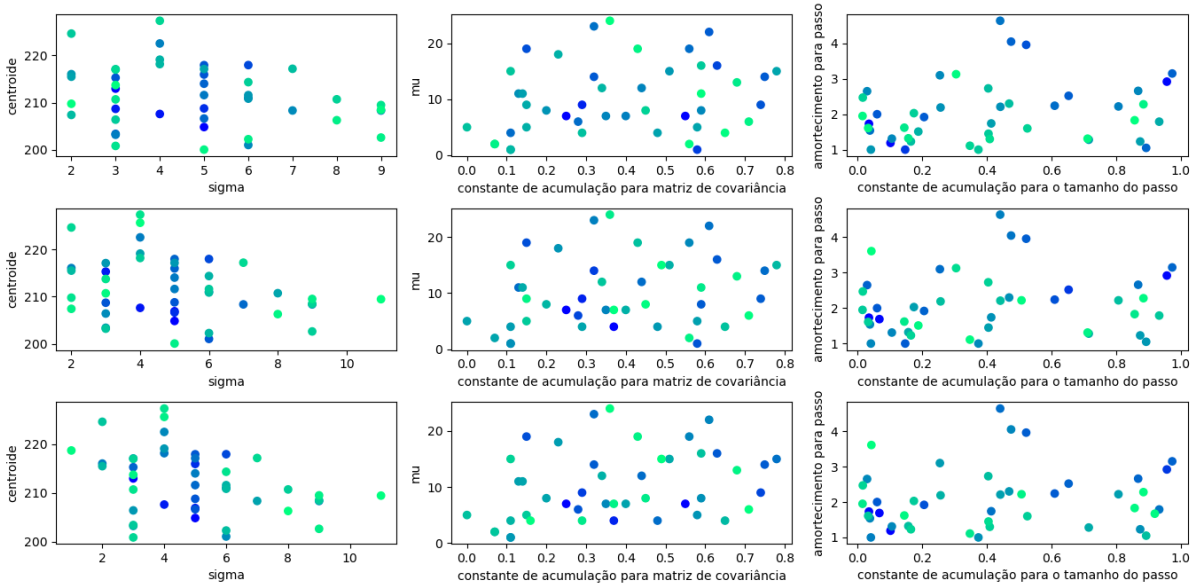

Figura 42 - Relação da aptidão com os parâmetros da CMA-ES separado por métodos para os 5% melhores indivíduos com o espaço de busca reduzido.

Na Tabela 14, são mostradas as dez seleções de parâmetros e métodos que obtêm as maiores aptidões para o algoritmo. Observou-se que as melhores seleções ocorrem com os mesmos parâmetros, modificando apenas os métodos. Para confirmar se existe diferença significativa entre os métodos foi realizado o teste-t e seu resultado é apresentado na Tabela 15. A primeira coluna da Tabela 15 mostra os métodos comparados, enquanto a segunda coluna mostra a porcentagem dos métodos serem iguais. Foi então observado que não existe diferença significativa entre os métodos. Sendo assim, para obter uma maior diversidade entre os parâmetros, foram selecionadas as dez melhores seleções de parâmetros para um mesmo método.

Tabela 14 - Melhores seleções de parâmetros encontradas para a CMA-ES.

método	centroide	σ	μ	cs	damps	ccum	aptidão
superlinear	206,88	5	4	0,068	1,69	0,37	7,005
igual	206,88	5	4	0,068	1,69	0,37	6,999
superlinear	204,83	5	7	0,036	1,73	0,25	6,998
linear	206,88	5	4	0,068	1,69	0,37	6,997
superlinear	212,97	3	7	0,103	1,19	0,55	6,995
linear	204,83	5	7	0,036	1,73	0,25	6,990
linear	212,97	3	7	0,103	1,19	0,55	6,988
igual	208,69	3	1	0,148	1	0,58	6,985
igual	204,83	5	7	0,036	1,73	0,25	6,976
igual	212,97	3	7	0,103	1,19	0,55	6,960

Tabela 15 - Teste de significância estatística entre os métodos da CMA-ES.

Métodos	Porcentagem
Superlinear x Linear	96,14
Superlinear x Igual	96,26
Igual x Linear	99,88

As dez seleções de parâmetros escolhidas são mostradas na Tabela 16. Todas as seleções foram simuladas para o método Igual. Este método foi escolhido pois não foram observadas diferenças relevantes entre os valores obtidos pelos métodos e este é o método com execução mais simples, então, os parâmetros escolhidos como melhores foram: centroide = 212,97, $\sigma = 3$, $\mu = 7$, constante de acumulação para o tamanho do passo = 0,103, amortecimento para passo = 1,19 e constante de acumulação para matriz de covariância = 0,55.

Tabela 16 - Melhores seleções de parâmetros encontradas para a CMA-ES com o método Igual para as gerações de zero a três.

centroide	σ	μ	cs	damps	ccum	geração 0	geração 1	geração 2	geração 3
212,97	3	7	0,103	1,19	0,55	6,964	12,984	19,688	26,962
207,6	4	9	0,957	2,92	0,29	6,946	12,968	19,680	26,883
204,83	5	7	0,036	1,73	0,25	6,968	12,986	19,667	26,871
206,88	5	4	0,068	1,69	0,37	6,999	12,989	19,645	26,806
206,64	5	6	0,610	2,24	0,28	6,966	12,966	19,640	26,799
208,69	3	1	0,148	1	0,58	6,988	12,976	19,626	26,768
215,95	5	9	0,206	1,92	0,74	6,966	12,951	19,620	26,793
217,94	5	14	0,973	3,15	0,75	6,937	12,900	19,536	26,706
208,78	5	16	0,522	3,96	0,63	6,740	12,573	19,217	26,296
217,95	6	19	0,061	2	0,15	6,597	12,473	19,147	26,263

4.2.5 Algoritmos

Uma vez que todos os algoritmos foram configurados com os melhores métodos e parâmetros obtidos, para descobrir o melhor algoritmo entre os simulados é realizado o teste-t para verificar se os seus resultados podem ser realmente considerados diferentes. Para realizar o teste a geração zero do dendrímero foi simulada cem vezes para cada algoritmo. Antes de realizar o teste-t é realizado o teste de Levene para garantir a possibilidade da utilização do teste-t. Os resultados desses testes são mostrados na Tabela 17, onde a primeira coluna mostra os algoritmos comparados e a segunda mostra o p-valor entre os algoritmos. O quanto menor o p-valor maior é a probabilidade de rejeitar a hipótese nula. Na Tabela 17, a sigla VBGA_U representa o GA utilizado originalmente pelo PyPolyBuilder, a sigla GA_DEAP representa o GA desenvolvido nesse trabalho utilizando a biblioteca DEAP, enquanto as siglas DE, PSO e CMA-ES representam os respectivos algoritmos. Essas nomenclaturas também são utilizadas na Tabela 18.

Tabela 17 - Teste de Levene comparando as variâncias dos algoritmos utilizados neste trabalho.

Algoritmos	p-valor
VBGA X DEAP	0,044
VBGA X DE	$4{,}224x10^{-18}$
VBGA X PSO	$1{,}834x10^{-11}$
VBGA X CMA-ES	$4{,}631x10^{-23}$
DEAP X DE	$1{,}176x10^{-16}$
DEAP X PSO	$1{,}403x10^{-8}$
DEAP X CMA-ES	$3{,}762x10^{-23}$
DE X PSO	0,002
DE X CMA-ES	$2{,}222x10^{-6}$
PSO X CMA-ES	$6{,}188x10^{-9}$

Como os resultados do teste de Levene são sempre menores que 0,05 é possível afirmar que o teste-t pode ser utilizado. Os resultados do teste-t são mostrados na Tabela 18, onde a primeira coluna mostra os algoritmos comparados e a segunda mostra a porcentagem dos algoritmos serem iguais.

Tabela 18 - Teste de significância estatística entre os algoritmos utilizados neste trabalho.

Algoritmos	Porcentagem
VBGA_U X GA_DEAP	$1{,}33x10^{-11}$
VBGA_U X DE	$1{,}9x10^{-60}$
VBGA_U X PSO	$1{,}19x10^{-45}$
VBGA_U X CMA-ES	$7{,}62x10^{-65}$
GA_DEAP X DE	$8{,}11x10^{-43}$
GA_DEAP X PSO	$1{,}04x10^{-23}$
GA_DEAP X CMA-ES	$6{,}99x10^{-49}$
DE X PSO	$8{,}54x10^{-14}$
DE X CMA-ES	$5{,}07x10^{-3}$
PSO X CMA-ES	$4{,}31x10^{-24}$

Na Tabela 18 é possível observar que todos os algoritmos são significativamente

diferentes dos outros. Inicialmente era esperado que os algoritmos VBGA_U e GA_DEAP não tivessem uma diferença significativa muito grande. O que pode justificar essa diferença são os métodos utilizados e os parâmetros utilizados. Em seguida, os algoritmos são testados para as gerações de zero a cinco, sete e dez. Na Tabela 19, são apresentadas as aptidões obtidas. Na Figura 43, são apresentados os gráficos das aptidões dos algoritmos ao longo das gerações.

Tabela 19 - Quantidade de melhores aptidões para cada método.

	VBGA_U	GA_DEAP	DE	CMA-ES	PSO
geração 0	6,26	6,49	6,93	6,99	6,75
geração 1	11,12	11,78	12,62	13,01	12,45
geração 2	15,90	17,31	18,78	19,74	18,63
geração 3	20,69	23,31	25,11	26,99	25,17
geração 4	24,36	28,98	31,60	34,59	31,79
geração 5	29,03	34,97	37,75	42,46	38,57
geração 7	37,12	46,69	49,29	58,44	51,54
geração 10	44,26	59,68	63,55	erro	70,40

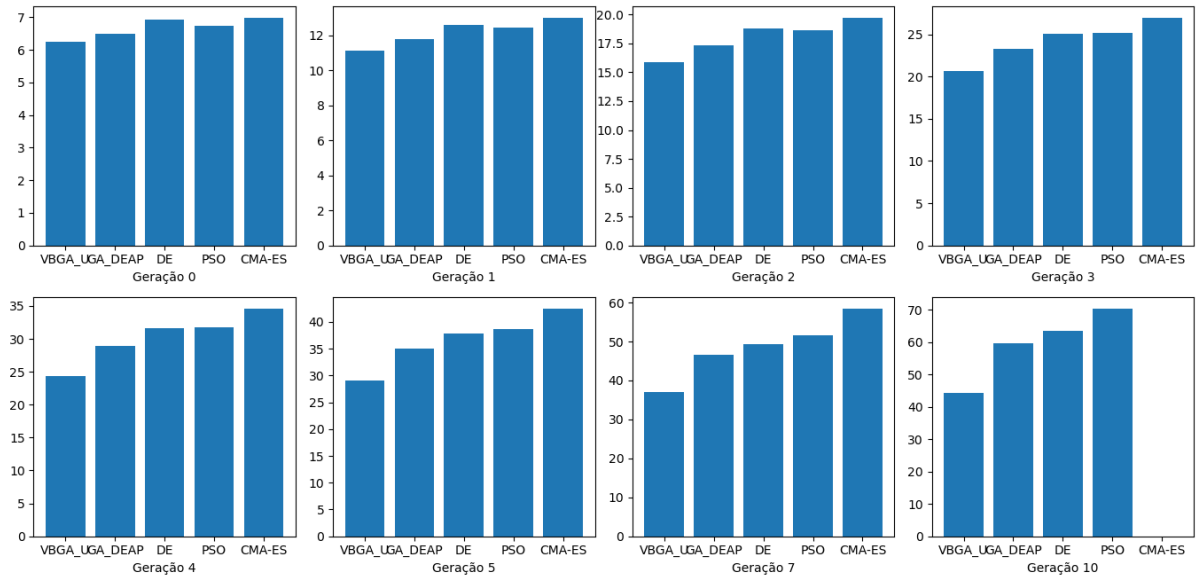

Figura 43 -Aptidões dos algoritmos nas gerações de zero a cinco, sete e dez.

Na Tabela 19, observa-se que os melhores resultados nas gerações menores ocorrem para a CMA-ES, porém na geração 10 o algoritmo apresenta um erro. Esse erro é

ocasionado pelo aumento do tamanho da matriz de covariância, que é uma matriz N por N, onde N é o número de valores a serem otimizados. Esse número de valores cresce de maneira exponencial ao longo das gerações, ocasionando assim um erro na memória. O erro pode ser visto na Figura 44.

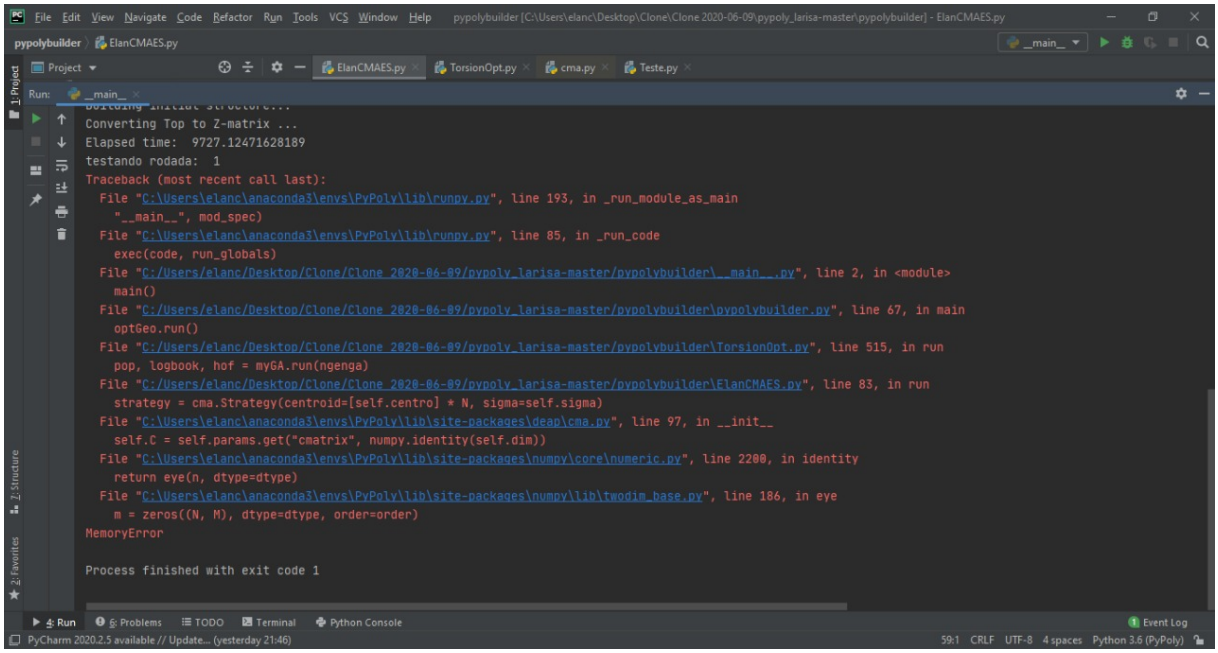

Figura 44 -Aptidões dos indivíduos com melhores parâmetros e métodos para cada algoritmo nas gerações de zero a cinco, sete e dez.

Observando os métodos restantes, pode-se concluir que aqueles baseados em GA são os menos eficientes. Já entre a DE e o PSO, a DE obtém melhores resultados para dendrímeros de gerações menores, enquanto o PSO obtém melhores resultados gerações maiores.

Dos algoritmos citados, o GA tem um total de 24 casamentos de métodos possíveis, a DE utiliza sete métodos e o PSO utiliza um método. Por sua vez, a CMA-ES utiliza três métodos, mas, durante o ajuste dos parâmetros, observou-se que os métodos não têm efeito sobre as aptidões. Com isso, é possível concluir que o GA e a DE dependem de um ajuste mais longo em relação aos métodos.

Em relação aos parâmetros, o GA depende de até seis deles, o DE depende de um ou dois, o PSO depende de seis e a CMA-ES, de seis. Sendo assim, o PSO e a CMA-ES dependem de um maior esforço para o ajuste de parâmetros. Porém, ao se restringir o valor da Taxa de Cruzamento Interno para de 0,2 a 04 , os resultados indicam

que os demais parâmetros do GA com seleção por Roleta e mutação Gaussiana não têm efeito muito representativo sobre a aptidão dos indivíduos Já a DE é sensível aos três parâmetros de maneira semelhante. Enquanto isso o PSO é muito dependente de três parâmetros:$Phi2$; P_{min}; S_{min}, varia pouco em relação aos P_{max} e S_{max} e é praticamente independente do $Phi1$. Além disso, durante o ajuste das aptidões observou-se que o algoritmo é bem comportado para valores altos de P_{max} e S_{max} e valores baixos de P_{min} e S_{min} porém as melhores aptidões ocorrem nas faixas onde o algoritmo também apresenta os piores resultados. E finalmente, a CMA-ES praticamente é dependente apenas do σ sendo muito pouco afetado pelos outros parâmetros.

Analisando-se a complexidade de se ajustar os métodos e os parâmetros conjuntamente, pode-se concluir, portanto, que todos os algoritmos têm uma complexidade parecida para o seu ajuste. Sendo assim, a aptidão deve sempre ser observada para se decidir qual o algoritmo a ser utilizado.

CONCLUSÃO

Esta dissertação abordou o uso de GA, DE, PSO e CMA-ES para otimizar a arquitetura de um dendrímero, controlando as suas características. O dendrímero utilizado foi o PAMAM que é largamente utilizado na indústria farmacêutica, bem como em outras aplicações. Além disso, este trabalho propõe diferentes métodos para a geração de uma matriz-Z que era o gargalo computacional do PypolyBuilder. Ainda neste trabalho, foram propostas novas funções de aptidão para a otimização global e um espaço de busca diferente.

Com relação ao primeiro objetivo desta dissertação, que foi otimizar a matriz-Z, foram desenvolvidos dois métodos que conseguiram criar a matriz de maneira muito mais rápida que a original, centenas de vezes mais rápida para as gerações menores do dendrímero e milhares de vezes para gerações um pouco maiores. Pela natureza das abordagens, essa diferença se torna maior a cada geração. Além disso, os métodos possuem robustez, uma vez que podem ser aplicados a outros dendrímeros ou polímeros.

Como subprodutos obtidos no desenvolvimento deste trabalho, foi proposta uma nova função de aptidão que pode variar a forma do dendrímeros, ajustando três pesos de maneira simples. O dendrímero pode atingir formas aproximadamente lineares, aproximadamente em forma de disco, ou aproximadamente esféricas (que é o objetivo no problema abordado nesta dissertação). Note-se que a função utilizada originalmente sempre forçava o dendrímero a ter uma forma aproximadamente linear. Outro subproduto foi a restrição do espaço de busca da otimização global para um arco entre noventa e duzentos e setenta graus. Esses valores foram escolhidos de maneira que os valores das torções estejam próximos a cento e oitenta graus para, assim, aumentar o raio da molécula, mas sem restringir muito o espaço de busca, evitando assim a perda de diversidade durante as evoluções.

Como segundo objetivo desta dissertação, diversos algoritmos de otimização global foram aplicados com a intenção de se aumentar o raio e a esfericidade do dendrímero. O GA apresentou os piores resultados, tendo aptidões piores que os outros algoritmos propostos em todas as gerações e superando apenas o algoritmo implementado originalmente. Os outros algoritmos foram competitivos entre si, com a CMA-ES apresentando melhores resultados que os outros. Porém, devido ao crescimento da matriz de covariância para

gerações mais altas, o algoritmo apresentou erro na memória. Supõe-se que liberando mais espaço na memória para o Python ou utilizando computadores mais potentes, esse problema seja resolvido. Porém, com o distanciamento social exigido pela pandemia, não foi possível ir ao laboratório realizar esses testes. Com isso, o algoritmo indicado para gerações maiores é o PSO, uma vez que este não apresentou problemas de memória até a maior geração testada e, quando comparado com a evolução diferencial, apresentou melhores aptidões.

Os trabalhos futuros sugeridos como sequência desta dissertação envolvem simulações utilizando CMA-ES com mais memória disponível. Além disso, ajustar a DE e o PSO para se evitar o cálculo da aptidão de indivíduos que já tenham sido calculadas nas gerações anteriores do algoritmo. Ademais, realizar tentativas de ajustes de parâmetros para gerações maiores (neste trabalho todo o ajuste de parâmetros e métodos foi realizado na geração zero do dendrímero). Também, pode-se utilizar métodos evolutivos para o ajuste de parâmetros e comparação dos resultados com os obtidos neste trabalho, o qual utilizou uma abordagem aleatória. Além disso, pode-se conduzir o ajuste de parâmetros em outros dendrímeros e polímeros com o objetivo de verificar a robustez dos parâmetros selecionados. Em adição aos resultados deste trabalho, usar diferentes abordagens para a utilização dos algoritmos de otimização que aproveitem as características repetitivas dos blocos construtores. E finalmente, o primeiro trabalho futuro a se desenvolver é a implementação dos métodos desenvolvidos neste trabalho ao PyPolyBulder.

REFERÊNCIAS

[1] TOMALIA, D. A. Birth of a new macromolecular architecture: dendrimers as quantized building blocks for nanoscale synthetic polymer chemistry. *Progress in Polymer Science*, Elsevier, v. 30, n. 3-4, p. 294–324, 2005.

[2] MAITI, P. K. et al. Structure of pamam dendrimers: Generations 1 through 11. *Macromolecules*, ACS Publications, v. 37, n. 16, p. 6236–6254, 2004.

[3] CHAUHAN, A.; ANTON, B.; SINGH, M. K. Dendrimers for drug solubilization, dissolution and bioavailability. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 59–92.

[4] TUNKI, L. et al. Pharmacokinetic considerations in design of dendrimer-based nanomedicines. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 93–106.

[5] TUNKI, L. et al. Dendrimer-based targeted drug delivery. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 107–129.

[6] TRIPATHI, P. K.; TRIPATHI, S. Dendrimers for anticancer drug delivery. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 131–150.

[7] RAJANI, C. et al. Cancer-targeted chemotherapy: Emerging role of the folate anchored dendrimer as drug delivery nanocarrier. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 151–198.

[8] REDDY, T. S.; BHARGAVA, S. K. Design of dendrimer based prodrugs. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 199–210.

[9] RAI, D. B.; POOJA, D.; KULHARI, H. Dendrimers in gene delivery. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 211–231.

[10] BONAM, S. R. et al. Dendrimers in immunotherapy and hormone therapy. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 233–249.

[11] RAI, D. B. et al. Dendrimers for diagnostic applications. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 291–324.

[12] CHAUHAN, A. et al. Dendrimer-based marketed formulations and miscellaneous applications in cosmetics, veterinary, and agriculture. In: *Pharmaceutical Applications of Dendrimers*. [S.l.]: Elsevier, 2020. p. 325–334.

[13] ALLEN, M. P.; TILDESLEY, D. J. *Computer simulation of liquids*. [S.l.]: Oxford university press, 2017.

[14] BALLAUFF, M.; LIKOS, C. N. Dendrimers in solution: insight from theory and simulation. *Angewandte Chemie International Edition*, Wiley Online Library, v. 43, n. 23, p. 2998–3020, 2004.

[15] HORTA, B. A. C. *PyPolyBuilder: An automatic topology builder for dendrimers, branched molecules and supramolecular structures*. Disponível em: <https://github.com/vitorhorta/pypolybuilder/blob/development/README.md>. Acesso em: 24 de Julho de 2020.

[16] HOLLAND, J. H. et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT press, 1992.

[17] NAMBA, A. M.; SILVA, V. B. da; SILVA, C. D. Dinâmica molecular: teoria e aplicações em planejamento de fármacos. *Eclética Química*, SciELO Brasil, v. 33, n. 4, p. 13–24, 2008.

[18] LIMA, A. L. d. Simulações de dinâmica molecular de dobras de grafeno. Programa de Pós-Graduação em Ciências–Física de Materiais. Departamento de ..., 2013.

[19] BOSMAN, d. A.; JANSSEN, H.; MEIJER, E. About dendrimers: structure, physical properties, and applications. *Chemical reviews*, ACS Publications, v. 99, n. 7, p. 1665–1688, 1999.

[20] RAMOS, M. C.; HORTA, V. A.; HORTA, B. A. Molecular dynamics simulations of pamam and ppi dendrimers using the gromos-compatible 2016h66 force field. *Journal of chemical information and modeling*, ACS Publications, v. 59, n. 4, p. 1444–1457, 2019.

[21] JEISON, D. *Matriz Z*. Disponível em: <http://www.quimica.ufpr.br/edulsa/LQT/zmatrix.html Acesso em: 24 de Julho de 2020.

[22] FORTIN, F.-A. et al. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, v. 13, p. 2171–2175, jul 2012.

[23] LI, X.; ENGELBRECHT, A.; EPITROPAKIS, M. Results of the 2013 ieee cec competition on niching methods for multimodal optimization. In: *Report presented at 2013 IEEE Congress on Evolutionary Computation Competition on: Niching Methods for Multimodal Optimization*. [S.l.: s.n.], 2013.

[24] BUHLEIER, E. et al. "cascade-and"nonskid-chain-like"syntheses of molecular cavity topologies. 1978.

[25] WÖRNER, C.; MÜLHAUPT, R. Polynitrile-and polyamine-functional poly (trimethylene imine) dendrimers. *Angewandte Chemie International Edition in English*, Wiley Online Library, v. 32, n. 9, p. 1306–1308, 1993.

[26] BERG, E. M. de Brabander-van den; MEIJER, E. Poly (propylene imine) dendrimers: large-scale synthesis by hetereogeneously catalyzed hydrogenations. *Angewandte Chemie International Edition in English*, Wiley Online Library, v. 32, n. 9, p. 1308–1311, 1993.

[27] TOMALIA, D. A. et al. A new class of polymers: starburst-dendritic macromolecules. *Polymer journal*, Nature Publishing Group, v. 17, n. 1, p. 117–132, 1985.

[28] YANG, H.; LOPINA, S. T. Stealth dendrimers for antiarrhythmic quinidine delivery. *Journal of Materials Science: Materials in Medicine*, Springer, v. 18, n. 10, p. 2061–2065, 2007.

[29] BERGSMA, J. P. *Combining mean field calculations with Monte Carlo simulations for polymer gels and dendrimers*. Tese (Doutorado) — Wageningen University, 2019.

[30] HAWKER, C. J.; FRECHET, J. M. Preparation of polymers with controlled molecular architecture. a new convergent approach to dendritic macromolecules. *Journal of the American Chemical Society*, ACS Publications, v. 112, n. 21, p. 7638–7647, 1990.

[31] ARAÚJO, R. V. d. et al. New advances in general biomedical applications of pamam dendrimers. *Molecules*, Multidisciplinary Digital Publishing Institute, v. 23, n. 11, p. 2849, 2018.

[32] TOMALIA, D. A.; FRÉCHET, J. M. Discovery of dendrimers and dendritic polymers: a brief historical perspective. *Journal of Polymer Science Part A: Polymer Chemistry*, Wiley Online Library, v. 40, n. 16, p. 2719–2728, 2002.

[33] MENJOGE, A. R.; KANNAN, R. M.; TOMALIA, D. A. Dendrimer-based drug and imaging conjugates: design considerations for nanomedical applications. *Drug discovery today*, Elsevier, v. 15, n. 5-6, p. 171–185, 2010.

[34] TOMALIA, D. A.; HALL, M.; HEDSTRAND, D. M. Starburst dendrimers. iii. the importance of branch junction symmetry in the development of topological shell molecules. *Journal of the American Chemical Society*, ACS Publications, v. 109, n. 5, p. 1601–1603, 1987.

[35] TAWFIK, M. A.; TADROS, M. I.; MOHAMED, M. I. Polyamidoamine (pamam) dendrimers as potential release modulators and oral bioavailability enhancers of vardenafil hydrochloride. *Pharmaceutical development and technology*, Taylor & Francis, v. 24, n. 3, p. 293–302, 2019.

[36] GOTHWAL, A. et al. Boosted memory and improved brain bioavailability of rivastigmine: targeting effort to the brain using covalently tethered lower generation pamam dendrimers with lactoferrin. *Molecular pharmaceutics*, ACS Publications, v. 15, n. 10, p. 4538–4549, 2018.

[37] LAI, S. et al. Liposomes for effective drug delivery to the ocular posterior chamber. *Journal of nanobiotechnology*, BioMed Central, v. 17, n. 1, p. 1–12, 2019.

[38] BRAUN, C. S. et al. Structure/function relationships of polyamidoamine/dna dendrimers as gene delivery vehicles. *Journal of pharmaceutical sciences*, Elsevier, v. 94, n. 2, p. 423–436, 2005.

[39] CHOI, J. S. et al. Enhanced transfection efficiency of pamam dendrimer by surface modification with l-arginine. *Journal of controlled release*, Elsevier, v. 99, n. 3, p. 445–456, 2004.

[40] CRAMPTON, H. L.; SIMANEK, E. E. Dendrimers as drug delivery vehicles: non-covalent interactions of bioactive compounds with dendrimers. *Polymer international*, Wiley Online Library, v. 56, n. 4, p. 489–496, 2007.

[41] TOLIA, G. T.; CHOI, H. H. The role of dendrimers in topical drug delivery. *Pharmaceutical technology*, v. 32, n. 11, p. 88–98, 2008.

[42] SATO, N. et al. Tumor targeting and imaging of intraperitoneal tumors by use of antisense oligo-dna complexed with dendrimers and/or avidin in mice. *Clinical cancer research*, AACR, v. 7, n. 11, p. 3606–3612, 2001.

[43] THOMAS, T. P. et al. In vitro targeting of synthesized antibody-conjugated dendrimer nanoparticles. *Biomacromolecules*, ACS Publications, v. 5, n. 6, p. 2269–2274, 2004.

[44] PATRI, A. K. et al. Synthesis and in vitro testing of j591 antibody- dendrimer conjugates for targeted prostate cancer therapy. *Bioconjugate chemistry*, ACS Publications, v. 15, n. 6, p. 1174–1181, 2004.

[45] SINGH, P. Dendrimers and their applications in immunoassays and clinical diagnostics. *Biotechnology and applied biochemistry*, Wiley Online Library, v. 48, n. 1, p. 1–9, 2007.

[46] SPANGLER, B. D.; SPANGLER, C. W. *Biosensors utilizing dendrimer-immobilized ligands and there use thereof*. [S.l.]: Google Patents, nov. 21 2006. US Patent 7,138,121.

[47] KHOPADE, A. J. et al. Effect of dendrimer on entrapment and release of bioactive from liposomes. *International journal of pharmaceutics*, Elsevier, v. 232, n. 1-2, p. 157–162, 2002.

[48] CHAUHAN, A. S. et al. Dendrimer-mediated transdermal delivery: enhanced bioavailability of indomethacin. *Journal of controlled release*, Elsevier, v. 90, n. 3, p. 335–343, 2003.

[49] VANDAMME, T. F.; BROBECK, L. Poly (amidoamine) dendrimers as ophthalmic vehicles for ocular delivery of pilocarpine nitrate and tropicamide. *Journal of controlled release*, Elsevier, v. 102, n. 1, p. 23–38, 2005.

[50] SHAUNAK, S. et al. Polyvalent dendrimer glucosamine conjugates prevent scar tissue formation. *Nature biotechnology*, Nature Publishing Group, v. 22, n. 8, p. 977–984, 2004.

[51] D'EMANUELE, A. et al. The use of a dendrimer-propranolol prodrug to bypass efflux transporters and enhance oral bioavailability. *Journal of controlled release*, Elsevier, v. 95, n. 3, p. 447–453, 2004.

[52] NAJLAH, M. et al. In vitro evaluation of dendrimer prodrugs for oral drug delivery. *International journal of pharmaceutics*, Elsevier, v. 336, n. 1, p. 183–190, 2007.

[53] DEVARAKONDA, B. et al. Comparison of the aqueous solubilization of practically insoluble niclosamide by polyamidoamine (pamam) dendrimers and cyclodextrins. *International journal of pharmaceutics*, Elsevier, v. 304, n. 1-2, p. 193–209, 2005.

[54] MA, M. et al. Evaluation of polyamidoamine (pamam) dendrimers as drug carriers of anti-bacterial drugs using sulfamethoxazole (smz) as a model drug. *European journal of medicinal chemistry*, Elsevier, v. 42, n. 1, p. 93–98, 2007.

[55] DEVARAKONDA, B. et al. Effect of ph on the solubility and release of furosemide from polyamidoamine (pamam) dendrimer complexes. *International journal of pharmaceutics*, Elsevier, v. 345, n. 1-2, p. 142–153, 2007.

[56] NA, M. et al. Dendrimers as potential drug carriers. part ii. prolonged delivery of ketoprofen by in vitro and in vivo studies. *European Journal of Medicinal Chemistry*, Elsevier, v. 41, n. 5, p. 670–674, 2006.

[57] PRAJAPATI, R. N. et al. Dendimer-mediated solubilization, formulation development and in vitro- in vivo assessment of piroxicam. *Molecular pharmaceutics*, ACS Publications, v. 6, n. 3, p. 940–950, 2009.

[58] NAVATH, R. S. et al. Dendrimer- drug conjugates for tailored intracellular drug release based on glutathione levels. *Bioconjugate chemistry*, ACS Publications, v. 19, n. 12, p. 2446–2455, 2008.

[59] WIWATTANAPATAPEE, R.; LOMLIM, L.; SARAMUNEE, K. Dendrimers conjugates for colonic delivery of 5-aminosalicylic acid. *Journal of controlled release*, Elsevier, v. 88, n. 1, p. 1–9, 2003.

[60] DEVARAKONDA, B.; LI, N.; VILLIERS, M. M. de. Effect of polyamidoamine (pamam) dendrimers on the in vitro release of water-insoluble nifedipine from aqueous gels. *AAPS PharmSciTech*, Springer, v. 6, n. 3, p. E504–E512, 2005.

[61] VENUGANTI, V. V. K.; PERUMAL, O. P. Poly (amidoamine) dendrimers as skin penetration enhancers: influence of charge, generation, and concentration. *Journal of pharmaceutical sciences*, Elsevier, v. 98, n. 7, p. 2345–2356, 2009.

[62] YANG, W. et al. Targeting cancer cells with biotin–dendrimer conjugates. *European journal of medicinal chemistry*, Elsevier, v. 44, n. 2, p. 862–868, 2009.

[63] XU, R. et al. In vivo evaluation of a pamam-cystamine-(gd-do3a) conjugate as a biodegradable macromolecular mri contrast agent. *Experimental Biology and Medicine*, SAGE Publications Sage UK: London, England, v. 232, n. 8, p. 1081–1089, 2007.

[64] KOBAYASHI, H. et al. Rapid accumulation and internalization of radiolabeled herceptin in an inflammatory breast cancer xenograft with vasculogenic mimicry predicted by the contrast-enhanced dynamic mri with the macromolecular contrast agent g6-(1b4m-gd) 256. *Cancer research*, AACR, v. 62, n. 3, p. 860–866, 2002.

[65] TALANOV, V. S. et al. Dendrimer-based nanoprobe for dual modality magnetic resonance and fluorescence imaging. *Nano letters*, ACS Publications, v. 6, n. 7, p. 1459–1463, 2006.

[66] KOBAYASHI, H. et al. Micro-magnetic resonance lymphangiography in mice using a novel dendrimer-based magnetic resonance imaging contrast agent. *Cancer research*, AACR, v. 63, n. 2, p. 271–276, 2003.

[67] KOBAYASHI, H. et al. Lymphatic drainage imaging of breast cancer in mice by micro-magnetic resonance lymphangiography using a nano-size paramagnetic contrast agent. *Journal of the National Cancer Institute*, Oxford University Press, v. 96, n. 9, p. 703–708, 2004.

[68] KOBAYASHI, H. et al. Renal tubular damage detected by dynamic micro-mri with a dendrimer-based magnetic resonance contrast agent. *Kidney international*, Elsevier, v. 61, n. 6, p. 1980–1985, 2002.

[69] GERRETSEN, S. et al. Cardiac cine mri: comparison of 1.5 t, non-enhanced 3.0 t and blood pool enhanced 3.0 t imaging. *European journal of radiology*, Elsevier, v. 65, n. 1, p. 80–85, 2008.

[70] XU, H. et al. Preparation and preliminary evaluation of a biotin-targeted, lectin-targeted dendrimer-based probe for dual-modality magnetic resonance and fluorescence imaging. *Bioconjugate chemistry*, ACS Publications, v. 18, n. 5, p. 1474–1482, 2007.

[71] KOBAYASHI, H. et al. Multimodal nanoprobes for radionuclide and five-color near-infrared optical lymphatic imaging. *ACS nano*, ACS Publications, v. 1, n. 4, p. 258–264, 2007.

[72] BOSWELL, C. A. et al. Synthesis, characterization, and biological evaluation of integrin $\alpha v\beta 3$-targeted pamam dendrimers. *Molecular pharmaceutics*, ACS Publications, v. 5, n. 4, p. 527–539, 2008.

[73] XU, H. et al. Toward improved syntheses of dendrimer-based magnetic resonance imaging contrast agents: new bifunctional diethylenetriaminepentaacetic acid ligands and nonaqueous conjugation chemistry. *Journal of medicinal chemistry*, ACS Publications, v. 50, n. 14, p. 3185–3193, 2007.

[74] KOBAYASHI, H. et al. 3d-micro-mr angiography of mice using macromolecular mr contrast agents with polyamidoamine dendrimer core with reference to their pharmacokinetic properties. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, Wiley Online Library, v. 45, n. 3, p. 454–460, 2001.

[75] KOBAYASHI, H. et al. Novel liver macromolecular mr contrast agent with a polypropylenimine diaminobutyl dendrimer core: comparison to the vascular mr contrast agent with the polyamidoamine dendrimer core. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, Wiley Online Library, v. 46, n. 4, p. 795–802, 2001.

[76] KOBAYASHI, H. et al. 3d mr angiography of intratumoral vasculature using a novel macromolecular mr contrast agent. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, Wiley Online Library, v. 46, n. 3, p. 579–585, 2001.

[77] BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear programming: theory and algorithms*. [S.l.]: John Wiley & Sons, 2013.

[78] DELAI, A. L. et al. Análise de computação natural aplicada ao projeto otimizado de antenas de microfita circularmente polarizadas. [sn], 2020.

[79] ZUBEN, F. J. V. Computação evolutiva: uma abordagem pragmática. *Tutorial: Notas de Aula da disciplina IA707, Faculdade de Engenharia Elétrica e de Computação-Universidade Estadual de Campinas*, 2000.

[80] FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. Artificial intelligence through simulated evolution. Wiley, 1966.

[81] SANTINI, C. C.; PACHECO, M. A. C.; AMARAL, J. F. M. do. Síntese evolucionária em nanotecnologia.

[82] SLOWIK, A.; KWASNICKA, H. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, Springer, p. 1–17, 2020.

[83] DUDDECK, F. Multidisciplinary optimization of car bodies. *Structural and Multidisciplinary Optimization*, Springer, v. 35, n. 4, p. 375–389, 2008.

[84] COELLO, C. A. C.; LAMONT, G. B. *Applications of multi-objective evolutionary algorithms*. [S.l.]: World Scientific, 2004.

[85] DASGUPTA, D.; MICHALEWICZ, Z. *Evolutionary algorithms in engineering applications*. [S.l.]: Springer Science & Business Media, 2013.

[86] SIEJA, M.; WACH, K. The use of evolutionary algorithms for optimization in the modern entrepreneurial economy: Interdisciplinary perspective. *Entrepreneurial Business and Economics Review*, Cracow University of Economics, v. 7, n. 4, p. 117–130, 2019.

[87] MITCHELL, M. *An introduction to genetic algorithms*. [S.l.]: MIT press, 1998.

[88] MIRJALILI, S. et al. Genetic algorithm: Theory, literature review, and application in image reconstruction. In: *Nature-Inspired Optimizers*. [S.l.]: Springer, 2020. p. 69–85.

[89] MILLER, B. L.; GOLDBERG, D. E. et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, [Champaign, IL, USA: Complex Systems Publications, Inc., c1987-, v. 9, n. 3, p. 193–212, 1995.

[90] GOLDBERG, D. E. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, Complex Systems Publications, Inc., v. 4, p. 445–460, 1990.

[91] HUTTER, M. Fitness uniform selection to preserve genetic diversity. In: IEEE. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. [S.l.], 2002. v. 1, p. 783–788.

[92] SYSWERDA, G. A study of reproduction in generational and steady-state genetic algorithms. In: *Foundations of genetic algorithms*. [S.l.]: Elsevier, 1991. v. 1, p. 94–101.

[93] SEMENKIN, E.; SEMENKINA, M. Self-configuring genetic algorithm with modified uniform crossover operator. In: SPRINGER. *International Conference in Swarm Intelligence*. [S.l.], 2012. p. 414–421.

[94] HU, X.-B.; PAOLO, E. D. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: *Multi-objective memetic algorithms*. [S.l.]: Springer, 2009. p. 71–89.

[95] TSUTSUI, S.; YAMAMURA, M.; HIGUCHI, T. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. [S.l.: s.n.], 1999. p. 657–664.

[96] LOUIS, S. J.; RAWLINS, G. J. Designer genetic algorithms: Genetic algorithms in structure design. In: *ICGA*. [S.l.: s.n.], 1991. p. 53–60.

[97] SPEARS, W. M.; JONG, K. D. D. *On the virtues of parameterized uniform crossover*. [S.l.], 1995.

[98] MEHRA, M. et al. Study on different crossover mechanisms of genetic algorithm for test interval optimization for nuclear power plants. *International Journal of Intelligent Systems and Applications*, v. 6, n. 1, p. 20–28, 2014.

[99] HINTERDING, R. Gaussian mutation and self-adaption for numeric genetic algorithms. In: IEEE. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. [S.l.], 1995. v. 1, p. 384.

[100] MAITI, A.; MAITI, M. Discounted multi-item inventory model via genetic algorithm with roulette wheel selection, arithmetic crossover and uniform mutation in constraints bounded domains. *International Journal of Computer Mathematics*, Taylor & Francis, v. 85, n. 9, p. 1341–1353, 2008.

[101] ARZAMENDIA, M. et al. Evolutionary path planning of an autonomous surface vehicle for water quality monitoring. In: IEEE. *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*. [S.l.], 2016. p. 245–250.

[102] STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer, v. 11, n. 4, p. 341–359, 1997.

[103] MENDES, R.; MOHAIS, A. S. Dynde: a differential evolution for dynamic optimization problems. In: IEEE. *2005 IEEE Congress on Evolutionary Computation*. [S.l.], 2005. v. 3, p. 2808–2815.

[104] KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. *Proceedings of ICNN'95-International Conference on Neural Networks*. [S.l.], 1995. v. 4, p. 1942–1948.

[105] SERGIO, A. T. *Otimização de Reservoir Computing com PSO*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2013.

[106] POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization. *Swarm intelligence*, Springer, v. 1, n. 1, p. 33–57, 2007.

[107] VENTER, G.; SOBIESZCZANSKI-SOBIESKI, J. Particle swarm optimization. *AIAA journal*, v. 41, n. 8, p. 1583–1589, 2003.

[108] SHI, Y.; EBERHART, R. C. Empirical study of particle swarm optimization. In: IEEE. *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. [S.l.], 1999. v. 3, p. 1945–1950.

[109] BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: IEEE. *2007 IEEE swarm intelligence symposium*. [S.l.], 2007. p. 120–127.

[110] HANSEN, N.; OSTERMEIER, A. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: IEEE. *Proceedings of IEEE international conference on evolutionary computation*. [S.l.], 1996. p. 312–317.

[111] HANSEN, N. et al. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*. [S.l.: s.n.], 2010. p. 1689–1696.

[112] HANSEN, N.; OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, MIT Press, v. 9, n. 2, p. 159–195, 2001.

[113] HANSEN, N.; MÜLLER, S. D.; KOUMOUTSAKOS, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, MIT Press, v. 11, n. 1, p. 1–18, 2003.

[114] PEREIRA, M. V. W. et al. Otimização multimodal para domínio contínuo com heurísticas de agrupamento adaptativo. 2015.

[115] BEYER, H.-G.; SCHWEFEL, H.-P. Evolution strategies–a comprehensive introduction. *Natural computing*, Springer, v. 1, n. 1, p. 3–52, 2002.

[116] STRICKLER, A. S. Análise do impacto das estratégias de seleção de tradicionais moeas em moedas: Cma-es e umda.

[117] MELO, M. G. d. *Adaptação local da matriz de covariância guiada por mecanismos de exploração em estratégias de evolução*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2019.

[118] SIPPER, M. et al. Investigating the parameter space of evolutionary algorithms. *BioData mining*, Springer, v. 11, n. 1, p. 1–14, 2018.

[119] BREST, J. et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, IEEE, v. 10, n. 6, p. 646–657, 2006.