UNIVERSIDADE ESTADUAL PAULISTA - UNESP

Instituto de Biociências, Letras e Ciências Exatas - campus de São José do Rio Preto

VITORIA ZANON GOMES

Algoritmo genético híbrido com meta-heurísticas e aprendizado por reforço para alinhamento múltiplo de sequências

São José do Rio Preto

2025

Vitoria Zanon Gomes

Algoritmo genético híbrido com meta-heurísticas e aprendizado por reforço para alinhamento múltiplo de sequências

Tese apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, campus de São José do Rio Preto, para obtenção do título de Doutora em Ciência da Computação.

Área de Concentração: Computação Aplicada

Orientador: Profº Dr. Geraldo Francisco Donegá Zafalon

São José do Rio Preto
2025

G633a

Gomes, Vitoria Zanon
Algoritmo genético híbrido com meta-heurísticas e aprendizado por reforço para alinhamento múltiplo de sequências / Vitoria Zanon Gomes. -- São José do Rio Preto, 2025
77 p.

Tese (doutorado) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Geraldo Francisco Donegá Zafalon

1. Bioinformática. 2. Alinhamento de Sequências. 3. Algoritmo genético. 4. Meta-heurísticas. 5. Aprendizado por reforço. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

IMPACTO POTENCIAL DESTA PESQUISA

O impacto esperado desta pesquisa é a geração de alinhamentos múltiplos com qualidade biológica significativa, auxiliando profissionais que trabalham em tópicos como *drug design*, identificação de patogênicos, estudos evolutivos e predição de estruturas de proteínas recém descobertas a aprimorar seus estudos com melhores alinhamentos, impactando positivamente tratamentos, fármacos, medidas sanitárias e ambientais que buscam melhorar o conhecimento humano acerca da vida e garantir saúde e qualidade de vida para a população.

POTENTIAL IMPACT OF THIS RESEARCH

The expected impact of this research is the generation of multiple sequence alignments with significant biological quality, aiding professionals working in areas such as drug design, pathogen identification, evolutionary studies and prediction of newly discovered protein structures to enhance their research with better alignments, causing positive influence in treatments, pharmaceuticals, public health and environmental measures aimed at improving human knowledge about life and ensuring health and quality of life for the population.

VITORIA ZANON GOMES

ALGORITMO GENÉTICO HÍBRIDO COM META-HEURÍSTICAS E APRENDIZADO POR REFORÇO PARA ALINHAMENTO MÚLTIPLO DE SEQUÊNCIAS :

Tese apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, campus de São José do Rio Preto, para obtenção do título de Doutora em Ciência da Computação.

Área de Concentração: Computação Aplicada

Data de defesa: 01/07/2025

BANCA EXAMINADORA

_____

Profº Dr. Geraldo Francisco Donegá Zafalon
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

_____

Profº Dr. Danilo Medeiros Eler
UNESP – Faculdade de Ciências e Tecnologia de Presidente Prudente

_____

Profº Dr. Carlos Roberto Valêncio
UNESP – Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto

_____

Profº Dr. Jorge Rady de Almeida Junior
USP – Departamento de Engenharia de Computação e Sistemas Digitais

_____

Profº Dr. Pedro Luiz Pizzigatti Corrêa
USP – Departamento de Engenharia de Computação e Sistemas Digitais

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Ana e Robson, por todo o incentivo aos meus estudos; à minha irmã Melina por todo o carinho, incentivo e por todas as vezes que me ouviu falar sobre meu projeto mesmo não entendendo muito, e ao meu cunhado Leandro e meu padrinho Luis Afonso, pelo carinho e apoio.

Agradeço ao Gustavo, meu *muchacho*, por todo o apoio e carinho durante esse longo processo. Obrigada pela paciência de me ouvir falar várias vezes sobre as minhas dificuldades, pela ajuda com a organização do código, por todos os sorvetes e macarronadas ao longo desses quatro anos e por estar sempre ao meu lado, independente da circunstância. Obrigada por me incentivar a sonhar cada dia mais alto e nunca desistir. Te amo muito.

Ao meu orientador, Prof. Geraldo, deixo aqui diversos agradecimentos. Agradeço por todo o apoio durante esses sete anos, desde o dia em que pedi para fazer parte do nosso querido laboratório de bioinformática; por sempre me incentivar e não me permitir desanimar pois, em suas palavras, "tudo sempre dá certo no final"; por confiar em mim e no meu trabalho a ponto de solicitar minha transferência para o doutorado direto, por me ajudar a crescer profissionalmente, me proporcionando diversas oportunidades únicas, e por ser, além de tudo, um grande amigo com quem sei que posso sempre contar. Obrigada por tudo, professor.

Também agradeço à Beatriz, também conhecida apenas por mim como Bianca, por todo o apoio, carinho, amor e conversas, das mais profundas às mais fúteis. Estou um passo mais perto da minha calça de balinhas, e você tem papel nisso. Obrigada, *best fren*.

Agradeço aos meus amigos Bárbara "Babs", Danika, Lara, Luís, Isabela, Pedro e à minha prima Andressa pelos anos de amizade e carinho. Vocês tornaram, cada um à sua maneira, essa jornada mais leve.

Deixo também meu agradecimento à Débora, que nos últimos um ano e meio me ouviu falar por horas sobre essa pesquisa e me ajudou a ser uma cientista e pessoa melhor.

Agradeço também às pessoas queridas que o Ibilce colocou na minha vida: Bruno, Luiza, Gabriel Prevato, Gabriel Leoni, Muriel, Matheus Andrade e Gabriel Covello. Obrigada pelas conversas e por toda a ajuda quando precisei.

Agradeço também ao Prof. Carlos Roberto Valêncio pela oportunidade de começar a atuar em sala de aula, por confiar no meu trabalho e por me ajudar todas as vezes em que precisei, e ao Prof. Allan Contessoto por me auxiliar e orientar não só durante meu estágio docência como depois e em todo o resto, além de ser uma pessoa incrível que me inspira desde as suas aulas durante minha graduação.

Agradeço à Unesp/Ibilce e a todos os docentes e funcionários, especialmente aos do Departamento de Ciências da Computação e Estatística (DCCE) por todo o suporte durante minha graduação e doutorado. Também agradeço ao Prof.Danilo Eler e à Aline, ambos da Unesp/FCT de Presidente Prudente pelo apoio e ajuda durante minha atuação como representante discente.

Agradeço também aos professores que compuseram minha formação durante os anos anteriores à Unesp, desde o jardim de infância até o ensino médio. Em especial, agradeço aos professores Marisa, Daniela, Omair, Adriana, Alexandre, Eliani, Fausto, Sandra e Rosicler. Vocês foram os primeiros a verem minha paixão pelo conhecimento e pela ciência e me incentivaram ao máximo, sempre com brilho nos olhos, a seguir essa carreira incrível. Com a mesma paixão e dedicação que vocês responderam todas as minhas perguntas, mesmo as mais complexas, eu sigo adiante, e busco em vocês a inspiração para ser uma professora melhor dia após dia em sala de aula.

Por fim, ao Núcleo de Computação Científica (NCC/GridUNESP) da Universidade Estadual Paulista (UNESP) pelos recursos computacionais disponibilizados que tornaram este trabalho possível.

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

“Remember whene’er you hear this song, life is a trip full of discovery.“
- Kapp’n (NINTENDO, 2020).

RESUMO

O alinhamento múltiplo de sequências é uma das tarefas mais importantes dentro do campo da bioinformática, sendo usado para diversos tipos de análises biológicas, como a de função e estrutura de proteínas desconhecidas, *design* de drogas e estudos evolucionários. A literatura apresenta diversas estratégias para a realização de um alinhamento com qualidade biológica significante, sendo o algoritmo genético um dos mais usados devido à sua adaptabilidade. Porém, apesar dos bons resultados produzidos, o algoritmo sofre com o chamado problema de máximo local, fazendo com que a solução final seja um alinhamento que ainda pode ser melhorado. Dessa forma, o presente trabalho propõe a modelagem e implementação de uma estratégia híbrida entre o algoritmo genético, meta-heurísticas conhecidas da área e técnicas de aprendizado de máquina por reforço, com o objetivo de amenizar essa dificuldade. Com os resultados obtidos, demonstra-se a eficácia da estratégia proposta em amenizar consideravelmente o problema de máximo local, produzindo resultados com ótima significância biológica, semelhante às principais ferramentas da área, como Clustal Omega e Kalign.

Palavras-Chave: bioinformática; alinhamento de sequências; algoritmo genético; meta-heurísticas; aprendizado por reforço.

ABSTRACT

The multiple sequence alignment is one of the most important tasks in bioinformatics, being used for different types of biological analysis such as the function and structure of unknown proteins, drugs design and evolutionary studies. There are several distinct strategies to get an alignment with significant biological quality, being the genetic algorithm one of the most used due to its adaptability. However, it suffers with the local maximum problem, causing the final solution to be an alignment that could be improved. Thus, this work proposes the modelling and implementation of a new hybrid method between genetic algorithm, well-known metaheuristics and machine reinforcement learning techniques as a way to smooth this problem. The results demonstrate the efficacy of the proposed strategy in mitigating considerably the local optima problem, achieving alignments with great biological significance, which are comparable with leading state-of-art tools, like Clustal Omega and Kalign.

Keywords: bioinformatics; multiple sequence alignment; genetic algorithm; metaheuristics; reinforcement learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABC	Artificial Bee Colony algorithm
ACO	Ant Colony Optimization
AG	Algoritmo Genético
AMS	Alinhamento Múltiplo de Sequências
AP	Alinhamento Progressivo
BLOSUM	Block Substituition Matrix
COFFEE	Consistency based Ojective Function For alignmEnt Evalutation
DNA	DeoxyriboNucleic Acid/Ácido Desoxirribonucleico
FOA	Fruit fly Optimization Algorithm
NGS	Next Generation Sequencing
PAM	Point Accepted Mutation
PCA	Principal Component Analysis
PD	Programação Dinâmica
PSO	Particle Swarm Optimization
Q	Quality score
RNA	RiboNucleic Acid - Ácido Ribonucleico
SAGA	Sequence Alignment by Genetic Algorithm
SOP	Sum of Pairs
SVM	Support Vector Machine
TC	Total Column score

SUMÁRIO

1 INTRODUÇÃO

A bioinformática caracteriza-se como a área que une conhecimentos e técnicas computacionais com atividades de cunho biológico, a fim de auxiliar profissionais como biólogos e biomédicos em suas tarefas com os benefícios da rapidez e precisão computacional (Khuri, 2008). Dentre essas tarefas, destacam-se o sequenciamento (Hu et al., 2021b), análise filogenética (Dunn; Luo; Wu, 2013), reconhecimento de padrões (Khan; He; Valverde, 2016), *design* de drogas (Macalino et al., 2015) e o alinhamento de sequências (Zafalon et al., 2021; Edgar; Batzoglou, 2006).

O alinhamento de sequências consiste no rearranjo das bases presentes em um grupo de sequências, sejam elas de DNA, RNA ou proteínas, de modo a maximizar a semelhança entre elas (Amorim et al., 2018). As aplicações que se beneficiam dessa técnica se encontram no contexto de estudos genômicos e estão relacionadas à questões como predição de estruturas e funções de proteínas (Edgar; Batzoglou, 2006; Lalwani; Sharma, 2019), estudos evolucionários (Kumar; Stecher; Tamura, 2016), desenho de drogas (Arcuri et al., 2010), construção de árvores filogenéticas (Feng; Doolittle, 1987), entre outras.

Para realizar um alinhamento de maneira consistente é preciso estabelecer previamente critérios a serem seguidos, o que levou ao desenvolvimento de algoritmos próprios para esse problema. Os primeiros métodos desenvolvidos foram os algoritmos de Needleman-Wunsch (1970) e de Smith-Waterman (1981). Ambos algoritmos são de natureza determínisca, utilizando programação dinâmica (PD) para a obtenção de um resultado preciso (Amorim et al., 2018). Porém, devido à complexidade envolvida no uso da PD, esses algoritmos são utilizados somente para pares de sequências, sendo proibitiva a aplicação dos mesmos em conjuntos com mais de duas sequências (Wang; Jiang, 1994).

Com o advento dos sequenciadores automáticos a necessidade de se analisar três ou mais sequências simultaneamente cresceu, o que levou ao desenvolvimento de algoritmos que conseguissem suprir a demanda pelos chamados alinhamentos múltiplos de sequências (AMS) (Edgar; Batzoglou, 2006). Como o nome sugere, um algoritmo de AMS alinha três ou mais sequências ao mesmo tempo e, ao contrário dos algoritmos de alinhamento de pares, possuem base estocástica, tendo como objetivo a obtenção de um resultado de boa qualidade e com significância biológica relevante, embora aproximado, em um tempo hábil (Nute; Saleh; Warnow, 2019).

Na literatura atual é possível encontrar uma diversidade de heurísticas e meta-heurísticas utilizadas para o problema de AMS, como o alinhamento progressivo (AP) (Sievers; Higgins, 2018), transformada de Fourier (Katoh et al., 2002), e os algoritmos inspirados pela natureza, como recozimento simulado (Kim; Pramanik; Chung, 1994; Correa et al., 2012), algoritmo de enxame artificial de abelhas (ABC, do inglês *Artificial Bee Colony*) (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016), algoritmo de otimização por enxame de partículas (PSO, do inglês *Particle Swarm Optimization*) (Chaabane, 2018) e o algoritmo genético (AG) (Gondro;

Kinghorn, 2007; Kaya; Kaya; Alhajj, 2016; Zafalon et al., 2021).

Apesar do AP ser a base de diversas ferramentas muito utilizadas, como a família Clustal (Thompson; Higgins; Gibson, 1994; Sievers; Higgins, 2018), a MUSCLE (Edgar, 2004) e a Kalign (Lassmann, 2020), essa abordagem sofre com o problema de propagação de erro. Isso significa que erros cometidos nas etapas iniciais não podem ser corrigidos ao longo da execução, deteriorando a qualidade do resultado final (Gondro; Kinghorn, 2007).

Enquanto isso o AG, conhecido pela sua adaptabilidade, não possui esse tipo de dificuldade, tornando possível a obtenção de bons resultados até mesmo quando sequências dissimilares são dadas como entrada (Gondro; Kinghorn, 2007). Essa característica advém do fato de que este algoritmo é inspirado na teoria da evolução, proposta por Darwin, onde indivíduos de uma população são expostos a processos de mutação e recombinação gênica ao longo de gerações e, com a ação da seleção natural, somente os melhor adaptados sobrevivem (Kaya; Kaya; Alhajj, 2016). Diversas ferramentas implementam o AG, sendo as mais conhecidas a SAGA (Notredame; Higgins, 1996) e a MSA-GA (Gondro; Kinghorn, 2007).

Mesmo com seu bom desempenho, o AG não está livre de desvantagens. Devido à sua natureza iterativa, ele pode facilmente adentrar um ponto de máximo local, o que impossibilita a busca por alinhamentos melhores e deteriora a qualidade de seus resultados (Lee et al., 2008). Essa dificuldade é intrínseca a métodos iterativos, porém se mostra presente com maior frequência no AG, prejudicando a busca por resultados de alta qualidade.

1.1 MOTIVAÇÃO

A obtenção de melhores alinhamentos realizados computacionalmente em tempo hábil é de suma importância, considerando a gama de atividades que se servem dos mesmos. Essas atividades realizadas por geneticistas, biomédicos e demais profissionais da área genômica trazem importante retorno para a sociedade, seja em forma de novos fármacos ou tratamentos para doenças conhecidas, ou em combate a crises sanitárias como a de COVID-19.

1.2 OBJETIVO

O objetivo do presente trabalho é melhorar a qualidade dos alinhamentos produzidos pelo algoritmo genético ao amenizar a ocorrência do problema de máximo local por meio da hibridização de técnicas realizada em duas etapas:

- Aplicação de um conjunto de meta-heurísticas para a geração de uma população inicial, impulsionando a ação do AG já no início de sua execução;
- Utilização de técnicas de aprendizado por reforço junto à execução do algoritmo para auxiliar na tomada de decisão acerca de quais operadores de mutação e recombinação gênica devem ser usados durante cada geração do AG;

1.3 JUSTIFICATIVA

É possível encontrar na literatura diversos trabalhos bem-sucedidos que justificam a hipótese do presente trabalho.

Dentro do escopo da hibridização, trabalhos como o de Lee et al. (2008), Zafalon et al. (2021) e Rubio-Largo, Vega-Rodríguez e González-Álvares (2016) aplicam essa técnica para resolver o problema de máximo local de algoritmos de AMS. Apesar dos bons resultados obtidos, os autores dos trabalhos observam uma degradação do resultado e/ou tempo de execução causada pela ação do novo método inserido no algoritmo, e sugerem que isso pode ser facilmente contornado ao acoplar uma terceira estratégia (Zafalon et al., 2015).

A hibridização também é observada em outros campos da bioinformática, como na predição de genes (Keller et al., 2011), predição de estruturas e funções de proteínas desconhecidas (Klepeis; Pieja; Floudas, 2003; Pierri; Parisi; Porcelli, 2010), reforçando a eficácia da mesma, apesar dos contratempos.

A geração da população inicial é uma etapa de vital importância tanto para o AG quanto para outros métodos evolucionários pois é a partir desse primeiro conjunto de soluções que o espaço inicial de busca do algoritmo é definido. A ferramenta MSA-GA (Gondro; Kinghorn, 2007), por exemplo, realiza essa etapa alinhando todas as sequências de entrada em pares e selecionando uma sequência de cada alinhamento para formar os indivíduos que compõem a população inicial, porém também aceita que arquivos contendo pré-alinhamentos realizados por outros métodos sejam usados como população inicial. É observada melhoria na solução obtida quando os pré-alinhamentos são usados em relação ao uso dos alinhamentos de pares. Assim, pode-se afirmar que uma população inicial formada por alinhamentos com garantia de uma boa qualidade pode alavancar os resultados do algoritmo (Gondro; Kinghorn, 2007).

A influência dos operadores nos resultados obtidos pelo AG é discutida por Bajaj e Sangwan (2019) e Lim et al. (2017), que demonstram que o equilíbrio entre exploração e explotação está no bom uso dos operadores gênicos, propiciando melhores resultados e um aproveitamento melhor do espaço de busca. Desse modo, a utilização de técnicas de aprendizado de máquina que aprendam o comportamento dos operadores auxilia na decisão sobre qual deles utilizar no momento, visando o maior aproveitamento das vantagens de cada um.

Por fim, em trabalhos como o de Lalwani e Sharma (2019) e Zafalon et al. (2015), no entanto, demonstra-se e justifica-se a hipótese de que a paralelização de etapas do AMS pode solucionar o problema do tempo de execução relacionado à hibridização. Versões totalmente paralelizadas de estratégias conhecidas como o AG (Luo; Zhang; Liang, 2011), o algoritmo de otimização por colônia de formigas (ACO, do inglês *Ant Colony Optimization*) (Zhou et al., 2018) e o PSO (Lalwani; Sharma, 2019) também corroboram essa afirmação.

1.4 ORGANIZAÇÃO DO TRABALHO

Assim, o presente trabalho é organizado da seguinte forma: no capítulo 2 discorre-se detalhadamente sobre a fundamentação teórica assim como sobre os trabalhos correlatos e o estado da arte; no capítulo 3 a metodologia é apresentada, assim como os algoritmos utilizados para implementar a estratégia proposta; no capítulo 4 são apresentados os experimentos, forma de análise e resultados obtidos e, por fim, o capítulo 5 encontram-se as conclusões acerca do trabalho e proposições de trabalhos futuros.

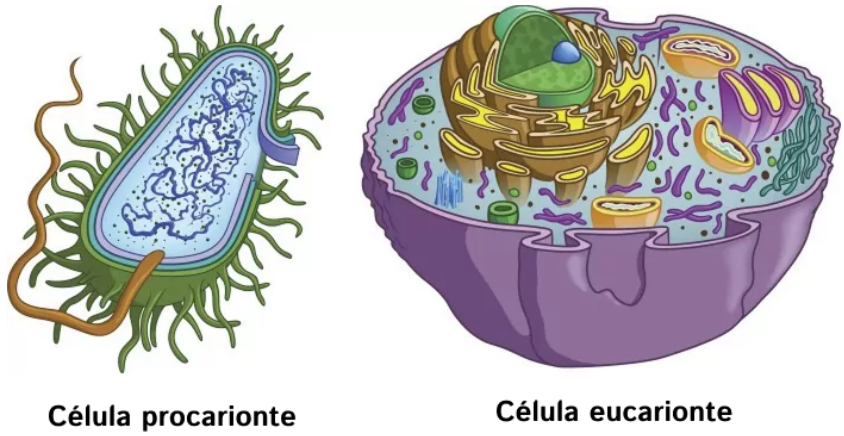
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão detalhados os conceitos teóricos necessários para o desenvolvimento deste trabalho, como biologia molecular, alinhamentos de sequências, meta-heurísticas, funcionamento do algoritmo genético, operadores de mutação e recombinação e aprendizado por reforço, além do estado da arte no que diz respeito ao uso de algoritmos bioinspirados para AMS.

2.1 CONTEXTO BIOLÓGICO

Todos os diversos organismos vivos existentes são compostos por uma mesma estrutura básica: células (Zaha et al., 2000). As células são responsáveis por todas as funções e estruturas dos seres vivos e são divididas em dois tipos: eucariontes e procariontes (Misteli, 2007; Zaha et al., 2000), ilustrados na Figura 1. A principal diferença entre os tipos é que as células eucariontes possuem um envoltório nuclear, mantendo o material nuclear reunido em um único lugar, enquanto as células procariontes tem seu material nuclear espalhado pelo citoplasma (Shafee; Lowe, 2017; Zaha et al., 2000).

Figura 1 – Célula procarionte e célula eucarionte.



Fonte: Retirado e adaptado de THIS ONE VS THAT ONE (2017)

Apesar das diferenças, ambos os tipos de células são constituídas pelos mesmos três tipos de polímeros: os polissacarídeos, os ácidos nucleicos e as proteínas (Zaha et al., 2000), sendo os dois últimos objetos de intenso estudo na bioinformática e melhor descritos a seguir.

2.1.1 Ácidos nucleicos: DNA e RNA

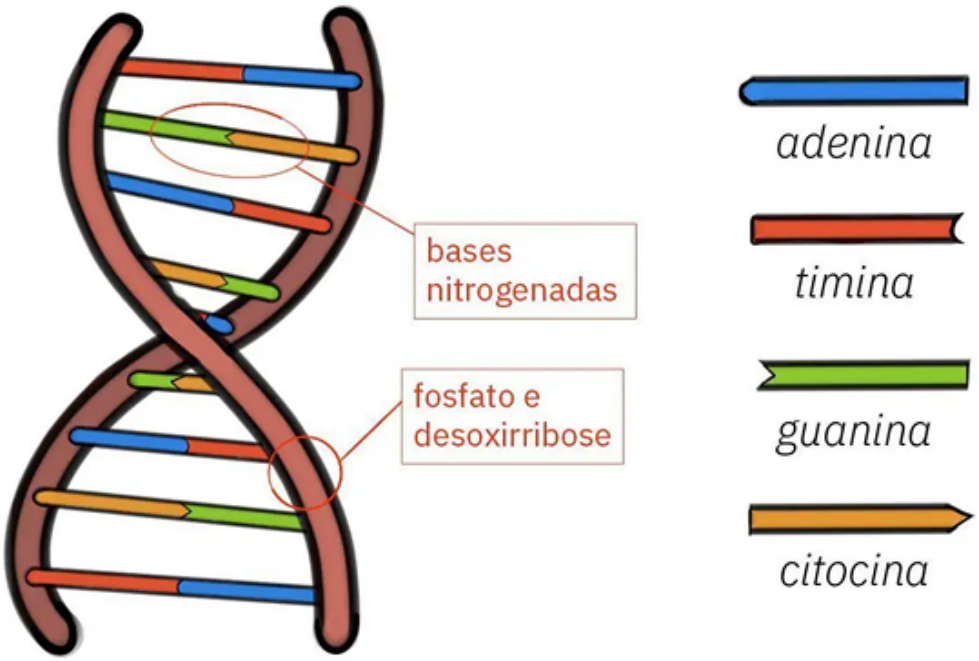
Os ácidos nucleicos são moléculas responsáveis por armazenar e transmitir todas as informações acerca dos organismos vivos, como quais proteínas sintetizar e quais suas funções (Zaha et al., 2000). Os dois tipos de ácidos nucleicos são o ácido desoxirribonucleico (DNA, do inglês

deoxyribonucleic acid) e o ácido ribonucleico (RNA, do inglês *ribonucleic acid*), e ambos são formados por cadeias lineares de nucleotídeos, também chamados de bases nitrogenadas, sendo de quatro tipos diferentes: Adenina (A), Timina (T), Citosina (C) e Guanina (G) nas moléculas de DNA e, nas moléculas de RNA, a Timina é substituída pela Uracila (U) (Zaha et al., 2000).

As cadeias formadas por essas letras são chamadas de sequências de nucleotídeos e, dentro do contexto dessas sequências, as letras são chamadas simplesmente de bases (Lemos; ao; Casanova, 2003), e podem representar desde trechos simples do chamado código genético até genomas inteiros.

O DNA é a principal molécula responsável por manter a informação genética dos seres vivos, sendo composto por duas fitas de bases nitrogenadas, formando uma estrutura conhecida como dupla-hélice, descrita pela primeira vez por Watson e Crick (1953). Nela, as bases de uma fita são ditas complementares umas as outras, de modo que um A em uma fita estará ligado a um T na outra fita, e um C estará ligado a um G, sendo o oposto também verdade. As sequências de nucleotídeos representam apenas uma das fitas, uma vez que é possível deduzir a outra por suas bases. Na Figura 2 ilustra-se a estrutura de dupla hélice.

Figura 2 – Ilustracão da dupla hélice.

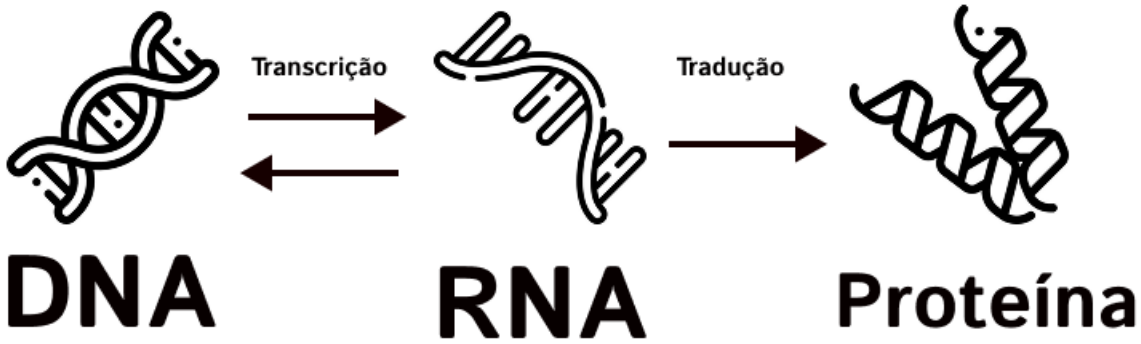


Fonte: Retirado de MEU DNA (2021).

Já o RNA é responsável pela expressão das características presentes no DNA, trazendo as informações ali presentes para fora do núcleo da célula, para que as organelas responsáveis possam executar suas funções (Alberts et al., 2002). Para entender melhor a função do RNA e seus diferentes tipos, é preciso entender o processo fundamental para a existência dos organismos que conhecemos: a síntese proteica.

A síntese proteica, dogma central da biologia molecular, é o processo pelo qual todas as diferentes proteínas que compõem um organismo são produzidas. Ela é composta, de um modo geral, por dois processos: transcrição e tradução (ver Figura 3) (Lemos; ao; Casanova, 2003).

Figura 3 – Síntese proteica: o dogma central da biologia molecular.



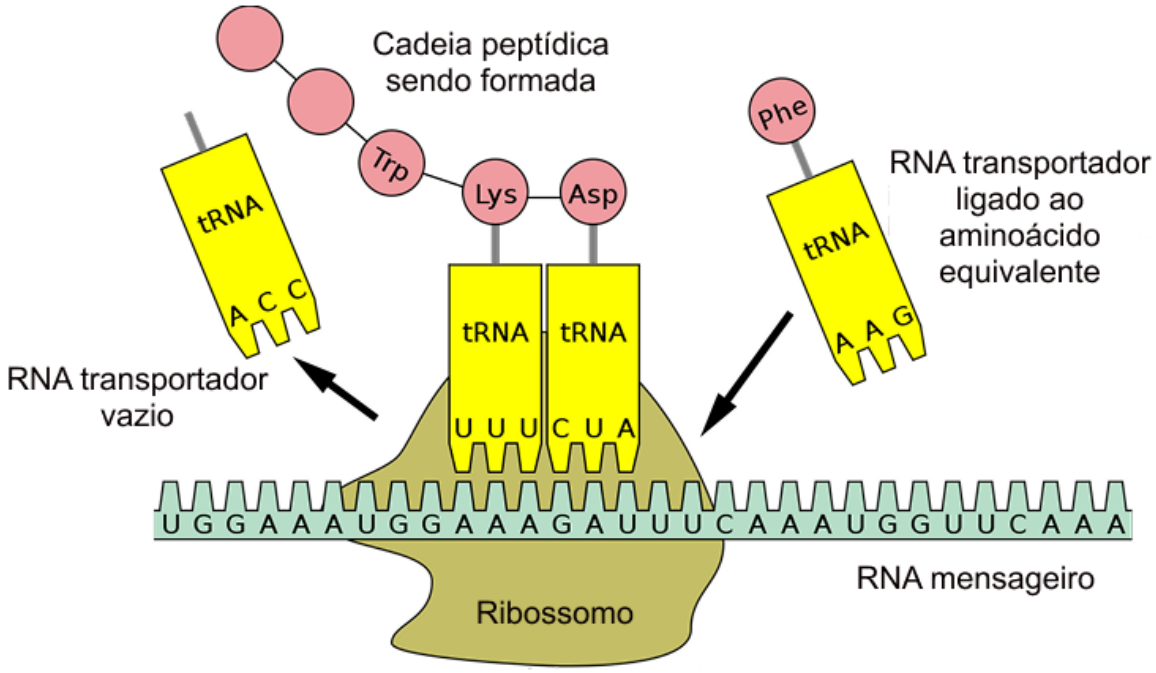
Fonte: Elaborado pela autora.

O primeiro processo é o responsável pelo surgimento do RNA. Nele, uma porção do DNA, geralmente equivalente a um gene, é copiada e convertida em uma sequência de RNA complementar à fita de DNA transcrita, substituindo a base T pela U (Alberts et al., 2002). O RNA gerado é conhecido como RNA mensageiro, ou mRNA, e é responsável por levar as informações contidas no DNA para fora do núcleo, permitindo que ela seja repassada sem que o DNA se exponha a possíveis danos fora da segurança do núcleo (Lemos; ao; Casanova, 2003; Alberts et al., 2002). Após a transcrição do DNA para mRNA, essa nova molécula é transportada para fora do núcleo, e nela estão contidas trios de bases, conhecidas como códons, responsáveis por determinar quais aminoácidos devem ser utilizados para sintetizar determinada proteína (quais aminoácidos existem e como compõem uma proteína será melhor detalhado na seção 2.1.2) (Lemos; ao; Casanova, 2003).

Para que os aminoácidos correspondentes sejam identificados e ligados entre si, caracterizando o processo de tradução, um outro tipo de RNA é utilizado, o RNA transportador, ou tRNA (Zaha et al., 2000; Lemos; ao; Casanova, 2003), capaz de representar apenas um aminoácido, ligando-se a ele em uma ponta de sua estrutura e se ligando ao códon presente no mRNA na outra ponta através da presença de um anticódon, que nada mais é que uma sequência de três bases complementadores as do códon alvo (Zaha et al., 2000). Assim, os aminoácidos são ligados sequencialmente, formando por fim as proteínas, conforme mostrado na Figura 4.

Vale ressaltar que existem diversos tipos de RNA além dos citados aqui, com o RNA ribossômico (rRNA), responsável por formar os ribossomos presentes na célula (Alberts et al., 2002).

Figura 4 – Processo de tradução.



Fonte: Retirado e traduzido de DTE (2024).

2.1.2 Proteínas

As proteínas são moléculas com estruturas características responsáveis por diversas funções dentro de um organismo, compostas por uma sequência de aminoácidos ligados entre si por meio de ligações pepitídicas (Lemos; Casanova, 2000). Existem na natureza um total de 20 aminoácidos – conforme a Tabela 1 – que, conforme a ordem em que se apresentam em uma sequência peptídica, dão origem a diversas proteínas diferentes em estrutura e função (Lemos; Casanova, 2000; Zaha et al., 2000).

As proteínas podem ser encontradas em quatro diferentes estruturas tridimensionais, que ajudam a determinar sua função. São elas (Zaha et al., 2000):

- **Estrutura primária:** forma mais simples, sendo correspondente apenas à cadeia sequencial de aminoácidos linearmente disposta. É a partir dela que as demais estruturas se moldam. Nela, a proteína ainda não é capaz de exercer sua função.
- **Estrutura secundária:** forma aonde dobras são formadas, transformando o arranjo espacial dos aminoácidos da cadeia central, mantendo o padrão da dobra ao longo de toda a proteína, podendo ser da forma de α-hélice (dobra onde os aminoácidos se curvam em torno de um eixo imaginário) ou folha β (dobra onde as curvaturas ocorrem na superfície da cadeia, se assemelhando à uma folha dobrada).

Aminoácido	Representação com 3 letras	Representação com 1 letra
Alanina	Ala	A
Cisteína	Cys/Cis	C
Aspartato (Ácido Aspártico)	Asp	D
Glutamato (Ácido Glutâmico)	Glu	E
Fenilalanina	Phe/Fen	F
Glicina	Gly/Gli	G
Histidina	His	H
Isoleucina	Ile	I
Lisina	Lys/Lis	K
Leucina	Leu	L
Metionina	Met	M
Asparagina	Asn	N
Prolina	Pro	P
Glutamina	Gln	Q
Arginina	Arg	R
Serina	Ser	S
Treonina	Thr/Tre	T
Valina	Val	V
Triptofano	Trp/Tri	W
Tirosina	Tyr/Tir	Y

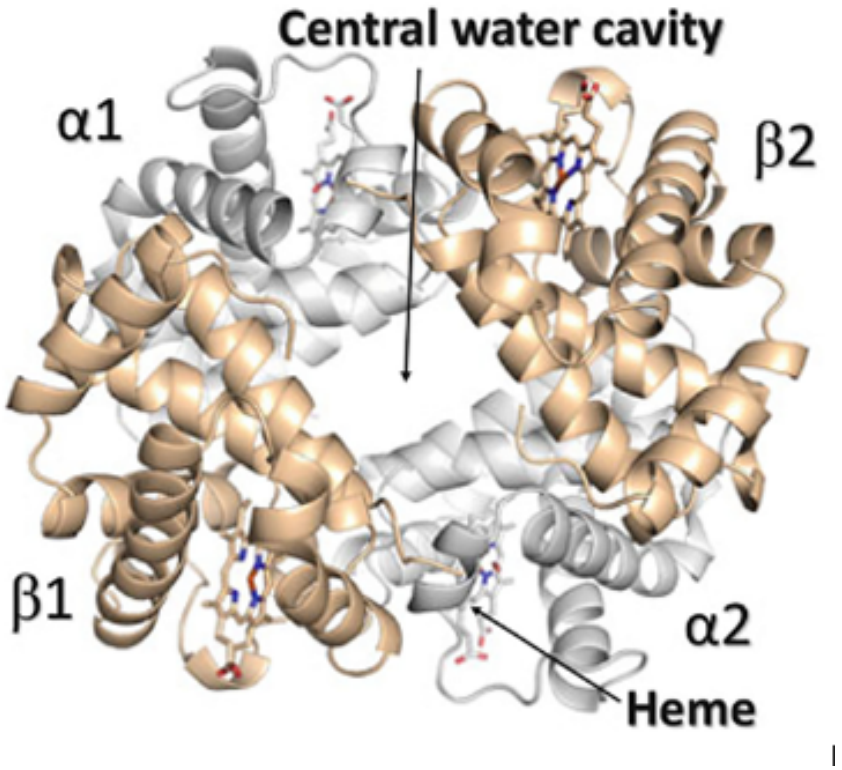
Tabela 1 – Tabela de aminoácidos encontrados na natureza.

- **Estrutura terciária:** forma assumida quando diversas estruturas secundárias se dispõem entre si após o processo de enovelamento, assumindo então sua forma funcional.
- **Estrutura quaternária:** forma onde diversas cadeias polipeptídicas se unem, ou seja, diversos grupos de aminoácidos diferentes, em suas estruturas terciárias, se unem formando uma outra proteína. A quantidade de subunidades depende da proteína em questão, e nem todas as proteínas conhecidas possuem essa forma. Um exemplo de proteína que assume uma estrutura quaternária é a hemoglobina, proteína responsável pelo transporte de oxigênio no sangue e que é composta por quatro subunidades. Na Figura 5 apresenta-se a hemoglobina em sua estrutura quaternária.

Para ilustrar os quatro níveis de estruturas tridimensionais de proteínas, pode-se verificar a Figura 6.

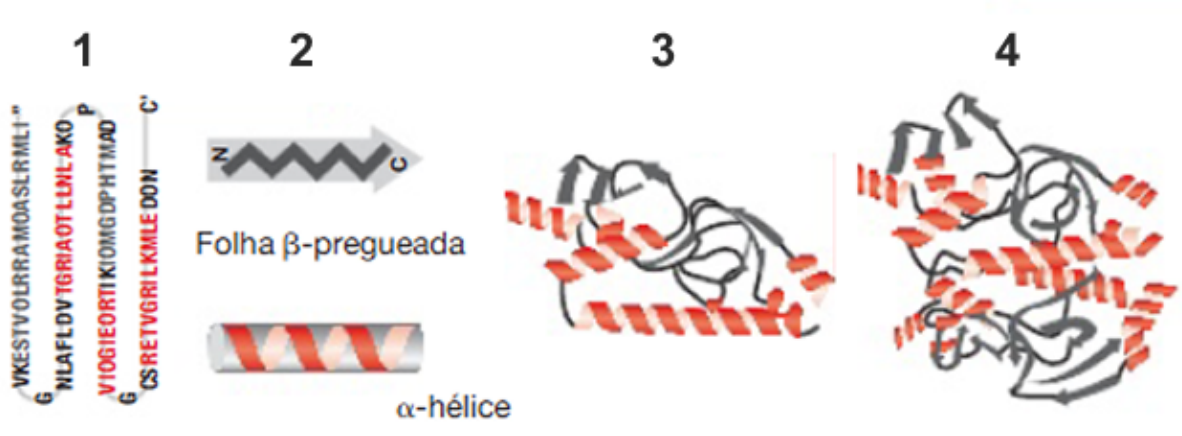
Até meados dos anos 1970 a obtenção de sequências, tanto de nucleotídeos (DNA e RNA) quanto de aminoácidos era um trabalho custoso, demandando uma quantidade alta de recursos humanos e materiais, porém essa situação se alterou graças ao surgimento do método de Sanger (Sanger; Coulson, 1975), que permitiu o sequenciamento eficaz de sequências de nucleotídeos

Figura 5 – Exemplo de proteína que possui uma estrutura quaternária: hemoglobina. Cada estrutura em branco e dourado representa uma estrutura terciária que compõe a estrutura quaternária, definindo a proteína.



Fonte: Ahmed, Ghatge e Safo, (2020).

Figura 6 – Os quatro níveis de estruturas tridimensionais de proteínas.



Fonte: Retirado e adaptado de Zaha *et al.* (2000).

(Pareek; Smoczynski; Tretyn, 2011). No que diz respeito a proteínas, utiliza-se a espectrometria de massa para identificar os aminoácidos que as compõem (Mann; Hendrickson; Pandey, 2001). Com o advento desses métodos, o estudo de padrões em biossequências se aperfeiçoou, permitindo a identificação de genes, famílias de proteínas e estudos evolutivos mais completos, e diversas bases de dados públicas foram criadas a fim de facilitar a identificação de sequências já

estudadas e suas informações estruturais e funcionais (Lemos; ao; Casanova, 2003). Para poder estudar e analisar essas sequências, no entanto, foi necessário o aperfeiçoamento de uma outra tarefa conhecida da bioinformática: o alinhamento de sequências.

2.2 ALINHAMENTO DE SEQUÊNCIAS

O alinhamento de sequências, como já mencionado, é o rearranjo de duas ou mais sequências buscando maximizar suas similaridades. Esse rearranjo é realizado a partir da inserção de espaços (*gaps*) entre as bases segundo critérios estabelecidos previamente. Este processo possui um destaque importante na bioinformática por ser a base para diversas outras tarefas, como a predição de estruturas e funções de proteínas (Edgar; Batzoglou, 2006), construção de árvores filogenéticas (Feng; Doolittle, 1987), desenhos de drogas (Arcuri et al., 2010) e estudos evolucionários (Kumar; Stecher; Tamura, 2016), e por auxiliar na análise eficaz de padrões biológicos (Lemos; ao; Casanova, 2003), principalmente após o surgimento dos chamados métodos de sequênciamento de nova geração (NGS, do inglês *next-generation sequencing*), que causou um aumento massivo na quantidade de dados biológicos disponíveis, desde pequenos genes até genomas inteiros (Amorim, 2021).

Considerando um conjunto de entrada S $= \{s1, s2, ..., sn\}$ sobre um alfabeto $\sum = \{$A, T, C, G$\}$ para sequências de DNA e $\sum = \{$A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$\}$ para sequências de aminoácidos, Rúbio-Largo, Vega-Rodriguez e González-Álvarez (2016) definem um alinhamento de sequências como o conjunto S' $= \{s1', s2', ..., sn'\}$, onde todas as sequências possuem o mesmo comprimento, sobre o alfabeto $\sum$' = $\sum \cup (-)$, sendo -"o símbolo que representa os *gaps*. *Gaps* representam inserções e/ou exclusões realizadas nas sequências.

Um alinhamento também pode ser entendido como uma matriz onde as linhas representam as sequências de entrada e as colunas representam as bases alinhadas de cada sequência. Essa matriz é gerada a partir das inserções de *gaps* de modo a igualar o comprimento das sequências e otimizar métricas de qualidade, não sendo permitidas colunas contendo somente *gaps*. Os índices onde bases iguais coincidem são chamados de *matches*, e quando bases diferentes são alinhadas esse alinhamento é chamado de *mismatch* (Bawono et al., 2017; Amorim et al., 2021). Na Figura 7 apresenta-se um exemplo de alinhamento de sequências.

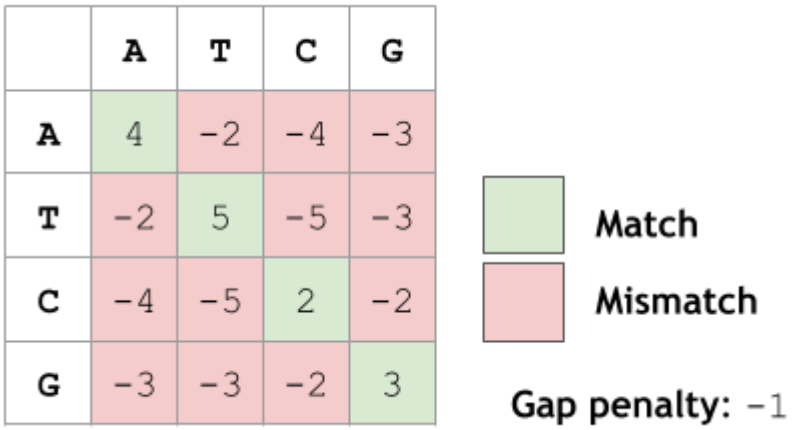
O primeiro algoritmo para execução de alinhamentos computacionalmente surgiu em 1970, desenvolvido por Needleman e Wunsch (1970). O algoritmo de Needleman-Wunsch, assim conhecido na literatura, é de característica determinística, fornecendo um resultado exato para o alinhamento global − ou seja, por todo o comprimento das sequências − do par de sequências fornecidas como entrada (Amorim et al., 2018). Sua execução é descrita a seguir:

- De ínicio, define-se os valores de pontuação para *matches*, *mismatches* e para a inserção de *gaps* através de uma matriz de pontuação, exemplificada na Figura 8.

Figura 7 – Um alinhamento de sequências.

Fonte: Elaborado pela autora.

Figura 8 – Exemplo de uma matriz de pontuação.



Fonte: Elaborado pela autora.

- Em seguida, uma matriz A de dimensões $M+1 \times N+1$ é criada, onde M é o comprimento da primeira sequência e N o comprimento da segunda sequência. Cada índice ij representa a pontuação obtida caso a base i da primeira sequência e a base j da segunda sequência coincidam. Essa pontuação é definida como:
  - Se $j = 0$:
  $$A(i,j) = gap * i \tag{1}$$
  - Se $i = 0$:
  $$A(i,j) = gap * j \tag{2}$$
  - Senão:
  $$A(i,j) = max \begin{cases} A(i-1,j-1) + match/mismatch \\ A(i,j-1) + gap \\ A(i-1,j) + gap \end{cases} \tag{3}$$

- Após preenchida toda a matriz, realiza-se a construção do alinhamento a partir do índice A(M,N), que segue até que se atinja $i = 0$ ou $j = 0$:
  - Se o primeiro caso da regra 3 foi utilizado para cálculo da pontuação do índice, a construção segue pelo índice $A(i-1, j-1)$ e o *match/mismatch* correspondente é adicionado ao alinhamento;
  - Se o segundo caso da regra 3 foi utilizado, a construção segue por $A(i, j-1)$ e o *gap* correspondente é adicionado à primeira sequência do alinhamento;
  - Por fim, se o terceiro caso da regra 3 foi utilizado, a construção segue por $A(i-1, j)$ e o *gap* correspondente é adicionado à segunda sequência do alinhamento.
- Ao final, temos o alinhamento global realizado. Na Figura 9 demonstra-se o processo de construção do alinhamento a partir da matriz.

Figura 9 – Visualização do Algoritmo de Needleman-Wunsch utilizando a pontuação da Figura 8.

		A	A	T	C	G
	0	-1	-2	-3	-4	-5
A	-1	4	3	2	1	0
G	-2	3	2	0	0	-1
G	-3	2	1	0	-1	3
C	-4	1	0	-1	2	2

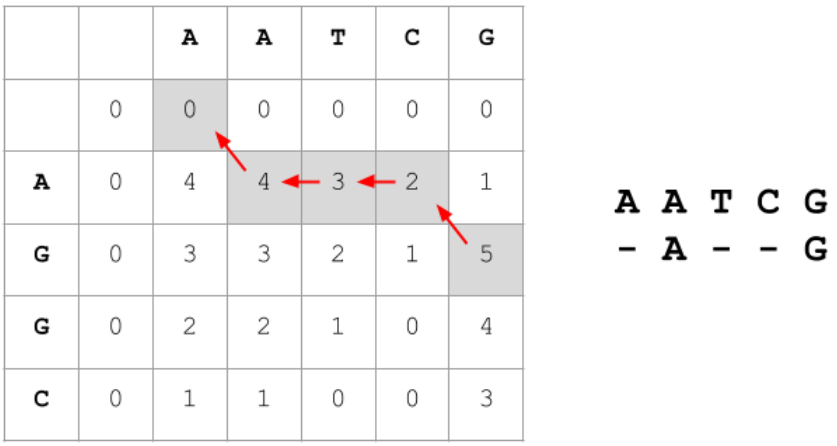
```
A A T C G -
- - A G G C
```

Fonte: Elaborado pela autora.

Com base na lógica anteriormente descrita, Smith e Waterman desenvolveram, em 1981, um algoritmo para alinhamento local − ou seja, quando somente o trecho de maior similaridade é dado como resultado (Smith; Waterman et al., 1981). A principal diferença desse algoritmo para o de Needleman-Wunsch é que, no anterior, pontuações negativas podem ser definidas (como uma penalidade por *mismatches*) e neste, caso a pontuação seja menor que zero, esta é redefinida para zero. Com isso, inicia-se a construção do alinhamento pelo índice com maior pontuação e o critério de parada se torna $A(i, j) = 0$, conforme ilustrado na Figura 10. Dessa forma, somente trechos com pontuações relevantes serão dados como resultado.

Por envolver o uso de programação dinâmica, cada sequência de entrada adiciona $O(n)$ à complexidade do algoritmo, sendo n o comprimento da maior sequência, já que cada sequência demanda uma dimensão a mais na matriz de cálculo. Dessa maneira, ao adicionarmos mais sequências ao conjunto de entrada, teríamos que lidar com um número relevantemente grande de

Figura 10 – Visualização do Algoritmo de Smith-Waterman utilizando a pontuação da Figura 8.



		A	A	T	C	G
	0	0	0	0	0	0
A	0	4	4	3	2	1
G	0	3	3	2	1	5
G	0	2	2	1	0	4
C	0	1	1	0	0	3

Fonte: Elaborado pela autora.

dimensões, fazendo com que o custo computacional dessas estratégias seja da ordem de $O(n^k)$, onde n é o tamanho da maior sequência e k é o número de sequências a serem alinhadas, e se torne proibitivo para $k >= 3$.

Wang e Jiang (1994) apresentam mais uma dificuldade em resolver problemas desse tipo. Em seu trabalho, os autores destacam a prova de que o ato de alinhar computacionalmente múltiplas sequências de maneira simultânea é um problema NP-Completo, o que significa que não há até o momento um algoritmo que resolva de forma exata e rápida essa questão.

Assim, considerando ambos os fatos citados e também o aumento da necessidade de se realizar esses alinhamentos, foram desenvolvidos os chamados algoritmos para alinhamento múltiplo de sequências.

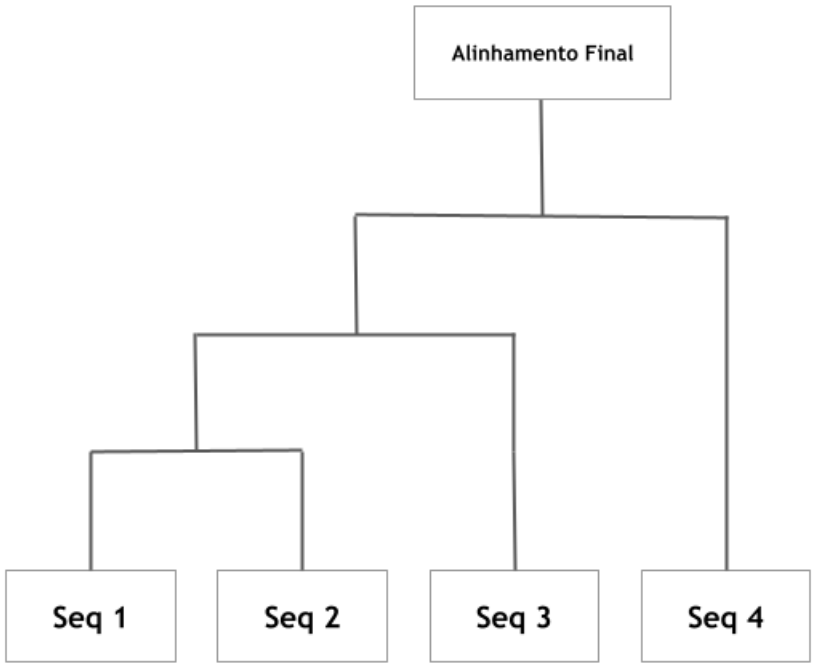
Esses algoritmos em sua maioria são de base probabilística, e podem ser vistos como dois grandes grupos: os algoritmos de alinhamento progressivo (Sievers; Higgins, 2018) e os algoritmos de alinhamento iterativo (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016).

Os algoritmos progressivos constroem o resultado final progressivamente, como o nome indica. Muitos utilizam uma estrutura de árvore para tal e, ao longo da execução, irão alinhar conjuntos menores de sequências e os combinarão ao final de cada etapa, culminando em um único trecho: o alinhamento completo (Hogeweg; Hesper, 1984; Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016). Na Figura 11 ilustra-se esse processo.

Devido à essa característica, erros cometidos nas etapas iniciais do alinhamento se propagam para as demais etapas, prejudicando a qualidade do alinhamento final. Mesmo assim, muitos profissionais abrem mão de uma resposta mais precisa em troca de um resultado em tempo ágil, o que é oferecido pelo AP. Exemplos de ferramentas baseadas em AP são a família Clustal (Thompson; Higgins; Gibson, 1994; Sievers; Higgins, 2018), a MUSCLE (Edgar, 2004) e a Kalign (Lassmann, 2020).

Algoritmos iterativos, por sua vez, trabalham com um ou mais alinhamentos iniciais advindos de outros métodos, como os progressivos, e os modificam ao longo das iterações, de modo a

Figura 11 – Ilustração do funcionamento de um algoritmo progressivo.



Fonte: Elaborado pela autora.

tentar melhorar o alinhamento em questão até que se atinja um critério de parada previamente estabelecido (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016). Para tal, muitas meta-heurísticas comumente usadas para outros problemas de otimização são aplicadas e serão discutidas mais detalhadamente na seção 2.3.

2.3 META-HEURÍSTICAS

Meta-heurísticas são uma classe de heurísticas amplamente aplicadas para problemas de otimização, provendo soluções aproximadas e em tempo hábil (Dokeroglu et al., 2019). Pesquisas nesse campo estão em alta devido à gama de problemas que podem ser resolvidos com essas estratégias, incluindo o AMS (Amorim et al., 2021). Como exemplo, pode-se citar os algoritmos baseados em inteligência de enxames (Katoch; Chauhan; Kumar, 2021) e os algoritmos evolutivos (Vikhar, 2016).

De uma forma geral, os algoritmos de inteligência de enxames são inspirados em comportamentos de espécies que convivem em sociedade, e trabalham com um conjunto de possíveis soluções e uma função objetivo usada para avaliar o quão próximo de uma resposta ótima as soluções estão. Dessa forma, o algoritmo trabalha manipulando essas candidatas à solução, alterando-as ao longo da execução, e a função objetivo avalia se ele está obtendo sucesso ou não. Uma vez que não há garantia de convergência para a melhor resposta, critérios de parada precisam ser estabelecidos, como um número máximo de iterações ou um teto para a pontuação das soluções (Neumann; Witt, 2010). Exemplos dessa classe de algoritmos são o PSO (Chaabane, 2018), o ABC (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016), o ACO (Lee et al.,

2008), o FOA e GWO (Jayapriya; Arock, 2015).

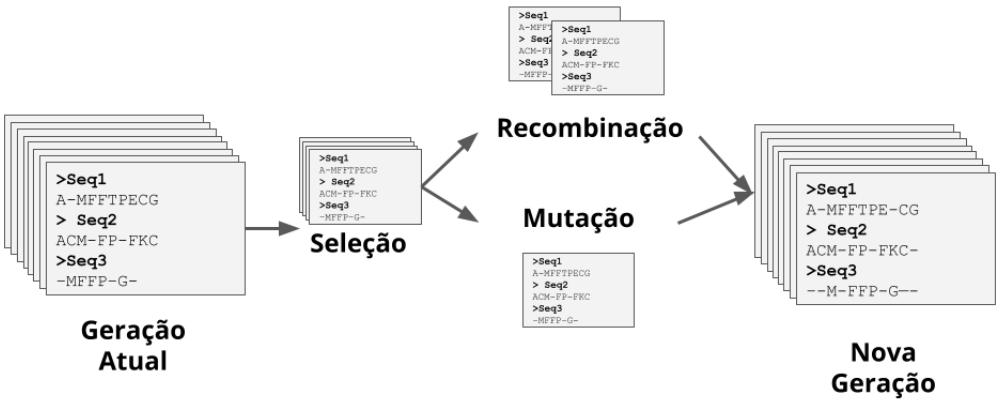
Os algoritmos evolutivos, por sua vez, se baseiam no mecanismo da evolução biológica, utilizando de conceitos como reprodução, mutação e seleção para varrer o espaço de busca visando encontrar melhores candidatos à solução, que são avaliados por uma função objetivo assim como nos algoritmos de inteligência de enxames. O algoritmo evolutivo mais conhecido é o algoritmo genético (Vikhar, 2016)

2.4 ALGORITMO GENÉTICO

O algoritmo genético é um algoritmo estocástico baseado na teoria da evolução de Charles Darwin (Kaya; Kaya; Alhajj, 2016). Primeiramente proposto por Holland (1975), o AG se tornou uma das meta-heurísticas mais utilizadas para AMS devido à sua adaptabilidade (Chowdhury; Garai, 2017).

Nele, cada possibilidade de alinhamento é denominada indivíduo, e o conjunto de indivíduos é chamado de população, sendo esse o objeto de trabalho do AG. Ao decorrer da execução a população é exposta a processos de mutação e recombinação gênica de modo que somente os mais adaptados seguem para a nova geração, passando pelo crivo da seleção natural e se reproduzindo, dando origem à geração seguinte (Chowdhury; Garai, 2017; Gomes et al., 2022). A responsável por aferir quais indivíduos estão melhor adaptados, ou seja, quais os alinhamentos com melhor qualidade, é a função objetivo (Amorim et al., 2018). Na Figura 12 pode-se verificar o funcionamento do AG, que é repetido iteração após iteração até que se atinja um critério de parada.

Figura 12 – Fluxograma da execução do AG.



Fonte: Elaborado pela autora.

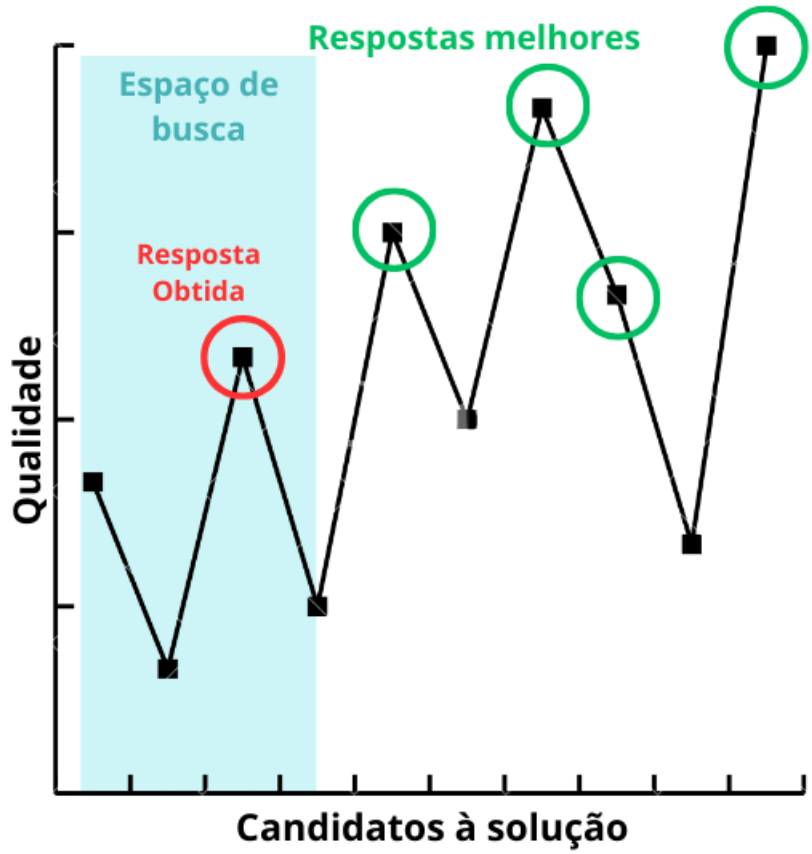
Na literatura é possível encontrar uma diversidade de ferramentas que implementam o AG para o problema de AMS, cada um com características específicas, sendo dentre as mais

conhecidas a SAGA (Notredame; Higgins, 1996) e a MSA-GA (Gondro; Kinghorn, 2007).

A ferramenta SAGA (do inglês, *Sequence Alignment by Genetic Algorithm*) (Notredame; Higgins, 1996) se destacou pelo seu conjunto de 22 operadores de mutação e recombinação gênica, ganhando prestígio dentre os profissionais da área. No entanto, estudos posteriores demonstraram que os resultados não são afetados pela complexidade dos operadores (Thomsen; Boomsma, 2004), abrindo espaço para que versões mais simples do AG fossem utilizadas, como a MSA-GA, que conta com apenas 3 operadores: 1 de mutação e 2 de recombinação gênica (recombinação vertical e horizontal) (Gondro; Kinghorn, 2007).

Porém, como todo algoritmo, o AG possui desvantagens. A mais notável delas é o problema de máximo local (Lee et al., 2008). Isso ocorre pois o algoritmo pode se deparar com um alinhamento localmente bom e não encontrar nenhum resultado de maior qualidade nas proximidades do mesmo no espaço de busca, ficando preso àquela solução local e a considerando como a solução ótima global. Mesmo sendo um problema intrínseco às meta-heurísticas devido a suas características, a estagnação em máximos locais se torna um evento especialmente recorrente no AG Uma vez que não é possível garantir a convergência do algoritmo para a resposta ótima. Na Figura 13 é apresentada a ilustração da situação descrita anteriormente.

Figura 13 – Ilustração do problema de máximo local.



Fonte: Elaborado pela autora.

A seguir, será discorrido sobre dois importantes recursos do AG: Os operadores e a função

objetivo.

2.4.1 Operadores

Os operadores do AG desempenham um importante papel no que diz respeito à qualidade final do resultado, trabalhando tanto na exploração e explotação realizada pelo algoritmo (Lim et al., 2017; Kumar; Sharma; Kumari, 2014) quanto na manutenção da população ao longo das gerações. Podem ser divididos em quatro tipos: codificação (*encoding*), seleção, mutação e recombinação gênica (*crossover*) (Katoch; Chauhan; Kumar, 2021).

2.4.1.1 Codificação

São os operadores responsáveis por codificar os cromossomos dos indivíduos (neste caso, representar as bases das sequências). A forma com que isso é feito varia de acordo com o escopo do problema (Katoch; Chauhan; Kumar, 2021), sendo que as mais conhecidas são (Kumar, 2013):

- **Binário:** os cromossomos são representados por *bit-strings*.
- **Octal:** a representação é feita através de numerais na base 8.
- **Hexadecimal:** a representação é feita através de numerais na base 16.
- **Permutação:** situação em que o cromossomo representa um elemento de uma sequência. No problema do caixeiro viajante, por exemplo, os cromossomos representam a sequência de cidades que se deve percorrer.
- **Baseado em valor:** Caso onde os cromossomos são representados por uma string de algum valor. Esse valor pode ser inteiro, de ponto flutuante, caracteres, ou até mesmo um objeto. O mais usado no contexto de AMS.
- **Por árvore:** Usado em casos onde o escopo do trabalho é a programação em si. Utiliza-se árvores binárias para representar a ordem de execução de comandos em linguagem de programação.

2.4.1.2 Seleção

A etapa de seleção é onde se define quais indivíduos da população irão compor parte da nova geração e participar da etapa de reprodução a fim de gerar descendentes que substituirão os menos adaptados na próxima geração (Katoch; Chauhan; Kumar, 2021). Os operadores mais conhecidos desse tipo são os de seleção por roleta, ranqueamento, torneio, e elitismo (Katoch; Chauhan; Kumar, 2021):

- **Roleta:** Estratégia onde uma estrutura representando uma roleta é criada, e cada um dos indivíduos da população recebe uma porção dela baseada na sua pontuação (qualidade). Então, "gira-se"a roleta de modo a escolher os indivíduos aleatoriamente. Uma espécie de aleatório com pesos.

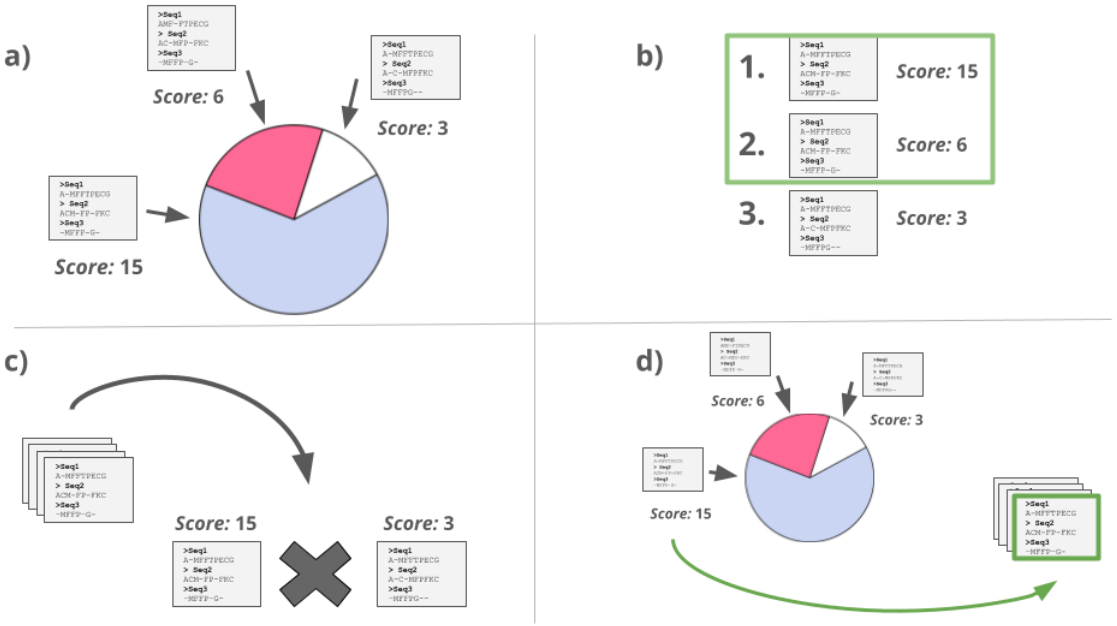
- **Ranqueamento:** Cria-se um *ranking* onde cada indivíduo recebe uma posição considerando sua pontuação. Ou seja, o melhor indivíduo recebe a primeira posição, o segundo melhor a segunda, e assim por diante. Os indivíduos podem ser selecionados aleatoriamente, todos com chances iguais, ou por ordem, sendo os x primeiros indivíduos do *ranking* selecionados.

- **Torneio:** Neste operador também se utiliza de uma espécie de roleta, porém seleciona-se dois indivíduos que serão comparados entre si e o maior seguirá para a próxima geração. Realiza-se essa seleção e comparação até que se tenha escolhido todos os indivíduos que devem seguir para a próxima geração.

- **Elitismo:** Uma versão modificada do operador de roleta. Nesta situação é feita a seleção por roleta, porém caso o melhor indivíduo não tenha sido selecionado ele é automaticamente incluído na próxima geração.

Pode-se verificar na Figura 14 o funcionamento dos quatro operadores.

Figura 14 – Ilustração representando os operadores de seleção citados.
a) Roleta. b) *Ranking* escolhendo os x primeiros elementos. c) Torneio. d) Elitismo.



Fonte: Elaborado pela autora.

2.4.1.3 Mutação

Os operadores de mutação são responsáveis por manter a variabilidade genética da população ao longo das gerações (Katoch; Chauhan; Kumar, 2021), modificando os genes de um indivíduo. Wong *et al.* (2003) afirma que os operadores de mutação são responsáveis pelo ato de exploração no algoritmo, ampliando o espaço de busca.

Existem diversos trabalhos na literatura que propõem diferentes operadores de mutação para diferentes tipos de problemas. Para AMS, estes operadores atuam otimizando o arranjo de *gaps* nas sequências do indivíduo alvo. Essa otimização pode ser através da inserção ou deleção de *gaps*, aumento, diminuição, mistura, separação ou embaralhamento aleatório de blocos de *gaps* já existentes. Todos eles trabalham selecionando aleatoriamente um indivíduo a ser mutado e um *gap* ou bloco de *gaps* das sequências deste indivíduo. (Chowdhury; Garai, 2017).

2.4.1.4 Recombinação gênica

Os operadores de recombinação gênica (ou *crossover*), ao contrário dos operadores de mutação, são responsáveis pela parte de explotação do AG, buscando analisar melhor o espaço de busca no qual o algoritmo se encontra (Wong et al., 2003). Eles atuam gerando novos indivíduos através da combinação entre dois ou mais pais e, assim como nos operadores de mutação, há diversos trabalhos baseados na busca por operadores de recombinação mais eficazes ou aprimoramento dos parâmetros dos atuais. No contexto de AMS, os mais usados são (Katoch; Chauhan; Kumar, 2021; Gondro; Kinghorn, 2007):

- ***K-point crossover:*** Nesse operador, são selecionados k pontos de recombinação vertical aleatórios no comprimento dos alinhamentos pais e os trechos gerados são trocados, de forma a gerar dois novos descendentes. Os valores mais comuns para k são $k = 1$ (*Single-point crossover*) e $k = 2$ (*Two-point crossover*).
- ***Uniform/horizontal crossover:*** Forma de recombinação onde não se troca trechos dos pais, mas sim genes (sequências) completos. Quantas sequências e como elas serão intercaladas fica a critério do desenvolvedor, sendo mais comum trocar uma a uma.

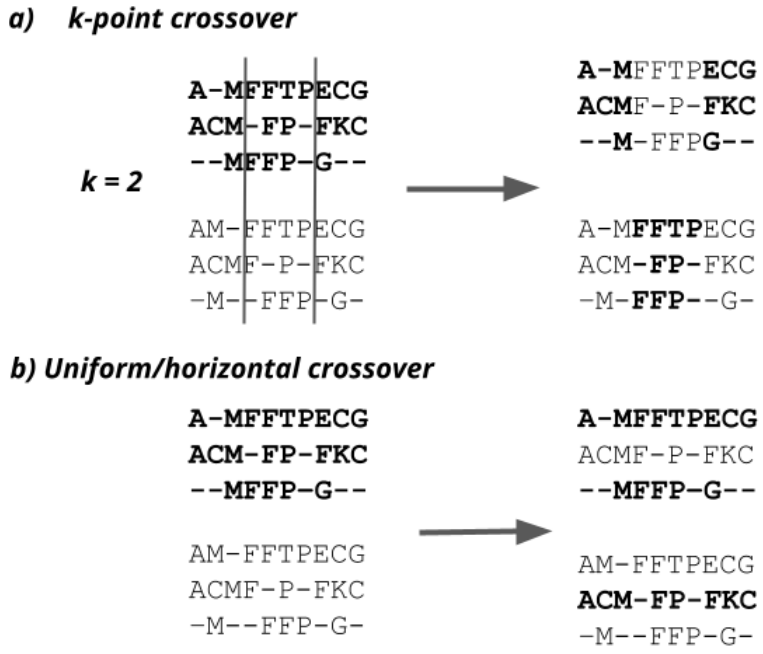
Ambos operadores são mostrados na Figura 15

2.4.2 Função objetivo

Dentro do contexto de AMS, a função objetivo é um método que avalia quantitativamente a qualidade dos alinhamentos produzidos, sendo esse valor chamado de pontuação (Chowdhury; Garai, 2017). Assim, é possível afirmar que o AG, assim como as demais meta-heurísticas para AMS, atuam de forma a otimizar a função objetivo, em busca do seu ponto máximo.

Ao contrário de outros problemas onde o AG pode ser aplicado, em AMS a pontuação atrelada aos indivíduos não considera somente fatores matemáticos e lógicos, mas também

Figura 15 – Ilustração representando os operadores de *crossover*.
a) *K-point crossover*. b) *Uniform/horizontal crossover*



Fonte: Adaptado de Gondro e Kinghorn (2007).

biológicos. Dessa forma, as matrizes de substituição são usadas como base para o cálculo das pontuações, uma vez que apresentam a taxa de chance $M(i, j)$ da base i ser substituída pela base j ou de ser conservada, considerando a ocorrência desse evento na natureza ao longo do tempo. As mais conhecidas são a *Point Accepted Mutation* (PAM) e a *Block Substituition Mutation* (BLOSUM) assim como suas variações, a PAM250 e a BLOSUM62 (Trivedi; Nagarajaram, 2020; Bawono et al., 2017), ilustradas na Figura 16.

Assim, a função objetivo utiliza as taxas encontradas na matriz de substituição usada em conjunto com as características do alinhamento para avaliá-lo. Os métodos mais usados são a *sum of pairs*, a pontuação baseada em consistência (COFFEE) e a pontuação probabilística.

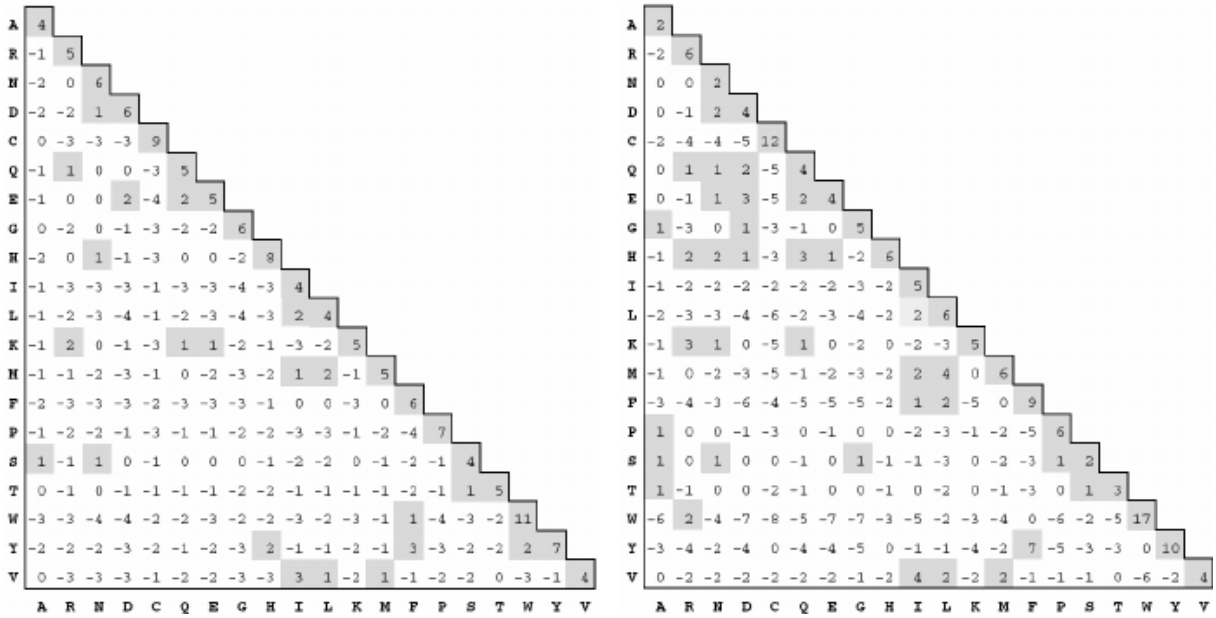
2.4.2.1 Sum of pairs

Sendo talvez a função objetivo mais conhecida, a *sum of pairs* (SOP) trabalha de forma simples: *matches* tem pontuações positivas e *matches* e *gaps* recebem pontuação negativa, e caso haja o uso de matrizes de substituição, elas são levadas em conta (Chowdhury; Garai, 2017).

Considerando k o número de sequências e $p(i, j)$ a pontuação do alinhamento de pares entre as sequências i e j, a SOP é definida na equação 4.

$$score = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} p(i, j) \tag{4}$$

Figura 16 – As matrizes BLOSUM62 e PAM250.



Fonte: Retirado de Orzechowski e Boryczko (2010).

2.4.2.2 Pontuação baseada em consistência (COFFEE)

Proposta em 1998 por Notredame, Holm e Higgins (1998), essa função objetivo leva em conta uma biblioteca contendo todos os $\frac{(n^2-n)}{2}$ alinhamentos de pares das n sequências envolvidas para o cálculo da pontuação de uma candidata à solução. A equação 5 apresenta esse cálculo, considerando $L_{i,j}$ como a pontuação do alinhamento i, j na biblioteca, $len(A_{i,j})$ como o comprimento desse alinhamento e $score(A_{i,j})$ o número de pares de resíduos alinhados em comum entre o alinhamento e a biblioteca. Dessa forma são avaliados todos os alinhamento de pares presentes na possível solução considerando os pesos atribuídos a eles na biblioteca (Amorim et al., 2015).

$$score = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} L_{i,j} * score(A_{i,j})}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} L_{i,j} * len(A_{i,j})} \tag{5}$$

2.4.2.3 Pontuação probabilística

Em constraste às funções anteriormente citadas, uma função objetivo probabilística leva em consideração a probabilidade de uma base ser alinhada com uma outra (Chowdhury; Garai, 2017). A ferramenta MUSCLE (Edgar, 2004), por exemplo, aplica uma função objetivo probabilística baseada na matriz de substituição 240 PAM VTML e na função de média logarítmica (Edgar, 2004; Chowdhury; Garai, 2017).

2.5 APRENDIZADO

Dentro do escopo deste trabalho, o aprendizado corresponde à criação de um modelo computacional que irá aprender sobre o algoritmo principal enquanto este trabalha, e o conhecimento obtido em cada etapa será retornado em forma de auxílio em tomadas de decisão do algoritmo. Desse modo, serão discutidos nessa seção conceitos da área conhecida como aprendizado de máquina.

2.5.1 Aprendizado de máquina

Algoritmos de aprendizado de máquina (do inglês, *machine learning*) são técnicas que buscam aprender padrões de comportamento ou características de um objeto, de modo a tomarem decisões sobre o mesmo sem terem sido explicitamente programadas para tal (Zhang, 2020). Esse aprendizado pode ser classificado em três tipos: Supervisionado, não supervisionado e por reforço (Janiesch; Zschech; Heinrich, 2021).

2.5.1.1 Aprendizado supervisionado

Esse tipo de aprendizado pode ser comparado à atuação de um professor ou mentor, uma vez que o algoritmo recebe dados já rotulados de entrada para que possa identificar a regra ou padrão que define seu rótulo (Zhang, 2020), assim sendo capaz de atribuir um rótulo ou valor a um novo dado. Essa etapa é chamada de treino, e após ela, é realizada uma etapa de teste para verificar o desempenho do método. Havendo uma taxa de acertos dentro do esperado, ele passa a lidar com novos dados não rotulados, tomando as decisões necessárias por si mesmo (Bonaccorso, 2017). Os algoritmos de aprendizado supervisionado são adequados para problemas de classificação (atribuir um valor categórico a uma variável a partir de um conjunto de atributos categóricos ou não) e problemas de regressão (prever um valor numérico para uma variável a partir do valor de outras variáveis) (Janiesch; Zschech; Heinrich, 2021). Alguns dos algoritmos de aprendizado supervisionado são as árvores de decisão, máquinas de vetor de suporte (SVM, do inglês *support vector machine*, redes neurais, regressão linear e regressão logística (Nasteski, 2017; Mahesh, 2020).

2.5.1.2 Aprendizado não supervisionado

Neste caso, não há um guia inicial para o funcionamento do algoritmo, de forma que os dados de treino não possuem rótulo algum. O método irá agrupar os dados de entrada em diferentes conjuntos de acordo com os padrões observados nos mesmos, de modo que dados com características parecidas tendem a serem colocados no mesmo conjunto. Essa abordagem é útil quando se pretende apenas agrupar dados similares sem lhes atribuir um rótulo (problema conhecido como agrupamento ou *clustering*) ou identificar dados anômalos (Bonaccorso, 2017; Janiesch; Zschech; Heinrich, 2021). Dentre os algoritmos de aprendizado não supervisionado

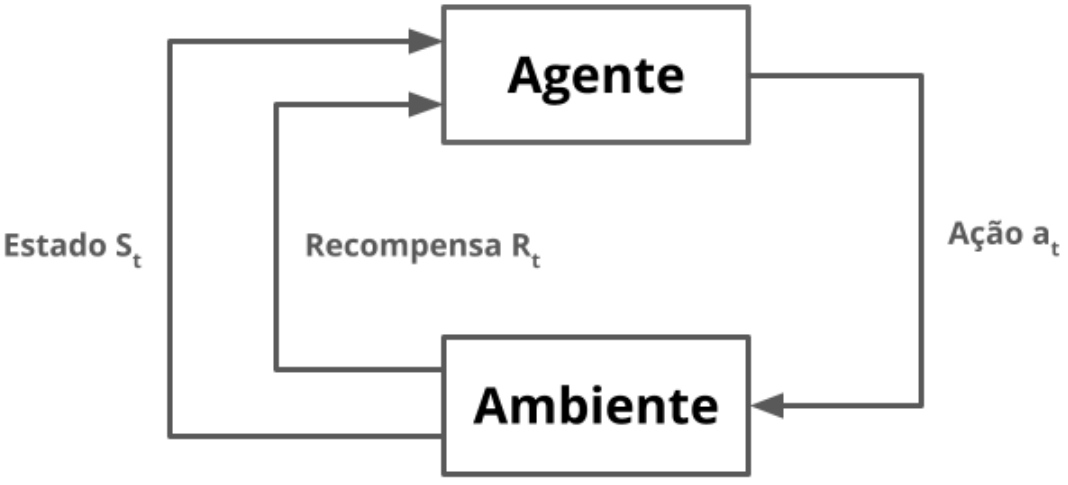
destacam-se o PCA (do inglês *Principal Component Analysis* - análise de componentes principais) e o *K-means* (Mahesh, 2020).

O aprendizado também pode ser realizado utilizando um conjunto de dados onde alguns possuem rótulos e os demais não. Essa abordagem recebe o nome de **aprendizado semi supervisionado**, e é muito utilizada em cenários onde dados sem rótulos são frequentes dentro de uma base de dados rotulada e o custo de rotulá-los é alto (Bonaccorso, 2017; Mahesh, 2020).

2.5.1.3 Aprendizado por reforço

O aprendizado por reforço não conta com um guia, como no supervisionado, porém possui um sistema de *feedback* quantitativo e cumulativo (Mahesh, 2020). Isso significa que o algoritmo não recebe um valor booleano (verdadeiro ou falso, certo ou errado) como retorno, mas sim uma espécie de premiação ou de penalidade, conforme ilustrado na Figura 17. O algoritmo, então, não busca diretamente melhorar sua taxa de acerto, mas sim maximizar as recompensas recebidas conforme interage com o ambiente (Arulkumaran et al., 2017). Com inspiração na psicologia behaviourista, o objetivo do aprendizado por reforço é que o agente aprenda baseado em tentativa e erro, realizando uma ação de acordo com as informações obtidas do estado atual, alterando o mesmo e recebendo uma recompensa positiva ou negativa em retorno proporcional ao impacto gerado por sua ação no ambiente (Arulkumaran et al., 2017).

Figura 17 – Diagrama representando o funcionamento de um modelo de aprendizado por reforço.



Fonte: Adaptado de Mahesh (2020).

Essa abordagem é muito utilizada em ambientes pouco determinísticos, ou seja, ambientes cujas informações mudam a todo instante ou onde não é possível aferir uma taxa de erro precisa (Bonaccorso, 2017), fator que levou à escolha desse tipo de aprendizado para o processo de decisão dos operadores do presente projeto. Alguns dos algoritmos mais conhecidos de aprendizado por reforço são o Sarsa e o Q-Learning (Shakya; Pillai; Chakrabarty, 2023), escolhido para aplicação neste trabalho.

O Q-Learning mantém suas políticas de aprendizado e atuação separadas de modo que, caso o agente venha a tomar uma decisão ruim e o resultado de sua ação seja inadequado, ele entenderá o peso de sua decisão devido ao seus sistema de recompensas e penalidades (Clifton; Laber, 2020) apresentado na equação 6. Nesta equação, $Q_n(s, a)$ é a qualidade Q do estado s depois da ação a, onde α é a taxa de aprendizado do modelo no intervalo $[0, 1]$, R é a recompensa/penalidade recebida pela decisão, γ é o fator de redução da recompensa ao longo do tempo, $Q(s', a')$ é a qualidade da melhor decisão tomada até então, e $Q(s, a)$ é a qualidade atual do estado.

$$Q_n(s, a) = Q(s, a) + \alpha[R + \gamma \ \max Q(s', a') - Q(s, a)] \tag{6}$$

2.6 TRABALHOS CORRELATOS E ESTADO DA ARTE

Em Zafalon *et al.* (2021) os autores buscam amenizar o problema de máximo local do AG por meio da hibridização de heurísticas, utilizando uma abordagem progressiva para realinhar localmente soluções estagnadas. De forma similar, Rúbio-Largo, Vega-Rodriguez e González-Álvarez (2016) aplicam a mesma estratégia para o algoritmo de colônia de abelhas. Ambos algoritmos possuem as mesmas características, assim como a mesma desvantagem.

O método proposto atua após o algoritmo não ter conseguido encontrar uma candidata à solução melhor que a atual após um número específico de iterações. Esse alinhamento tem, então, uma porção de tamanho aleatório retirada que será realinhada pela ferramenta *Kalign*. Após o realinhamento, o novo trecho é reinserido e o novo alinhamento tem sua pontuação avaliada. Os autores de ambos trabalhos constataram que essa abordagem consegue amenizar o problema com sucesso, alcançando resultados ainda melhores. Contudo, Zafalon *et al*. (2021) afirma que o problema de propagação de erros do AP afeta negativamente os resultados do novo método quando se lida com conjuntos de sequências pouco similares. Desse modo, Gomes *et al.* (2022) acrescenta uma abordagem baseada em consistência para amenizar a propagação de erros que pode vir a acontecer no método citado, obtendo melhora nos resultados.

Amorim *et al*. (2018), por sua vez, aplica uma função objetivo baseada em consistência, a COFFEE (Notredame; Holm; Higgins, 1998), no lugar da *Weighted Sum of Pairs* presente na ferramenta MSA-GA (Gondro; Kinghorn, 2007) a fim de alcançar melhores resultados ao alinhar sequências pouco similares entre si. Os autores também modificaram a COFFEE para facilitar seu uso em sequências de nucleotídeos. Os resultados obtidos superaram ferramentas conhecidas, como a Clustal W (Thompson; Higgins; Gibson, 1994).

Mishra, Soam e Tripathi (2021) apresentam uma nova abordagem para sanar as dificuldades do AG baseada em divisão-e-conquista. No trabalho em questão os autores implementam uma etapa de otimização ao final de cada iteração do AG, que consiste em dividir cada indivíduo da população em 4 partes através de cortes verticais, aplicar diferentes operadores de mutação em cada parte, avaliá-las individualmente e recombinar as partes de acordo com as pontuações, visando construir indivíduos melhores. Os resultados obtidos, especialmente quando comparados

aos resultados de ferramentas como Clustal W e Omega (Thompson; Higgins; Gibson, 1994; Sievers; Higgins, 2018), MUSCLE (Edgar, 2004) e PRANK (Löytynoja, 2014), demonstram o sucesso da estratégia.

Em D'Angelo e Palmieri (2021) os autores utilizam dois algoritmos baseados em gradiente descendente para amenizar o problema de máximo local do AG, visando refinar o espaço de busca do AG através da manipulação dos melhores indivíduos, inserindo-os em áreas do espaço de busca mais prováveis de possuirem máximos globais. Os autores aplicaram o método em problemas de otimização numérica e obtiveram resultados positivos, demonstrando a importância de se lidar com o problema de máximo local do AG e como amenizá-lo se traduz em resultados de maior qualidade.

Hussain e Muhammad (2020) trabalham no escopo dos operadores de recombinação gênica, desenvolvendo um novo operador cuja principal vantagem é o equilíbrio entre exploração e explotação no contexto do problema do caixeiro viajante. O novo operador pode facilmente abstraído para o escopo de AMS de modo a contornar os problemas já conhecidos.

Por fim, Jafari, Javid e Rafsanjani (2019) combinam técnicas de aprendizado por reforço e aprendizado profundo (do inglês, *deep learning*) para resolver o problema de AMS. Essa abordagem foi capaz de se igualar a técnicas como a Clustal W (Thompson; Higgins; Gibson, 1994), MAFFT (Katoh et al., 2002) e o ACO em uma abordagem híbrida com o AG (Xiang et al., 2010). Esse trabalho demonstra a eficácia do aprendizado por reforço ao lidar com o ambiente não-determinístico em que o problema de AMS se encontra.

3 METODOLOGIA

Neste capítulo é apresentado o desenvolvimento do método proposto por esse trabalho, elucidando os detalhes da modelagem e dos algoritmos utilizados. Detalhes relativos à implementação em si serão melhor abordados na seção 4.1.

3.1 A GERAÇÃO DA POPULAÇÃO INICIAL

A primeira etapa da abordagem proposta neste trabalho consiste na utilização de quatro meta-heurísticas diferentes para a geração da população inicial. A quantidade de algoritmos foi decidida levando em conta o equilíbrio entre desempenho computacional e exploração adequada do espaço de busca, uma vez que o uso de poucas técnicas não garantiria grande variação entre os indivíduos e o uso excessivo de métodos degradaria desnecessariamente o desempenho computacional da implementação.

As meta-heurísticas são executadas antes do início da execução do AG, de modo paralelo com o auxílio de *threads*, ficando responsáveis cada uma pela geração de 1/4 dos indivíduos da população inicial. Caso o tamanho da população não seja divisível por 4 com resto zero, o excedente dos indivíduos são redistribuídos aos algoritmos seguindo uma ordem pré-estabelecida.

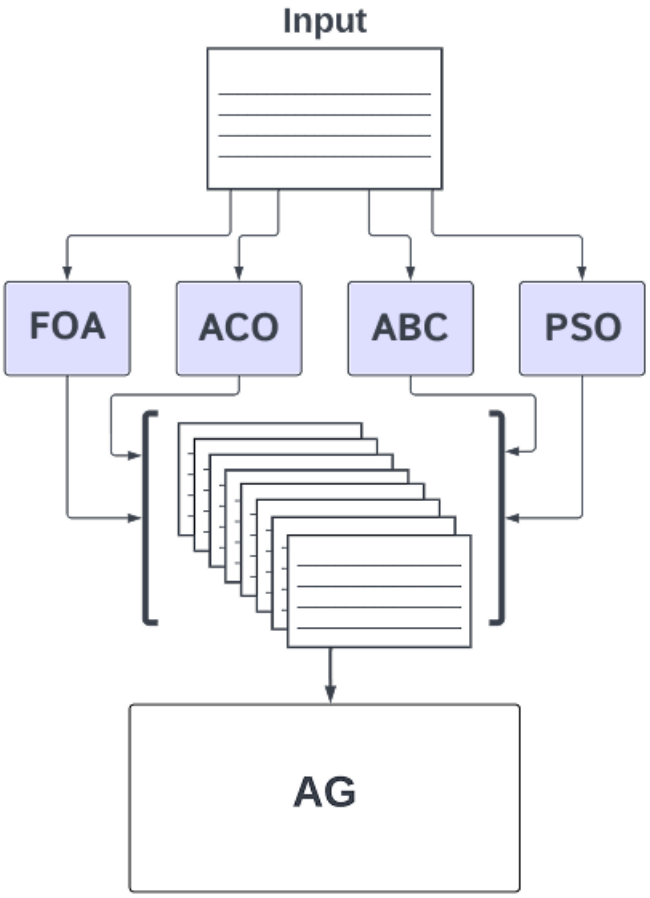
Para esta implementação foram realizados estudos com o objetivo de avaliar diversas meta-heurísticas existentes, havendo modificações para o problema de AMS ou não, por meio dos trabalhos publicados sobre as mesmas e a qualidade dos resultados obtidos. Ao final, foram escolhidos e implementados quatro algoritmos: ACO, ABC, PSO e FOA, cujos serão apresentados e detalhados nas Seções 3.1.1, 3.1.2, 3.1.3 e 3.1.4. Na figura Figura 18 é ilustrada a etapa apresentada.

3.1.1 *Ant Colony Optimization*

O ACO é um algoritmo inspirado no comportamento de busca de alimentos de formigas, que exploram o ambiente em busca de boas fontes de comida e depositam feromônio ao longo do caminho, de modo que caminhos que levam à melhores fontes de comida recebem a visita de mais formigas que, por sua vez, depositam mais feromônio nos mesmos, intensificando a quantidade e reforçando a qualidade do alimento, enquanto caminhos que levam a fontes piores ou pouco interessantes de comida não tem sua trilha de feromônios reforçada, e os mesmos evaporam com o tempo, fazendo com que as formigas parem de escolher aquele trajeto (Amorim, 2017).

Desenvolvido e aprimorado por Dorigo (1992, 1999), o ACO foi pensado inicialmente para problemas como o o caixeiro viajante e o problema quadrático de alocação (Dorigo; Caro; Gambardella, 1999), sendo adaptado posteriormente para o problema de AMS por Moss e

Figura 18 – Ilustração do processo de geração da população inicial.



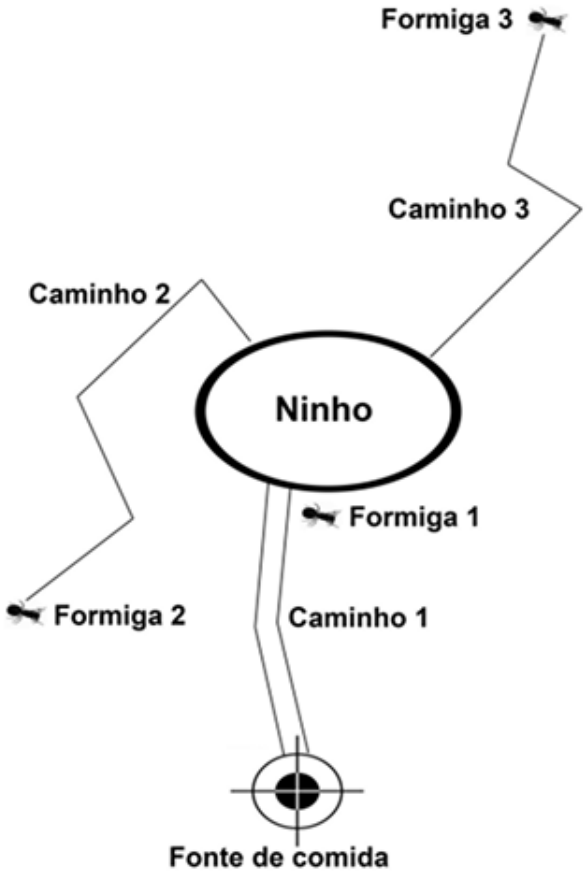
Fonte: Elaborado pela autora.

Johnson (2003) e utilizado em abordagens híbridas em trabalhos como o de Lee *et al.* (2008), Chen *et al.* (2009) e Amorim (2017), tendo eficácia reconhecida para o problema de AMS.

Considerando o problema do caixeiro viajante como base, o ACO trabalha iniciando uma população de formigas que irão percorrer as arestas do grafo que representa o problema depositando certas quantidades de feromônio conforme o fazem. Quais arestas serão percorridas dependem da quantidade de feromônio já depositado na mesma. A tendência é que conforme as formigas percorram melhores caminhos, as arestas que os compõem possuam maior concentração de ferômonio do que as demais, levando a uma maior probabilidade de serem escolhidas por outras formigas e promovendo a convergência do algoritmo para um único caminho: o melhor. Ao final de cada iteração do algoritmo, é aplicada uma taxa de evaporação de feromônio a cada aresta, a fim de garantir que arestas pouco vantajosas sejam ignoradas com o tempo, evitando máximos locais (Dorigo; Birattari; Stutzle, 2007; Lee et al., 2008). Na Figura 19 demonstra-se o funcionamento do ACO.

Para o trabalho atual, a modelagem do ACO utilizada foi baseada no trabalho de Moss e Johnson (2003) e Amorim (2017) e pode ser observada no Algoritmo 1. Primeiramente, o algoritmo gera todos os alinhamentos de pares possíveis entre as sequências de entrada utilizando a ferramenta Kalign (Lassmann, 2020), que servirão como base para compor os

Figura 19 – Busca por alimento em uma colônia de formigas.



Fonte: Retirado de Amorim (2017)

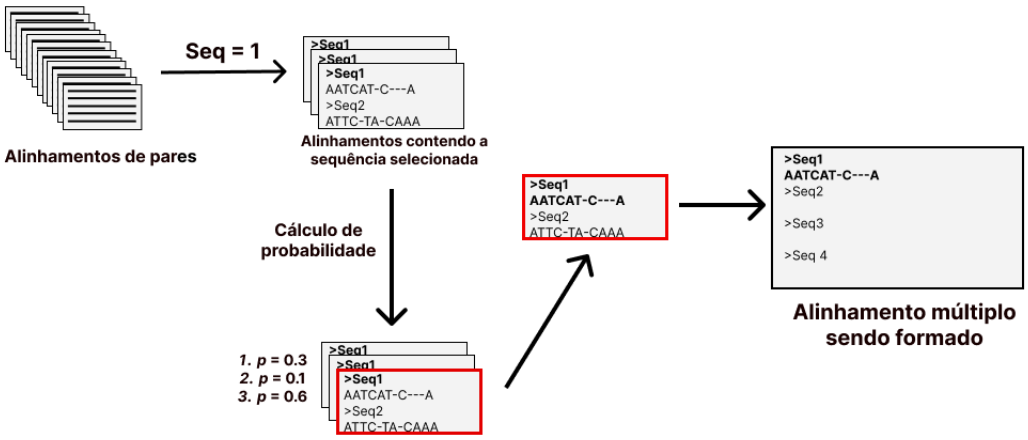
possíveis alinhamentos múltiplos finais, análogo às arestas de um grafo que compõem um caminho no problema do caixeiro viajante. Em seguida, as n formigas são inicializadas e seus caminhos estão inicialmente vazios. Para formá-los, cada sequência de entrada será escolhida com base nos alinhamentos de pares produzidos anteriormente, com o algoritmo selecionando os alinhamentos que contenham a sequência alvo e calculando a probabilidade de cada alinhamento ser escolhido com base na quantidade de feromônio que aquele alinhamento possui em relação ao conjunto. Escolhido o alinhamento, a sequência é então adicionada ao caminho da formiga atual (ver Figura 20) e o processo é repetido até que o caminho da formiga seja equivalente a um alinhamento múltiplo das sequências de entrada. Esse alinhamento terá sua pontuação calculada através da função *Sum of Pairs* e a mesma servirá de base para o cálculo da quantidade de feromônio a ser depositada pela formiga nos alinhamentos de pares utilizados (correspondendo às arestas do caminho). Após todas as formigas terem determinado seus caminhos, ou seja, construído os respectivos alinhamentos múltiplos, é realizado o processo de evaporação do feromônio dos alinhamentos de pares para que uma nova iteração do algoritmo ocorra, de modo que ao longo da execução alinhamentos de pouca qualidade sejam evitados pelas formigas, evitando máximos locais. Ao final da execução, o melhor alinhamento obtido é retornado.

Algoritmo 1 ACO para AMS.

Entrada: seqs, it_{max}, $N_{formigas}$, t_0, α, β, Q, ρ
Saída: Melhor alinhamento múltiplo das sequências de entrada obtido

Gerar todos os $(n^2 - n)/2$ alinhamentos de pares utilizando a ferramenta Kalign
Inicializar todas as $N_{formigas}$ vazias
Para $it \leftarrow 0$ **até** it_{max} **faça**
    Para cada formiga **faça**
        Inicializa $Caminho_{formiga}$ como uma lista vazia, que será o alinhamento formado por ela
        Para $Seq \leftarrow 0$ até Num_{seqs} **faça**
            Seleciona todos os alinhamentos de pares que contém Seq
            Para cada alinhamento a **faça**
                $p_a \leftarrow (f_a)^{\alpha} \cdot (1/(score_a)^{\beta}) / \sum_{n=1}^{total_{aln}} (f_a)^{\alpha} \cdot (1/(score_a)^{\beta})$
            Fim Para
            Escolhe um alinhamento de acordo com as probabilidades calculadas e adiciona a Seq equivalente a $Caminho_{formiga}$
        Fim Para
        Calcula a pontuação do novo alinhamento formado
        Para cada alinhamento de pares a escolhido **faça**
            $f_a \leftarrow (Q \cdot score_{caminho}) \cdot f_a$
            Se $f_a \leq 0$ **Então**
                $f_a \leftarrow t0$
            Fim Se
        Fim Para
    Fim Para
    Para cada alinhamento a de pares **faça**
        $f_a \leftarrow ((1 - \rho) \cdot f_a) \cdot f_a$
    Fim Para
Fim Para

Figura 20 – Ilustração exemplificando o processo de escolha de sequências no ACO.



Fonte: Elaborado pela autora.

3.1.2 *Artificial Bee Colony*

Inspirado no comportamento de forrageamento das abelhas, o ABC é uma metaheurística muito utilizada para problemas de otimização das mais variadas naturezas, e foi desenvolvido por Karaboga (2005) visando inicialmente problemas de otimização multidimensional e multimodal (Karaboga et al., 2005).

Na natureza, as abelhas são divididas em três castas: a abelha-rainha, responsável pela reprodução da população; os zangões, machos haplóides com a única função de reproduzir com a rainha; e as abelhas operárias, que se dividem entre as diversas funções necessárias para a manutençao da colméia, como limpeza, segurança, alimentação e busca por alimento (Karaboga et al., 2005). Esta última função é atribuída às operárias forrageadoras, cujo comportamento inspirou a criação do algoritmo, que se subdividem em três outros grupos: exploradoras, empregadas e observadoras (Karaboga; Akay, 2009).

- **Empregadas:** operárias responsáveis por uma fonte de alimento específica. Tem como função passar todas as informações relativas a essa fonte para as operárias observadoras.
- **Observadoras:** Recebem informações das empregadas para decidir quais fontes de alimento devem ser utilizadas.
- **Exploradoras:** Buscam por fontes de alimentos no ambiente exterior à colmeia.

Assim, a busca por alimento é realizada pelas operárias exploradoras que, ao encontrarem uma fonte de alimento, se tornam empregadas, colhem amostras desta fonte e retornam à colmeia, repassando as informações e amostras às observadoras, que decidem as fontes a serem melhor exploradas (Karaboga; Akay, 2009; Bansal; Sharma; Jadon, 2013).

Dessa forma, o algoritmo trabalha com um conjunto de possíveis soluções representando as fontes de alimento que terão suas informações representadas por uma pontuação obtida através de uma funçao objetivo adequada ao problema (Furlanetto; Gomes; Breve, 2023). Será atribuída, para cada fonte, uma abelha exploradora que se tornará uma abelha empregada, caracterizando a população inicial do algoritmo. Após isso, três passos são executados (Furlanetto; Gomes; Breve, 2023):

- **Passo 1:** Cada possível solução é avaliada pela função objetivo e, de acordo com a pontuação recebida, calcula-se a probabilidade de cada fonte ser selecionada pelas abelhas observadoras, ou seja, ser escolhida para passar pelo processo de exploitação.
- **Passo 2:** As abelhas observadoras selecionam as fontes a serem exploitadas de acordo com as probabilidaddes calculadas, e essas fontes passam por pequenas modificações visando melhorias que ajudem a varrer o espaço de busca das soluções.
- **Passo 3:** Cada fonte de alimento tem sua estagnação verificada. Se alguma das fontes estiver a mais de x iterações sem melhora em sua pontuação, sendo x definido previamente,

a abelha responsável abandona essa fonte, se tornando uma abelha exploradora novamente e escolhendo aleatoriamente uma nova fonte de alimento.

Esses três passos são realizados até que se atinja o critério de parado desejado, podendo ser um número limite de iterações ou a convergência até uma pontuação específica.

Para o problema de AMS tratado neste trabalho, a modelagem do algoritmo foi baseada nos trabalhos de Rúbio-Largo *et al*. (2016) e Furlanetto *et al*. (2023) e é apresentada no Algoritmo 2.

De início, o algoritmo inicializa as $N_{abelhas}$ utilizando alinhamentos produzidos pela ferramenta Kalign. Em seguida, em cada iteração, três etapas são executadas:

1. **Etapa das abelhas empregadas:** Percorre-se a lista de abelhas e, para cada abelha a_i, seleciona-se uma outra abelha aleatória a fim de realizar um processo de *crossover* para gerar uma nova abelha que, caso seja melhor que a_i, a nova abelha substitui a anterior na listas de abelhas. Ao final, calcula-se a probabilidade de cada abelha ser escolhida na etapa seguinte, sendo proporcional a sua pontuação.

2. **Etapa das abelhas observadoras:** Serão selecionadas $N_{abelhas}$ novas abelhas da seguinte forma: uma abelha a_i é escolhida para passar por pequenas alterações (similares a operações de mutação) e, caso haja melhora da pontuação da abelha a_i, a nova abelha é salva. Caso contrário, aumenta-se o contador de estagnação da abelha a_i.

3. **Etapa das abelhas exploradoras:** Nesta etapa, identifica-se as abelhas que atingiram o limite de estagnação e as substitui por novos alinhamentos, tendo como base alinhamentos da ferramenta Kalign que passaram por pequenas modificações.

Algoritmo 2 ABC para AMS.

Entrada: seqs, $N_{abelhas}$, it_{max}, max_estag
Saída: Melhor alinhamento múltiplo das sequências de entrada obtido

Inicializa a lista de abelhas vazia
Gera cada uma das $N_{abelhas}$ abelhas utilizando a ferramenta Kalign e calcula a pontuação inicial das abelhas
Para $it \leftarrow 0$ **até** it_{max} **faça**
`// Etapa das abelhas empregadas:`
    Para cada abelha a_i **faça**
        Seleciona uma abelha aleatória a_x, com $x \neq i$
        Aplica o processo de *one-point crossover* em ambas as abelhas, gerando uma nova abelha $a_{crossover}$
        Se score($a_{crossover}$) $>$ score(a_i) **Então**
            $a_i \leftarrow a_{crossover}$
        Fim Se
    Fim Para
    Calcula a probabilidade de cada abelha ter sua fonte escolhida de acordo com a pontuação de seu alinhamento
`// Etapa das abelhas observadoras:`
    Enquanto $abelhas\_escolhidas < N_{abelhas}$ **faça**
        Escolhe uma abelha a_i aleatória de acordo com a probabilidade calculada
        $a_{nova} \leftarrow mutacao(a_i)$
        Se score(a_{nova}) **>** score(a_i) **Então**
            $a_i \leftarrow a_{nova}$
        Senão
            $a_i.sem\_melhora++$
        Fim Se
        $abelhas\_escolhidas++$
    Fim Enquanto
`// Etapa das abelhas exploradoras:`
    Para cada abelha a_i **faça**
        Se $a_i.sem\_melhora = max\_estag$ **Então**
            Cria um novo indivíduo utilizando um alinhamento base produzido pelo Kalign que passará por pequenas modificações e será salvo no lugar da abelha a_i
            $a_i.sem\_melhora \leftarrow 0$
        Fim Se
    Fim Para
Fim Para

3.1.3 *Particle Swarm Optimization*

O PSO é um dos principais algoritmos inspirados pela natureza (Jain et al., 2022), sendo proposto por Eberhart e Kennedy (1995) inicialmente para a otimização de funções não-lineares e tendo como base o comportamento social de busca por alimento de espécies que vivem em enxames, como pássaros e peixes (Xu; Chen, 2009; Jain et al., 2022). Seu funcionamento se baseia na manutenção de uma população de indivíduos, assim como outros algoritmos evolutivos,

porém nele cada indivíduo é retratado como uma partícula, representando uma possível solução para o problema, com uma posição no espaço e uma velocidade que é ajustada de acordo com o seu desempenho e o de seus vizinhos, que nada mais são do que outras partículas próximas a ela no espaço de busca (Eberhart; Kennedy, 1995; Xu; Chen, 2009). Durante a execução, a posição de cada partícula é atualizada, influenciada pela sua velocidade e pela posição de seus vizinhos – especialmente o melhor deles – o que leva a uma convergência para a melhor posição no espaço (Jain et al., 2022). O cálculo para reajuste da posição de cada partícula varia de acordo com o problema alvo.

No contexto do problema de AMS, o PSO foi utilizado em diversos trabalhos como os de Xu e Chen (2009), Rodriguez, Nino e Alonso (2007) e Chaabane *et al.* (2021), os quais serviram de base para a abordagem utilizada no trabalho atual, que pode ser observada no Algoritmo 3.

Algoritmo 3 PSO para AMS.

Entrada: seqs, $N_{particulas}$, it_{max}
Saída: Melhor alinhamento múltiplo das sequências de entrada obtido

1: Inicializa a lista de partículas vazia
2: Gera cada uma das $N_{partculas}$ partículas utilizando a ferramenta Kalign, gera uma velocidade inicial aleatória, altera a posição de cada *gap* e calcula a pontuação de cada uma delas
3: Atribui a cada partícula duas outras partículas vizinha, sendo as partículas diretamente anterior e posterior à ela na lista de partículas
4: **Para** $it \leftarrow 0$ **até** it_{max} **faça**
5: **Para cada** partícula p_i **faça**
6: Seleciona a melhor partícula entre p_i e suas duas vizinhas
7: **Para cada** seq **em** p_i **faça**
8: **Para cada** gap **em** seq **faça**
9: $velocidade_{gap} \leftarrow c_1\phi(gap_{local\_best} - gap_i) + c_2\phi(gap_{global\_best} - gap_i)$
10: $posicao_{gap} \leftarrow posicao_{gap} + velocidade_{gap}$
11: **Fim Para**
12: **Fim Para**
13: Atualiza a pontuação da partícula
14: **Fim Para**
15: Atualiza a melhor partícula global
16: **Fim Para**

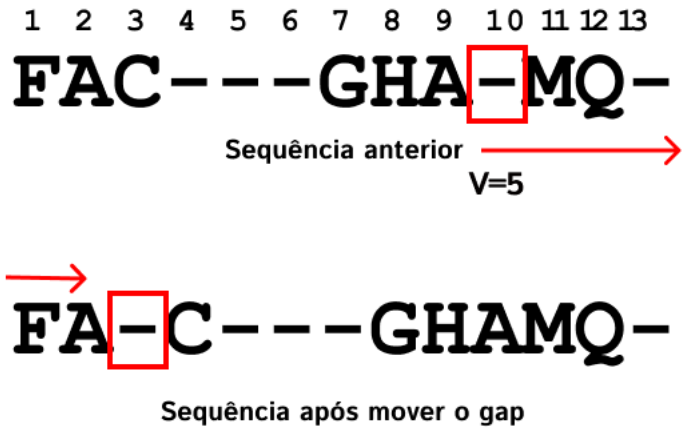
Inicialmente, a lista de partículas é inicializada vazia e será preenchida por alinhamentos produzidos pela ferramenta Kalign, de maneira similar à realizada no Algoritmo 2, porém após gerado o alinhamento inicial, é gerada uma velocidade inicial aleatória v_i e a posição dos *gaps* das sequências de cada partícula são alteradas de acordo com v_i. Em seguida cada partícula tem atribuída a si duas vizinhas, sendo as partículas anterior e posterior à ela na lista de partículas. No caso das partículas das extremidades da lista (primeira e última), que não teriam partículas anterior e posterior a elas, respectivamente, considera-se como se a lista fosse circular e a primeira partícula é vizinha da última, e vice-versa.

Em seguida, durante cada iteração do algoritmo até que se atinja o número máximo de iterações, percorre-se a lista de partículas realizando os seguintes passos para cada partícula p_i:

- Seleciona a melhor partícula local, ou seja, a melhor partícula entre p_i e suas duas vizinhas;
- Para cada *gap* em cada sequência, calcula-se a velocidade do mesmo através da equação apresentada na linha 9 do Algoritmo 3, onde $c1$ e $c2$ são pesos constantes, ϕ é um valor aleatório de distribuição uniforme no intervalo $[0, 1]$, gap_{local_best} é a posição do *gap* equivalente na sequência da melhor partícula local, gap_{global_best} é a posição do *gap* equivalente na sequência da melhor partícula global (de toda a lista) e gap_i é a posição do *gap* da sequência da partícula p_i. Caso a velocidade não seja um valor inteiro, ela é arredondada para o maior inteiro mais próximo;
- Calcula-se a nova posição do *gap* na sequência apenas somando o novo valor da velocidade à posição atual do *gap*. Caso a nova posição ultrapasse o comprimento da sequência, trata-se a sequência de maneira análoga a uma lista circular, somando o valor excedente a partir do início da lista, conforme a Figura 21. Após calculada a nova posição, o *gap* é deslocado para a mesma;
- A pontuação da partícula é atualizada utilizando a função *Sum of Pairs*.

Ao final, a melhor partícula global é atualizada, verificando dentre todas as partículas da lista qual a com melhor pontuação.

Figura 21 – Ilustração exemplificando o deslocamento de *gap* caso a nova posição ultrapasse o comprimento da sequência.



Fonte: Elaborado pela autora.

3.1.4 *Fruit fly Optimization Algorithm*

O FOA é um algoritmo evolutivo inspirado pelo comportamento de forrageamento das moscas-da-fruta, insetos do gênero *Drosophila* que voam ao redor de alimentos, especialmente

frutas em processo de fermentação (Pan, 2012; Mitić et al., 2015). Esses animais são conhecidos pela sua excelente capacidade de busca por alimentos e percepção superior a outros animais, especialmente no que diz respeito à visão e olfato (Pan, 2012). Durante a busca por alimento, as moscas exploram o ambiente utilizando seu apurado olfato para identificar os diferentes cheiros presentes no ar e, ao identificar uma fonte de alimento próxima a si, elas se deslocam para a localização dessa fonte, utilizando a visão como meio para localizar a mesma (Pan, 2012).

O funcionamento do algoritmo se baseia na execução de quatro fases: inicialização, forrageamento pelo olfato, avaliação das novas fontes e forrageamento pela visão, definidas abaixo (Pan et al., 2014; Ranjan; Kumar, 2023).

- **Inicialização:** Inicializa os parâmetros do algoritmo assim como a localização das moscas;
- **Forrageamento pelo olfato:** um conjunto de fontes de alimento são geradas aleatoriamente próximas ao enxame de moscas, representando a exploração do ambiente pelas moscas através do olfato, e cada mosca ficará responsável por explorar cada uma delas.
- **Avaliação das novas fontes:** A concentração de cheiro de cada fonte é avaliada através de uma função objetivo, e a melhor fonte é identificada.
- **Forrageamento pela visão:** As demais moscas então seguirão em direção à melhor fonte, realocando-se em torno dela. Assim, as soluções (localizações) de cada mosca são alteradas baseadas na melhor fonte encontrada, explorando localmente a região em torno da mesma.

Esses passos são executados iterativamente até que se atinja um critério de parada previamente estabelecido, seja ele a convergência até uma pontuação específica ou um número máximo de iterações.

É possível encontrar na literatura a aplicação do FOA em diversos problemas, como otimização numérica (Pan et al., 2014; Mitić et al., 2015), otimização de modelos financeiros (Pan, 2012), identificação de danos estruturais em engenharia civil (Xiong; Lian, 2021), identificação de complexos proteicos (Lei et al., 2016), otimização de parâmetros (Hu et al., 2021a), entre outros. Entretanto, no que diz respeito a AMS, não foram encontrados trabalhos na literatura, sendo a aplicação neste trabalho inédita. Assim, a modificação do FOA para AMS é apresentada no Algoritmo 4, e é baseado no trabalho de Pan (2012).

De maneira similar às demais meta-heurísticas, as N_{moscas} moscas são inicializadas a partir de alinhamentos da ferramenta Kalign submetidos a pequenas perturbações — como a inserção e/ou deleção de *gaps* — efetivando a fase de inicialização. Em seguida, a fase do forrageamento pelo olfato é iniciada, de modo que cada alinhamento, representado pelas moscas, passará por um processo maior de perturbação onde uma quantidade aleatória de *gaps* será deslocada para uma posição igualmente aleatória, a fim de explorar as regiões próximas à solução atual em busca de soluções próximas de melhor qualidade. Após essa fase é realizada a fase de avaliação

Algoritmo 4 FOA para AMS.

Entrada: seqs, N_{moscas} , it_{max}
Saída: Melhor alinhamento múltiplo das sequências de entrada obtido
`// Fase de inicialização`
Inicializa a lista de moscas vazia
Gera cada uma das N_{moscas} moscas utilizando a ferramenta Kalign e calcula a pontuação de cada uma delas e as adicionando à lista de moscas
Para $it \leftarrow 0$ **até** it_{max} **faça**
`// Fase do forrageamento pelo olfato`
Para cada mosca m_i **faça**
Gera uma nova fonte de comida aleatória após aplicar uma perturbação na fonte em que m_i se encontra atualmente
Fim Para
`// Fase da avaliação das fontes`
Calcula a pontuação das novas fontes de alimento através da função *Sum of Pairs*
$melhor\_fonte\_atual \leftarrow max(lista\_moscas)$
`// Fase do forrageamento pela visão`
Se $cheiro(melhor\_fonte\_atual) > cheiro(melhor\_fonte\_global)$ **Então**
$melhor\_fonte\_global \leftarrow melhor\_fonte\_atual$
Fim Se
Para cada mosca m_i **faça**
$m_i \leftarrow melhor\_fonte\_global$
Fim Para
Fim Para

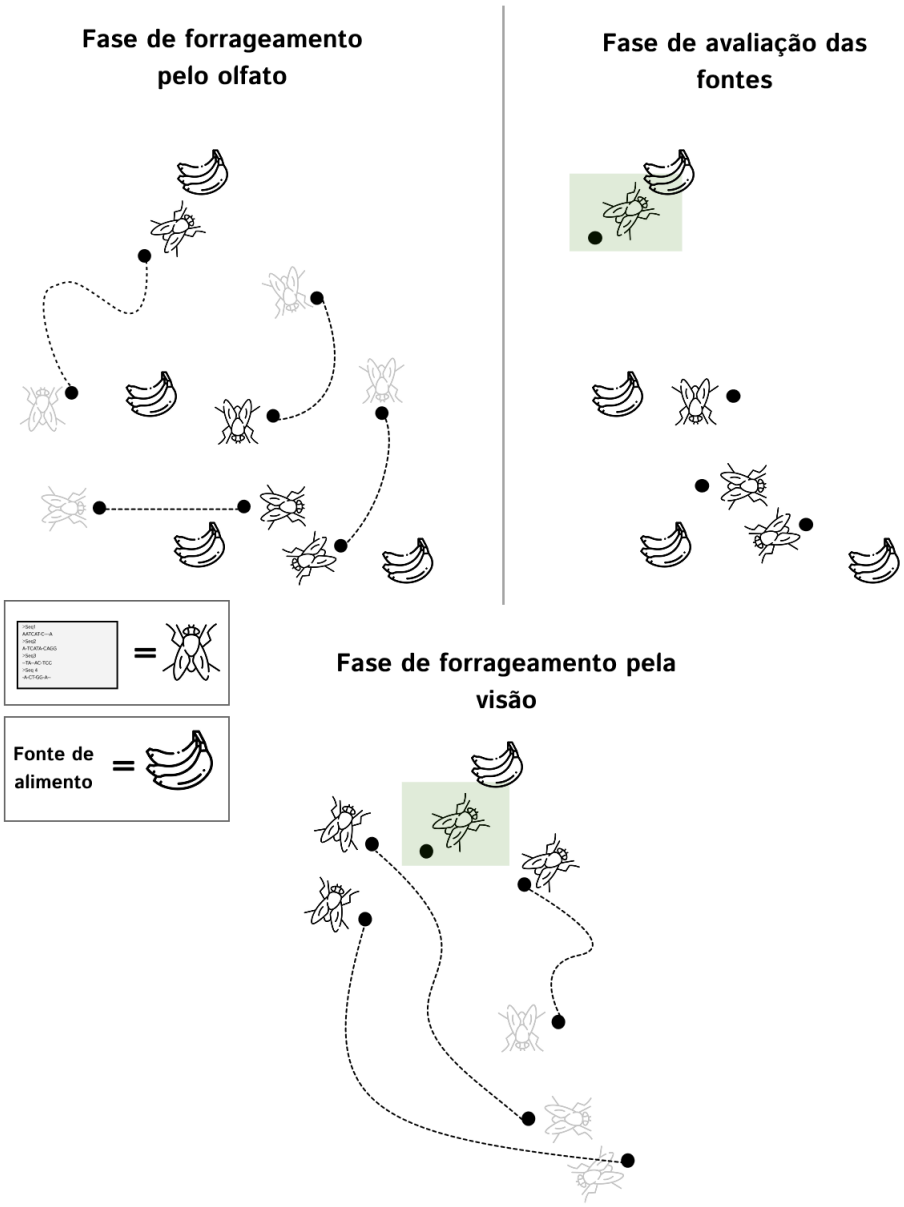
das fontes, aferindo a qualidade dos novos alinhamentos produzidos através da função *Sum of Pairs* e seleciona-se a fonte com maior pontuação como a melhor fonte atual. Por fim, a fase do forrageamento pela visão é executada: caso a melhor fonte atual seja melhor que a melhor fonte global, ela se torna a nova melhor fonte global e, de maneira análoga ao vôo das moscas para uma fonte de alimento melhor do que as que estavam, os alinhamentos cuja pontuação é menor do que a do melhor alinhamento encontrado são substituídos por uma cópia da melhor solução global, que sofrerá futura perturbação na iteração seguinte na tentativa de obter-se melhores respostas com base em uma solução já reconhecidamente boa. Na Figura 22 pode-se verificar esse processo.

Novamente, de maneira semelhante aos demais algoritmos, o processo descrito é repetido a cada nova iteração até que se atinja o número máximo de iterações.

3.2 O ALGORITMO GENÉTICO

O algoritmo genético, como citado na seção 2.4, pode ser definido por três etapas: geração da população inicial, seleção dos melhores indivíduos e criação da nova população. A primeira etapa, neste trabalho, é tratada pela estratégia da seção 3.1, sendo a implementação do AG focada nas duas últimas etapas, que são repetidas a cada iteração. A função objetivo utilizada foi a *Sum of pairs*.

Figura 22 – Representação do comportamento de forrageamento das moscas no algoritmo.



Fonte: Elaborado pela autora.

3.2.1 Seleção dos melhores indivíduos

A seleção dos indivíduos que irão compor a geração seguinte é realizada por meio de um operador de roleta elitista. Nele, os indivíduos recebem uma porção x da roleta proporcional à sua pontuação, sendo x calculado por meio da equação 7.

$$x = \frac{score * 360}{total\_score} \tag{7}$$

Ao final do processo, com todos os indivíduos tendo um pedaço da roleta, $n/2$ indivíduos serão selecionados aleatoriamente para fazerem parte da nova população, sendo a escolha ponderada pelo tamanho da respectiva porção na roleta. Caso o melhor indivíduo da população

não tenha sido selecionado, a característica elitista da roleta entra em ação, e o indivíduo é selecionado automaticamente para compor a próxima população.

3.2.2 Criação da nova população

Selecionados os indivíduos que irão compor a próxima geração, os mesmos participarão de processos de *crossover* e mutação. No primeiro processo, os indivíduos serão escolhidos dois a dois, aleatoriamente, para se reproduzirem e gerarem dois novos descendentes. Esses descendentes terão sua pontuação calculada e serão adicionados à nova população. O processo segue até que se tenha preenchido a nova população, ou seja, até que $n/2$ novos indivíduos tenham sido gerados. Ao final, todos os indivíduos poderão ou não passar por processos de mutação, dependendo da taxa de ocorrência da mesma, definida previamente pelo usuário. Caso o indivíduo que sofreu mutação tenha sua pontuação melhorada, o mesmo é substituído na população pela sua versão mutada. Ao final o melhor indivíduo da população atual é identificado e, caso tenha pontuação maior que o melhor alinhamento encontrado até o momento, ele se torna o novo melhor alinhamento, sendo substituído apenas se uma solução melhor for encontrada.

A seguir são listados os operadores de *crossover* e mutação utilizados neste trabalho e os mesmos são mostrados na Figura 23 e Figura 24, respectivamente:

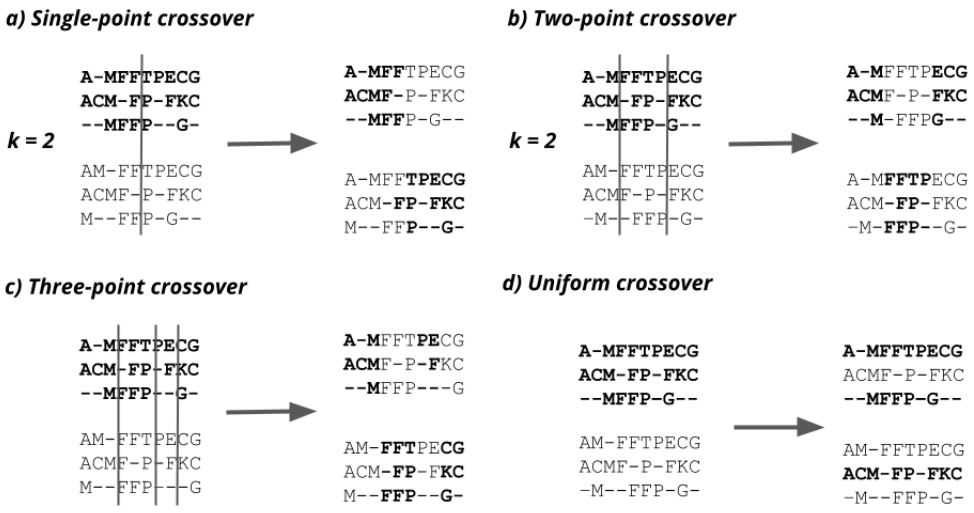
3.2.2.1 Crossover

- *Single-Point Crossover*: Um único ponto de corte é selecionado em ambos os pais, e seus pedaços são intercaladamente trocados. O ponto de corte é decidido aleatoriamente, levando em conta o comprimento das sequências do primeiro pai e o ponto de corte equivalente em cada sequência do segundo pai é definido de modo a garantir que os trechos de ambos os pais possuam as mesmas bases;
- *Two-Point Crossover*: Similar ao anterior, porém utilizando dois pontos de corte.
- *Three-Point Crossover*: Também similar aos outros dois, porém com três pontos de corte.
- *Uniform Crossover*: Ao invés de trocar trechos verticais, os pais trocam sequências inteiras entre si, de modo intercalado.

3.2.2.2 Mutação

- Realinhamento local: uma porção do alinhamento é selecionada para ser realinhada pela ferramenta Kalign. O tamanho da porção é definido aleatoriamente, sendo de 10% a 50% do comprimento total do alinhamento. Após o realinhamento, o trecho é reinserido em sua posição original no alinhamento;

Figura 23 – Operadores de *crossover* utilizados.



Fonte: Elaborado pela autora.

- Deslocamento de bloco de *gaps*: Um bloco de *gaps*, caracterizado pela existência de 2 ou mais *gaps* consecutivos em uma sequência, é identificado em cada sequência do alinhamento e é deslocado, tendo sua posição inicial alterada aleatoriamente. No caso de não existirem blocos para serem movidos, apenas um *gap* aleatório é movido;
- Deslocamento de *gap* único: Um único *gap* de cada sequência do indivíduo tem sua posição alterada aleatoriamente;
- Distribuição de *gaps*: um bloco de *gaps* é selecionado em cada sequência e tem seus *gaps* redistribuídos ao longo do comprimento das sequências, desfazendo o bloco. No caso de não haver blocos disponíveis, todos os *gaps* da sequência são redistribuídos.

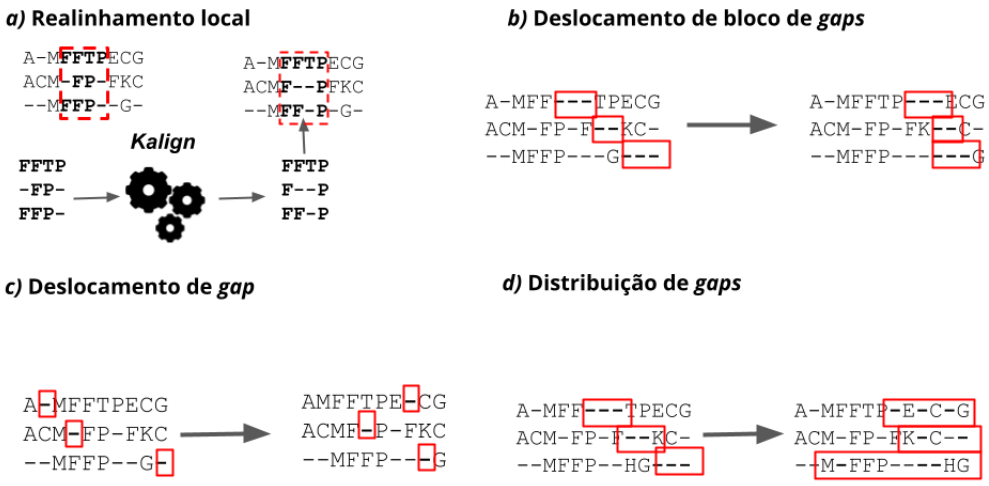
A escolha de qual dos operadores será utilizado, tanto para *crossover* quanto para mutação, é realizada pelo modelo de aprendizado por reforço.

3.2.2.3 Seleção dos operadores: aprendizado por reforço

Para a criação do modelo responsável por selecionar os operadores do AG, foi utilizado o algoritmo Q-Learning, conforme mencionado na seção 2.5.1.3.

Inicialmente, a tabela responsável por guardar a qualidade dos pares ação-estado (s, a) é inicializada com todos os valores iguais a zero, o que faz com que na primeira iteração do AG as escolhas de operadores sejam de fato aleatórias, uma vez que todos possuem o mesmo peso (a qualidade). Conforme as iterações são executadas, os valores da tabela vão sendo alterados, favorecendo a escolha de operadores que se mostrem mais efetivos para a execução atual e desestimulando a escolha de operadores que não sejam tão efetivos.

Figura 24 – Operadores de mutação utilizados.



Fonte: Elaborado pela autora.

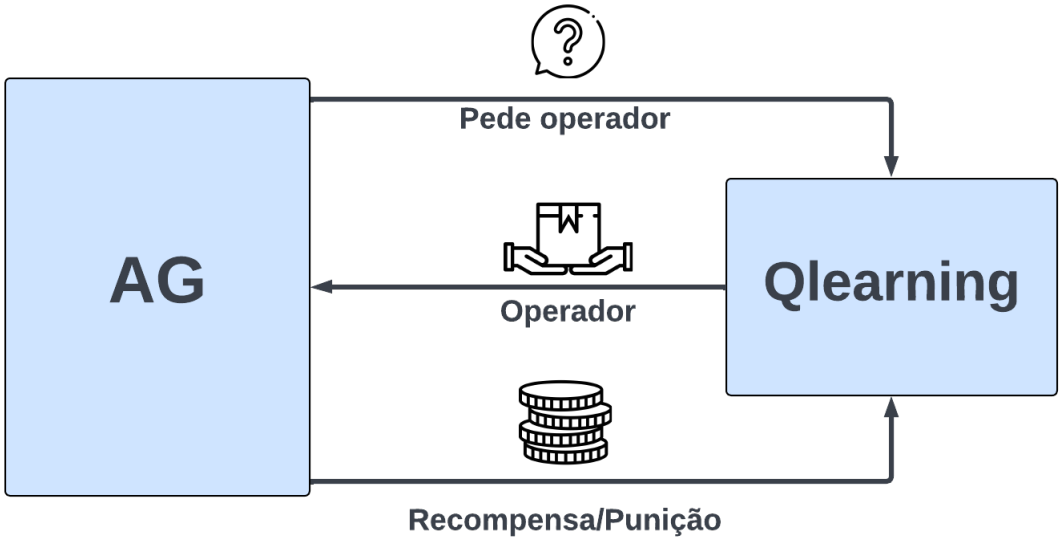
As recompensas e/ou penalidades necessárias para recalcular a qualidade dos pares (s, a) (segundo a Equação 6) são calculadas a partir da diferença de pontuação entre o alinhamento antes e após a mutação, sendo positiva quando a pontuação aumenta, caracterizando uma recompensa, ou negativa quando a pontuação diminui, caracterizando uma penalidade. No caso do *crossover*, é utilizada a média das diferenças entre pais e filhos, conforme a Equação 8.

$$reward = \frac{(c_1 - p_1) + (c_1 - p_2) + (c_2 - p_1) + (c_2 - p_2)}{4} \tag{8}$$

Ao longo da execução do AG, no momento de realizar as operações para criação de novos indivíduos, o agente é acionado e informado sobre qual o estado atual, isto é, o tipo de operador desejado (*crossover* ou mutação) e sua resposta é o número equivalente a um dos operadores da operação desejada. Ao final da ação, ele recebe sua recompensa e reajusta sua tabela de qualidade, conforme mostrado na Figura 25.

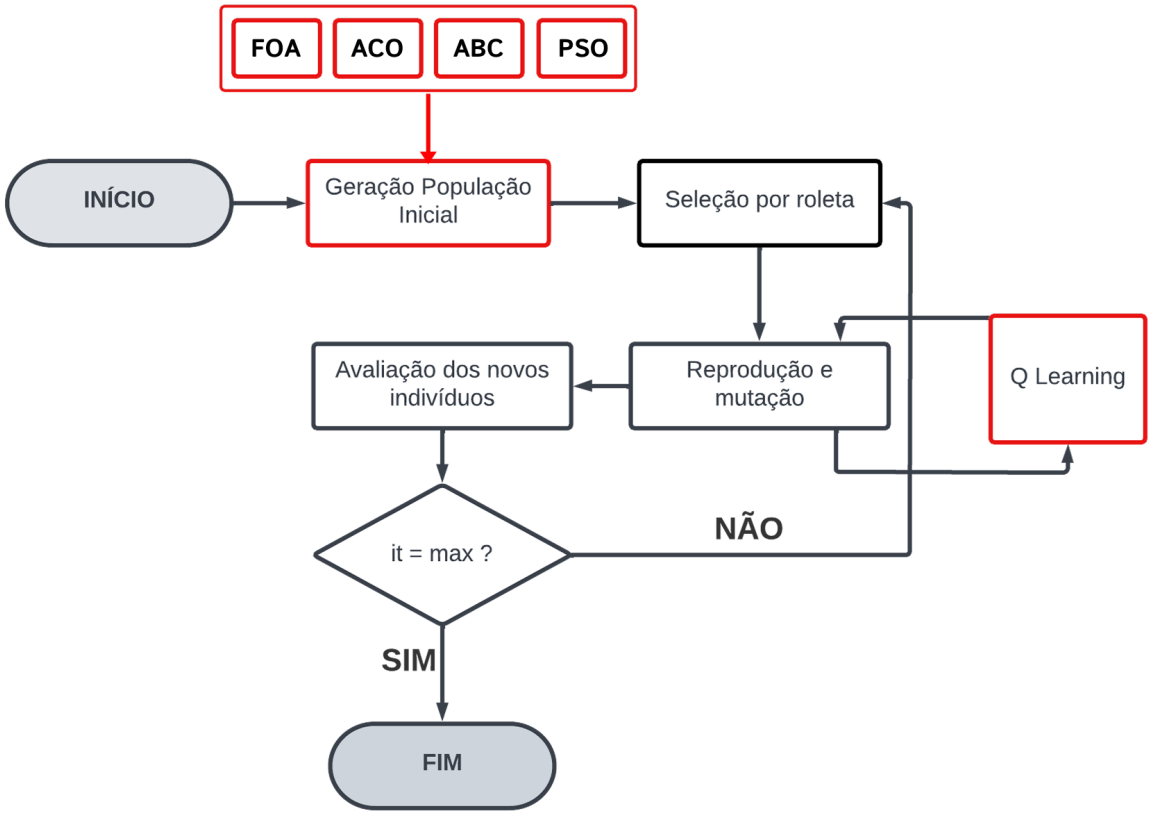
Os processos descritos se repetem até que o critério de parada seja atingido e o melhor alinhamento obtido seja retornado. Neste caso, optou-se pelo critério de número de iterações, sendo este definido previamente à execução pelo usuário. Na Figura 26 apresenta-se o fluxograma do funcionamento geral da estratégia proposta.

Figura 25 – Ilustração da comunicação entre o AG e o Q-learning para seleção dos operadores.



Fonte: Elaborado pela autora.

Figura 26 – Fluxograma da execução do AG. Os quadros em vermelho representam as estratégias propostas nesse trabalho.



Fonte: Elaborado pela autora.

4 IMPLEMENTAÇÃO, EXPERIMENTOS E RESULTADOS OBTIDOS

Nesta seção são apresentados os detalhes relativos à implementaçao do método proposto, os cenários de teste e experimentos conduzidos, além dos resultados obtidos e a discussão acerca dos mesmos.

4.1 IMPLEMENTAÇÃO DA ESTRATÉGIA

A estratégia proposta na seção 3 foi implementada utilizando a linguagem de programação Python na versão 3.10, escolhida pela sua praticidade, quantidade de bibliotecas disponíveis e pelo foco do trabalho ser a eficácia do método proposto e não seu desempenho computacional. O código da ferramenta relativa ao método proposto foi dividido em bibliotecas customizadas contendo estruturas de dados e funções específicas para cada algoritmo trabalhar e/ou reunindo os métodos comuns à maioria dos algoritmos. São elas:

- **ABC**: Reúne as funções responsáveis pela implementação do algoritmo ABC como o próprio algoritmo e funções auxiliares como o cálculo de probabilidade de escolha das abelhas e o *crossover* aplicado às abelhas como forma de modificação das mesmas;
- **ACO**: Análoga à ABC, porém relativa ao ACO. Inclui uma classe *Ant* que agrupa os principais atributos que caracterizam as formigas no contexto de AMS: o alinhamento gerado, o caminho que a formiga percorreu e sua pontuação;
- **FOA**: Contém apenas uma função: a que implementa o algoritmo, sem a necessidade de métodos adicionais além dos utilizados igualmente por todos os algoritmos;
- **PSO**: Contém uma classe *Particle* e a função que implementa o algoritmo. Nesta classe estão contidos os atributos relativos ao alinhamento, aos vizinhos da partícula, a melhor pontuação global e a melhor partícula entre a mesma e suas vizinhas; e os métodos relativos ao cálculo de velocidade e posição da partícula;
- **QL**: Responsável por abrigar as funções e tabela interna de estado/ação que caracterizam o algoritmo Q-Learning, permitindo que o mesmo se comunique com o restante da ferramenta.
- ***Gap***: Contém as funções relativas à modificação de *gaps* em alinhamentos que são de uso comum entre os quatro algoritmos de criação da população inicial e o próprio AG como: criação de geração inicial dos quatro algoritmos, verificação da existência de colunas de *gaps* e do comprimento das sequências (caso alguma modificação acabe por desigualar o comprimento das sequências), inserção e deleção de *gaps* individuais ou blocos e identificação das posições das sequências que contém *gaps*;

- **Criação de população**: Reúne os métodos para a criação da população inicial do AG, calculando a quantidade de *threads* necessárias, qual das quatro meta-heurísticas será executada em determinada *thread* e invoca os algoritmos e reúne os resultados obtidos, formando a população

- **AG**: reúne todos os métodos para a implementação da versão modificada do AG proposta neste trabalho, como os operadores de *crossover* e mutação utilizados, a seleção por roleta e o AG em si, utilizando a população inicial gerada e a seleção de operadores através do modelo do algoritmo Q-Learning;

A metodologia é coordenada através da função *main*, presente em um arquivo à parte das bibliotecas. A *main* é responsável por receber os argumentos de execução do AG – como nome do arquivo de entrada e saída e os hiperparâmetros do AG, como tamanho da população, número de gerações e taxas de *crossover* e mutação –, abrir o arquivo de entrada, guardar as sequências em uma lista, invocar a função de criação da população inicial, receber seu resultado, invocar o AG enviando seus hiperparâmetros e população inicial e recebendo, ao final de sua execução, seu resultado, escrevendo-o em um arquivo FASTA.

4.2 EXPERIMENTOS E FORMA DE ANÁLISE DOS RESULTADOS

Os experimentos realizados para avaliar a eficácia da estratégia proposta utilizaram os casos de teste disponíveis na base de dados BAliBase 3.0 (Thompson et al., 2005). Esta base conta com diferentes casos de teste com quantidade e comprimento de sequências de aminoácidos variados, separados em famílias de acordo com as características das sequências nos casos: A Referência 1 (famílias 11 e 12) contém sequências equidistantes entre si em termos de homologia, isto é, semelhança entre as sequências; A Referência 2 (família 20) contém conjuntos contendo sequências órfãs — que são muito diferentes das demais; A Referência 3 (família 30) contém subfamílias diferentes entre si; A Referência 4 (família 40) engloba casos com sequências de comprimento significativo; e a Referência 5 (família 50) são sequências que possuem grandes inserções, ou seja, possuem uma diferença de comprimento notavelmente diferente das demais, dificultando a comparação direta.

Devido à natureza estocástica do problema e, consequentemente, dos algoritmos, cada caso de teste da BAliBase foi executado cinco vezes a fim de garantir resultados estatisticamente confiáveis e foram avaliados utilizando a ferramenta qscore[1], que compara o alinhamento obtido com um alinhamento de referência, realizados manualmente, tidos como corretos e fornecidos também pela BAliBase. O qscore retorna duas métricas relativas à quantidade de bases corretamente alinhadas e a quantidade de colunas alinhadas corretamente, chamadas de Q (*quality score*) e TC (*Total Columns score*) respectivamente. Ambos valores variam no intervalo $[0, 1]$, onde 0 é o pior alinhamento e 1 é o melhor, de modo que quanto maior os valores de Q e

[1] https://www.drive5.com/qscore/

TC, maior a qualidade do alinhamento produzido. Destaca-se que tanto a base de dados quanto a ferramenta estão disponibilizados gratuitamente na internet.

Os resultados obtidos foram comparados com diversas ferramentas da área, separadas em três conjuntos:

- **Ferramentas populares**: Clustal Omega (Sievers; Higgins, 2018), Kalign (Lassmann, 2020), MAFFT (Katoh et al., 2002) e MUSCLE (Edgar, 2004), ferramentas populares devido ao baixo tempo de execução por serem, em sua maioria, progressivas;
- **Meta-heurísticas**: ABC (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016), ACO (Moss; Johnson, 2003), PSO (Xu; Chen, 2009) e FOA, este último implementado para AMS pela primeira vez neste trabalho. São os mesmos algoritmos utilizados para gerar a população inicial do AG a fim de avaliar o quanto o AG foi capaz de melhorar a qualidade dos alinhamentos;
- **Algoritmo genético**: Este conjunto é composto apenas da ferramenta MSA-GA (Gondro; Kinghorn, 2007) que serve como AG base para comparar a evolução da estratégia proposta em relação ao AG sem as modificações implementadas.

A análise dos resultados obtidos e a comparação com cada grupo de ferramentas serão apresentados na Seção 4.3.

Os hiperparâmetros utilizados no AG e nas meta-heurísticas responsáveis pela população inicial foram escolhidos através do *framework* de otimização de hiperparâmetros Optuna (Akiba et al., 2019) e a função objetivo *sum of pairs* foi aplicada através da biblioteca PyMSA [2] – que fornece funções para avaliação de alinhamentos múltiplos – e a matriz de substituição escolhida foi a PAM250.

Os parâmetros de cada algoritmo, incluindo o AG, podem ser vistos nas tabelas 2, 3, 4, 5, 6 e 7.

Parâmetro	**Valor**
Tamanho da população	28
Número de gerações	491
Taxa de *crossover*	0.926
Taxa de mutação	0.089

Tabela 2 – Parâmetros do algoritmo genético proposto.

No Q-Learning, os parâmetros ϵ e decaimento ϵ são responsáveis por incentivar a exploração das ações por parte do modelo. A cada nova interação, ϵ diminui, fazendo que conforme o modelo aprende, ele desista de explorar novas possibilidades e tenda para as ações com resultados já conhecidos. Essa política é conhecida como ϵ-*greedy* (Dann et al., 2022).

[2] https://github.com/benhid/pyMSA

Parâmetro	Valor
Número de abelhas	100
Limite máximo de iterações	345
Limite máximo de estagnação	50

Tabela 3 – Parâmetros do ABC.

Parâmetro	Valor
Número de formigas	100
Limite máximo de iterações	128
$t0$	6.79×10^{-4}
α	0.957
β	0.522
Q	0.221
ρ	0.299

Tabela 4 – Parâmetros do ACO.

Parâmetro	Valor
Número de moscas	100
Limite máximo de iterações	1779

Tabela 5 – Parâmetros do FOA.

Parâmetro	Valor
Número de partículas	100
Limite máximo de iterações	1266

Tabela 6 – Parâmetros do PSO.

Parâmetro	Valor
Número de estados	2
Número de ações	4
α	0.5
γ	0.9
ϵ	0.5
Decaimento ϵ	0.9

Tabela 7 – Parâmetros do Q-Learning.

4.3 RESULTADOS

4.3.1 Primeiro conjunto: Ferramentas populares

Os primeiros testes consistiram na execução dos casos de teste da BAliBase pelas ferramentas do primeiro conjunto com o objetivo de comparar o desempenho da abordagem proposta com os métodos de referência da área. Foram utilizados os parâmetros padrão de cada ferramenta que podem ser consultados no servidor do EMBL-EBI[3] que abriga e disponibiliza as mesmas gratuitamente. Os resultados são mostrados na Tabela 8, onde as colunas Q e TC equivalem aos valores obtidos pela ferramenta qscore, separados por família.

Tabela 8 – Média dos resultados obtidos para cada ferramenta nas famílias do BAliBase.

Método	**Famílias**											
	11		**12**		**20**		**30**		**40**		**50**	
	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC
Tese	0.5627	0.3308	0.9046	0.7962	0.8207	0.3213	0.6442	0.2148	0.7674	0.4606	0.7339	0.3985
Clustal Ω	0.5900	0.3621	0.9087	0.8037	0.7773	0.3906	0.8234	0.5316	0.9300	0.7760	0.8515	0.5245
Kalign	0.6052	0.3687	0.9186	0.8107	0.8800	0.3810	0.7992	0.4316	0.9146	0.7227	0.8219	0.4524
MAFFT	0.6129	0.3538	0.9274	0.8168	0.7893	0.3890	0.8222	0.4592	0.9231	0.7348	0.8720	0.5519
MUSCLE	0.6891	0.4481	0.9471	0.8727	0.7910	0.3983	0.8382	0.5150	0.9302	0.7495	0.9017	0.6213

Como é possível observar, os resultados alcançados pela estratégia proposta não superam os resultados das demais ferramentas em todos os casos, porém os valores são muito próximos, tanto para Q quando para TC. A fim de avaliar a significância dessa pequena diferença, o teste de hipótese não-paramétrico Kruskal-Wallis (Kruskal; Wallis, 1952) foi aplicado. O teste em questão avalia as diferenças entre três ou mais grupos de amostras em situações onde a normalidade dos dados não é garantida (McKight; Najab, 2010), o que ocorre nos dados utilizados (Rubio-Largo; Vega-Rodríguez; González-Álvarez, 2016). O valor de confiança utilizado foi de 5% (*p-value* < 0.05). O valor obtido para os valores de Q e TC, respectivamente, foram $p = 0.0149$ e $p = 0.0396$, o que indica que algum dos grupos tem uma diferença relevante do ponto de vista estatístico. Ao verificarmos, no entanto, ao aplicar o teste de Dunn (Dunn, 1961) — teste *post-hoc* aplicado para determinar quais grupos se destacam em relação aos demais — foi possível notar que a única ferramenta que se destaca estatisticamente em relação à estratégia foi a MUSCLE em ambas as pontuações sendo, em média, 14.5% e 28.8% superior ao presente trabalho nas pontuações Q e TC, respectivamente. É importante ressaltar que a ferramenta MUSCLE possui uma estratégia híbrida entre AP e métodos iterativos (Edgar, 2004), o que potencializa seus resultados em relação a outras técnicas.

As Figuras 27 e 28 apresentam os *boxplots* relativos às pontuações Q e TC de cada ferramenta, respectivamente. Nelas é possível observar a proximidade das medianas, o intervalo de valores e *outliers* observados nos resultados de cada método. É possível notar que o AG híbrido obteve

[3] https://www.ebi.ac.uk/jdispatcher/msa

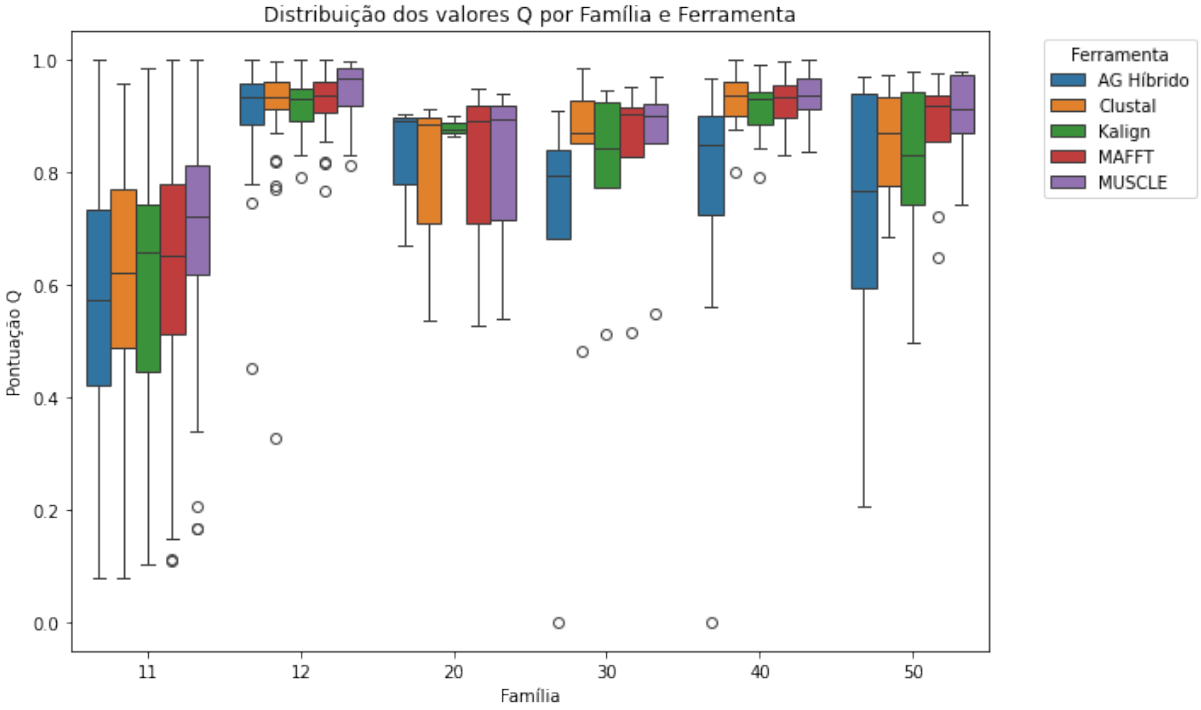


Figura 27 – *Boxplot* que demonstra a distribuição dos valores Q obtidos.

um maior intervalo de valores nas famílias 40 e 50, indicando uma possível dificuldade com cenários dentro dessas características.

O BAliBase contém também dados sobre a porcentagem de similaridade entre as sequências de cada caso, sendo este valor conhecido como homologia. Para avaliar os resultados obtidos de acordo com essas taxas, os casos de teste foram divididos em três intervalos: <20%, >20% e <50%, e >50%. A média dos resultados para cada ferramenta dentro desses intervalos foi calculada e é apresentada na Tabela 9. Os valores demonstram um desempenho inferior por parte do AG híbrido no que diz respeito a sequências mais semelhantes entre si. Esse fator pode ser um indicativo de divergência do algoritmo nessas situações, hipótese que será melhor investigada nas Seções 4.3.2 e 4.3.3.

Método	**<20%**	**>20% e <50%**	**>50%**
AG Híbrido	0.2494	0.7660	0.6344
Clustal Omega	0.2787	0.8049	0.9542
Kalign	0.3290	0.8084	0.9470
MAFFT	0.3181	0.8191	0.9706
MUSCLE	0.4076	0.8558	0.9910

Tabela 9 – Média dos resultados dos métodos de acordo com os intervalos definidos.

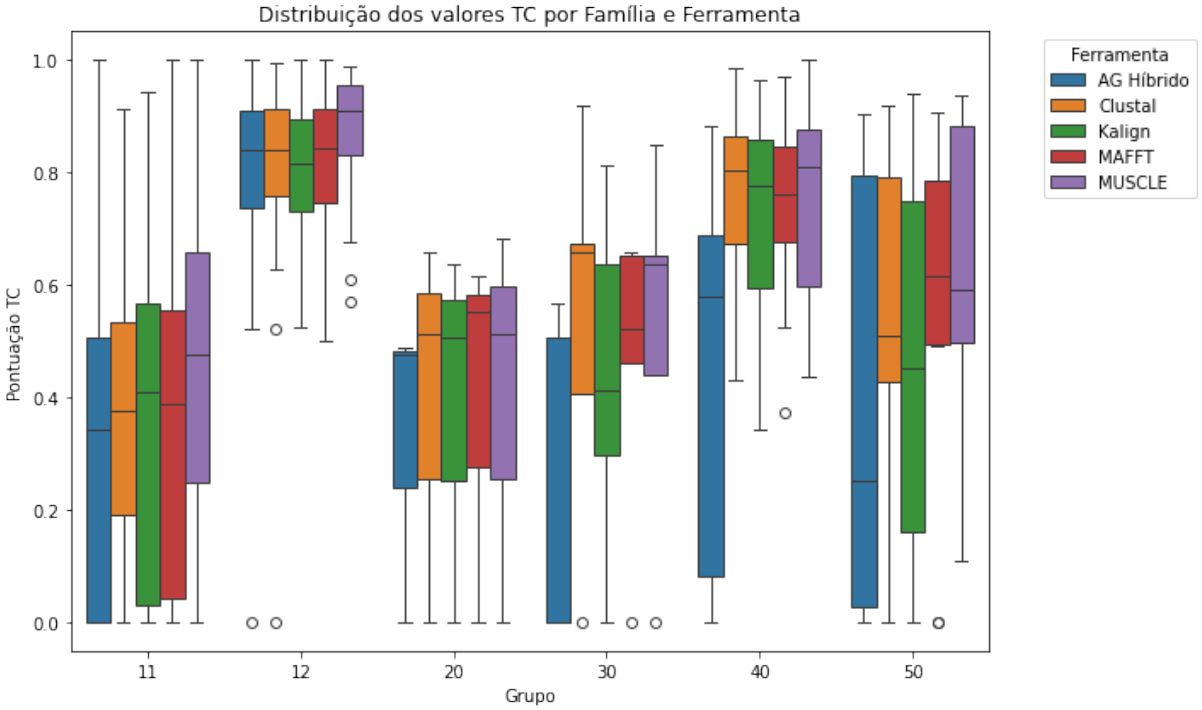


Figura 28 – *Boxplot* que demonstra a distribuição dos dos valores TC obtidos.

4.3.2 Segundo conjunto: Meta-heurísticas

Neste conjunto de testes o foco foi demonstrar como o método proposto é capaz de evoluir os alinhamentos que compõem a população inicial a partir da comparação direta dos resultados obtidos pelas quatro meta-heurísticas utilizadas para tal e do AG híbrido. Os parâmetros dos algoritmos foram os mesmos mostrados nas tabelas 2, 3, 4, 5 e 6, e as médias obtidas para Q e TC, por família, são apresentadas na Tabela 10

Tabela 10 – Média dos resultados obtidos para cada meta-heurística nas famílias do BAliBase.

Método	**Famílias**											
	11		**12**		**20**		**30**		**40**		**50**	
	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC
Tese	0.5627	0.3308	0.9046	0.7962	0.8207	0.3213	0.6442	0.2148	0.7674	0.4606	0.7339	0.3985
ABC	0.1148	0.0250	0.5560	0.4445	0.4869	0.3520	0.2489	0.1647	0.0292	0.0000	0.3305	0.2571
ACO	0.3360	0.0398	0.4830	0.1790	0.2080	0.0000	0.0860	0.0000	0.0806	0.0000	0.1320	0.0000
FOA	0.4390	0.0927	0.7260	0.5420	0.1990	0.0000	0.1010	0.0000	0.0730	0.0000	0.0749	0.0000
PSO	0.3770	0.1320	0.4280	0.1590	0.1421	0.0000	0.1710	0.0000	0.2280	0.0000	0.2110	0.0000

É possível observar, inicialmente, que o AG híbrido superou os resultados das demais meta-heurísticas em todas as famílias. Seguindo as análises que foram conduzidas no primeiro conjunto, o teste de Kruskal-Wallis foi aplicado obteve-se $p = 1.1032 \times 10^{-34}$ para a pontuação Q e $p = 2.5896 \times 10^{-40}$ para a pontuação TC. Utilizando o teste *post-hoc*, comprovou-se de que a estratégia do presente trabalho é estatisticamente superior às demais meta-heurísticas utilizadas, com pontuações Q e TC em média 186.26% e 554.96% superiores, respectivamente.

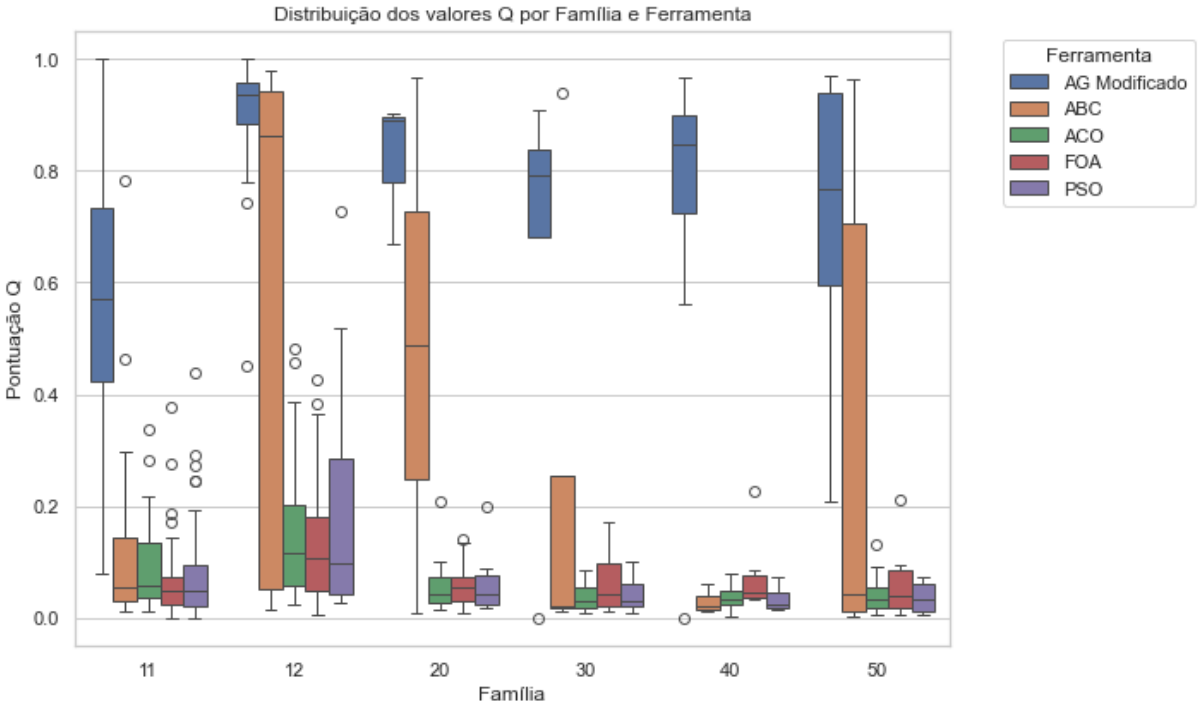


Figura 29 – *Boxplot* ilustrando a distribuição dos valores Q obtidos por todos os algoritmos.

Isso demonstra a capacidade do método implementado de melhorar pré-alinhamentos fornecidos no ínicio da execução como descrito inicialmente por Gondro e Kinghorn (2007).

As Figuras 29 e 30 apresentam *boxplots* que ilustram a distribuição dos resultados. Nota-se que os resultados das meta-heurísticas possuem certa constância, ainda que apresentem baixos valores, especialmente na pontuação TC. Essa constância também é observada no AG híbrido, havendo uma maior taxa de dispersão nas famílias 11 e 50, tendo seus valores mínimos distantes da mediana.

Assim como foi feito na Seção 4.3.1, os resultados foram classificados pela homologia de suas sequências com os mesmos intervalos anteriormente usados. Os resultados são apresentados na Tabela 11. Através desses valores é possível notar que as outras meta-heurísticas também seguem a tendencia comumente vista em ferramentas de AMS: conforme a semelhança entre as sequências aumenta, a qualidade do alinhamento também cresce, reforçando a hipótese de que o AG híbrido pode estar divergindo nessas situações.

Método	**<20%**	**>20% e <50%**	**>50%**
AG Híbrido	0.2494	0.7660	0.6344
ABC	0.0625	0.2965	0.5912
ACO	0.0640	0.1079	0.1145
FOA	0.0368	0.1155	0.1822
PSO	0.0284	0.0974	0.1084

Tabela 11 – Média dos resultados das meta-heurísticas de acordo com os intervalos definidos.

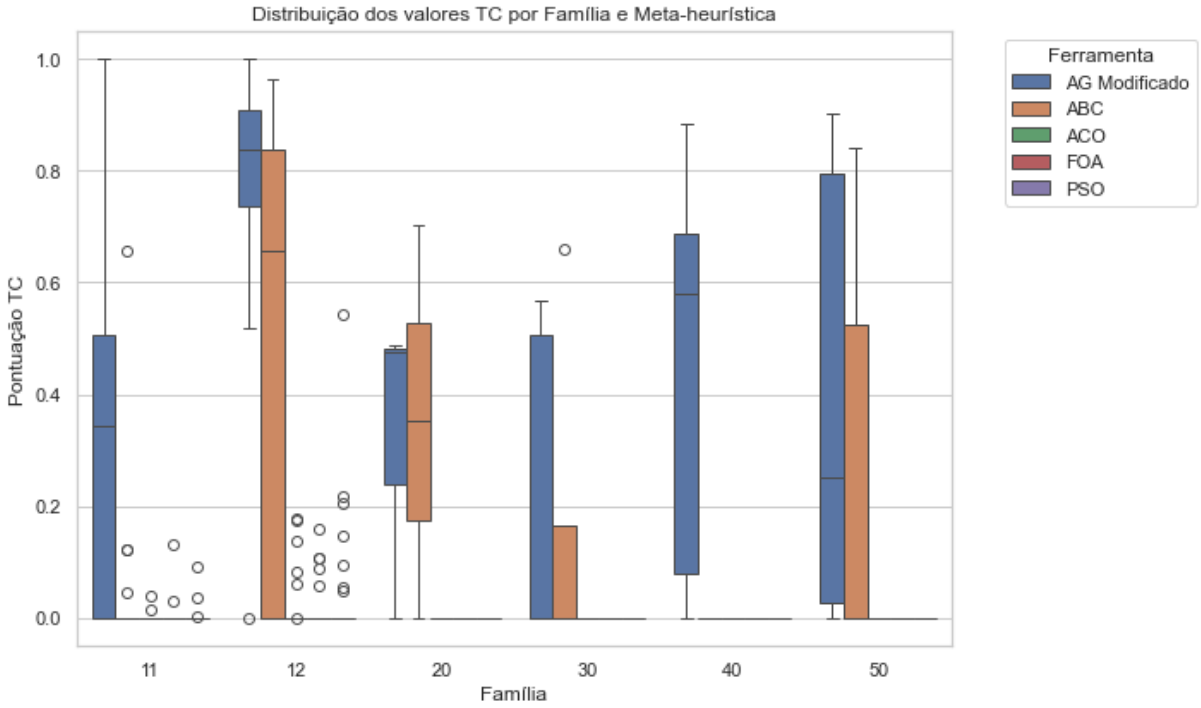


Figura 30 – *Boxplot* ilustrando a distribuição dos valores TC obtidos por todos os algoritmos.

4.3.3 Terceiro conjunto: Algoritmo Genético

Por fim, os últimos testes foram realizados com o objetivo de comparar o desempenho do método proposto em relação ao AG original – representado aqui pela ferramenta MSA-GA – e validar a hipótese de que as alterações propostas e implementadas amenizam o problema de máximo local do algoritmo. Neste contexto, no entanto, ao invés de executar todos os casos de teste do BAliBase, foi selecionada uma amostra aleatória de casos de cada família para executar os testes devido ao alto tempo de execução da MSA-GA, inviabilizando a utilização de toda a base. Os parâmetros da ferramenta foram os mesmos utilizados por Gondro e Kinghorn (2007). Os resultados médios de cada estratégia separados por família são apresentados na Tabela 12.

Tabela 12 – Média dos resultados obtidos para cada versão do AG nas famílias do BAliBase.

Método	**Famílias**											
	11		**12**		**20**		**30**		**40**		**50**	
	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC
Tese	0.5146	0.3094	0.7013	0.4573	0.9030	0.4770	0.7950	0.2530	0.7993	0.4110	0.8760	0.5920
MSA-GA	0.2049	0.0814	0.4364	0.0341	0.2173	0.0000	0.2666	0.0000	0.3626	0.0000	0.3261	0.0135

Pelos valores obtidos é possível notar um desempenho consideravelmente melhor do AG híbrido em relação à MSA-GA. Para verificar e validar que a diferença entre os resultados é significante estatisticamente, o teste de Mann-Whitney U (Mann; Whitney, 1947) (também conhecido como teste de Wilcoxon *rank-sum*) foi aplicado, considerando que o objetivo do teste é avaliar apenas dois grupos amostrais diferentes e independentes sem distribuição específica

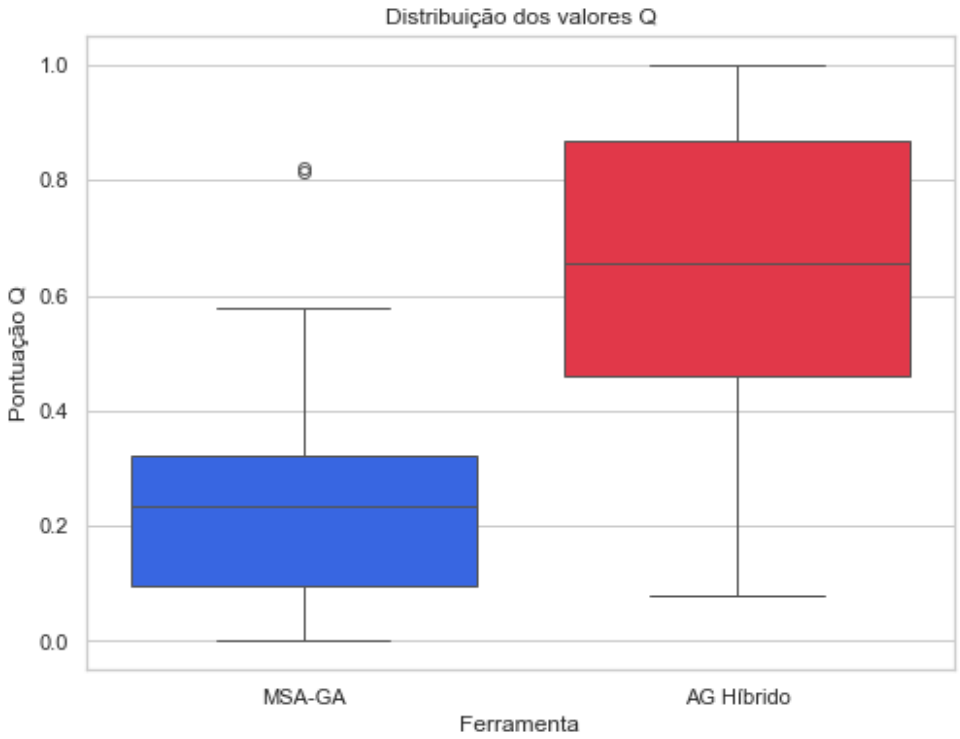


Figura 31 – *Box-plots* da distribuição dos valores Q para a MSA-GA e o presente trabalho.

e especificar se um deles se destaca estatisticamente em relação ao outro (McKnight; Najab, 2010). O valor de confiança utilizado foi também de 5% e os $p - value$ obtidos para Q e TC foram 2.7561×10^{-7} e 1.2698×10^{-5}, reforçando que a diferença entre os grupos é significativa. Observando a média total das pontuações Q e TC de cada ferramenta, observados na Tabela 13, nota-se que a abordagem proposta neste trabalho alcançou 144.15% de melhoria na pontuação Q e 585.82% na pontuação TC em relação à MSA-GA.

Método	**Média Q total**	**Média TC total**
MSA-GA	0.2523	0.0522
AG Híbrido	0.6160	0.3580

Tabela 13 – Médias totais de Q e TC para ambas versões do AG.

Nas figuras 31 e 32 é possível visualizar a distribuição dos dados de ambas as ferramentas, reforçando a diferença entre as mesmas.

De maneira análoga aos dois conjuntos anteriores, também observou-se os dados de ambas versões do AG sob o ponto de vista da homologia. Os valores encontram-se na Tabela 14, e demonstram um comportamento diferente do AG híbrido: desta vez, os resultados obtidos no intervalo >50% foi o esperado dada a circunstância de aumento da similaridade entre as sequências. Isso confronta a hipótese levantada nas seções anteriores de que existe um problema de divergência nestas situações, uma vez que neste conjunto de testes a amostra de casos do BAliBase foi diferente dos demais. Entretanto, ainda é um comportamento que deve ser

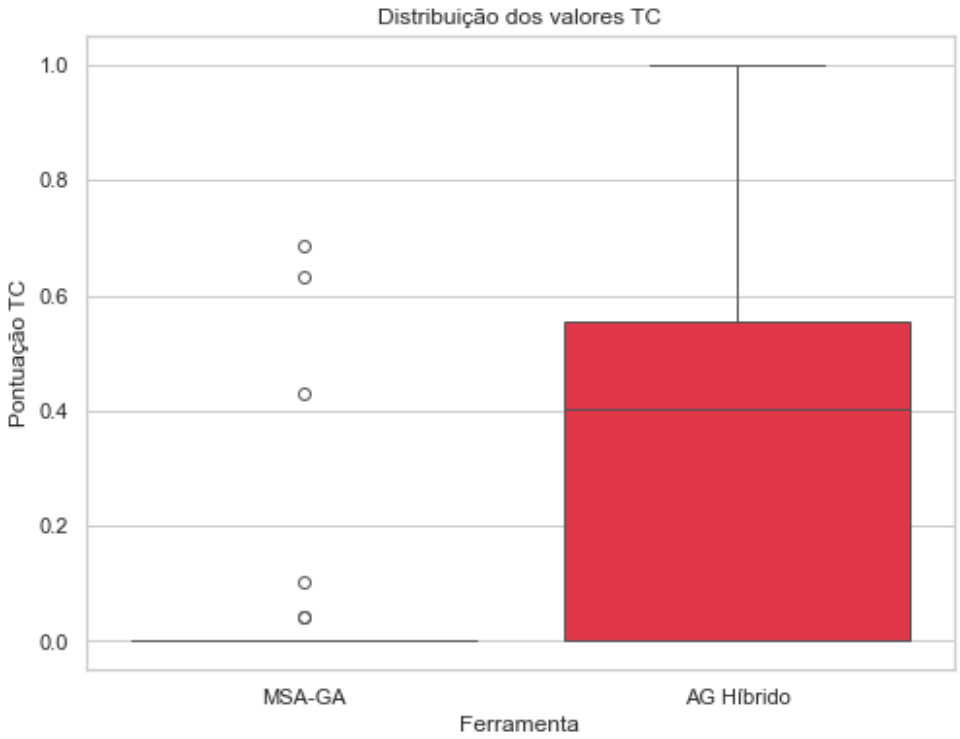


Figura 32 – *Box-plots* da distribuição dos valores TC para a MSA-GA e o presente trabalho.

considerado, analisado e, se validada a hipótese, corrigido. Essa investigação será realizada em trabalhos futuros.

Método	**<20%**	**>20% e <50%**	**>50%**
AG Híbrido	0.2161	0.6748	0.9670
MSA-GA	0.0462	0.2737	0.4628

Tabela 14 – Média dos resultados da estratégia proposta e da MSA-GA de acordo com os intervalos definidos.

Considerando os dados obtidos pelos três conjuntos de teste, é possível afirmar que a estratégia aplicada para melhorar os resultados do AG foi efetiva, validando a hipótese de que as modificações propostas e aplicadas no AG amenizam o problema de máximo local, melhorando significativamente a qualidade biológica dos resultados obtidos pelo AG, superando inclusive outras meta-heurísticas conhecidas. Além disso, a estratégia foi capaz de obter valores comparáveis às principais ferramentas da área, reforçando a qualidade de seus resultados não só como uma modificação do AG, mas também como uma ferramenta para AMS.

5 CONCLUSÃO

Neste trabalho foi apresentada uma estratégia híbrida para amenizar o problema de máximo local no algoritmo genético composta por duas partes: a criação da população inicial por um conjunto de meta-heurísticas e a seleção de operadores de mutação e *crossover* por um agente de aprendizado por reforço. Apesar de ser conhecido pela sua adaptabilidade, o problema apresentado causa a degradação dos resultados do algoritmo, sendo essa questão um alvo de trabalho constante na literatura.

Os resultados obtidos demonstram que a estratégia foi capaz de produzir alinhamentos consideravelmente melhores do que o AG sem modificações e outras meta-heurísticas, e seus resultados foram semelhantes aos resultados de ferramentas amplamente utilizadas, como Clustal Omega, Kalign, MAFFT e MUSCLE. Desse modo, a hipótese de que a hibridização proposta consegue amenizar o problema de máximo local presente no AG, permitindo que o mesmo alcance resultados de maior significância biológica é validada, uma vez que as diferenças de resultado entre o AG e o método implementado são estatisticamente significativas.

Como trabalhos futuros, propõe-se a investigação da hipótese de divergência do método em casos de alta similaridade levantada inicialmente na Seção 4.3.1, assim como a aplicação de outras técnicas de paralelismo a fim de otimizar o tempo de execução da estratégia – especialmente no que diz respeito à criação da população inicial, uma vez que meta-heurísticas tendem a consumir mais tempo – de modo a facilitar a utilização da ferramenta desenvolvida em computadores pessoais. Neste contexto, também visa-se a disponibilização da ferramenta de modo público através de um servidor para permitir seu uso sem a necessidade de instalação local. Pretende-se também disponibilizar publicamente o código fonte através da plataforma GitHub [1].

5.1 PUBLICAÇÕES

Durante o período do presente trabalho diversos artigos científicos foram produzidos a partir deste trabalho e de outros que a aluna participou junto ao Laboratório de Bioinformática. No que diz respeito ao presente trabalho, o artigo intitulado *"Fruit fly optimization algorithm for multiple sequence alignment"* foi submetido ao periódico *Computational Biology and Chemistry* da Elsevier apresentando a aplicação do FOA para o problema de AMS; o artigo *"A Hybrid Genetic Algorithm using Progressive Alignment and Consistency based Approach for Multiple Sequence Alignments"*, que trata sobre modificações implementadas anteriormente no AG durante a fase de estudos sobre o problema de máximo local do AG, complementando estudos anteriores da aluna, foi publicado na *International Conference on Enterprise Information Systems* (ICEIS) no ano de 2022; e o artigo intitulado *"Hybrid genetic algorithm with metaheuristics and reinforcement*

[1] https://github.com/

learning for multiple sequence alignment", englobando todo o presente trabalho e os resultados apresentados, está em fase de confecção.

Quanto a participação da aluna em outros trabalhos do Laboratório de Bioinformática, foram publicados também na ICEIS os seguintes artigos:*"A Hybrid Approach using Progressive and Genetic Algorithms for Improvements in Multiple Sequence Alignments"*, em 2021, *"Optimization of SNP Search Based on Masks Using Graphics Processing Unit"* em 2023, *"Recommendation Systems: A Deep Learning Oriented Perspective"* e *Optimizing Natural Language Processing Applications for Sentiment Analysis* em 2024, e *"Pattern Recognition in Biosequences Using Artificial Immune System"* e *"Multiple Sequence Alignment Using Ant Colony Optimization with Chaotic Jump"* em 2025.

Por fim, um trabalho derivado de uma das disciplinas cursadas pela aluna junto ao Programa de Pós-Graduação em Ciência da Computação foi publicado na conferência *International Conference on Computational Science and Its Applications* (ICCSA) em 2023, intitulado *"Artificial Bee Colony Algorithm for Feature Selection in Fraud Detection Process"*.

REFERÊNCIAS

AHMED, M. H.; GHATGE, M. S.; SAFO, M. K. Hemoglobin: structure, function and allostery. **Vertebrate and invertebrate respiratory proteins, lipoproteins and other body fluid proteins**, Springer, p. 345–382, 2020.

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: **Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining**. [S.l.: s.n.], 2019. p. 2623–2631.

ALBERTS, B. et al. From dna to rna. In: **Molecular Biology of the Cell. 4th edition**. [S.l.]: Garland Science, 2002.

AMORIM, A. R. **Alinhamento múltiplo de sequências utilizando algoritmo genético multifunção e colônia de formigas**. 2017. Dissertação (Mestrado) — Universidade Estadual Paulista (Unesp), 2017.

AMORIM, A. R. **Algoritmo híbrido de otimização de leões multiobjetivo e paralelo para alinhamento múltiplo de sequências**. 2021. Tese (Doutorado) — Universidade de São Paulo, 2021.

AMORIM, A. R. et al. An approach for coffee objective function to global dna multiple sequence alignment. **Computational biology and chemistry**, Elsevier, v. 75, p. 39–44, 2018.

AMORIM, A. R. et al. Metaheuristics for multiple sequence alignment: a systematic review. **Computational biology and chemistry**, Elsevier, p. 107563, 2021.

AMORIM, A. R. et al. Improvements in the sensibility of msa-ga tool using coffee objective function. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2015. v. 574, n. 1, p. 012104.

ARCURI, H. A. et al. Skpdb: a structural database of shikimate pathway enzymes. **BMC bioinformatics**, Springer, v. 11, n. 1, p. 1–7, 2010.

ARULKUMARAN, K. et al. Deep reinforcement learning: A brief survey. **IEEE Signal Processing Magazine**, IEEE, v. 34, n. 6, p. 26–38, 2017.

BAJAJ, A.; SANGWAN, O. P. Study the impact of parameter settings and operators role for genetic algorithm based test case prioritization. In: **Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India**. [S.l.: s.n.], 2019.

BANSAL, J. C.; SHARMA, H.; JADON, S. S. Artificial bee colony algorithm: a survey. **International Journal of Advanced Intelligence Paradigms**, Inderscience Publishers Ltd, v. 5, n. 1-2, p. 123–159, 2013.

BAWONO, P. et al. Multiple sequence alignment. In: **Bioinformatics**. [S.l.]: Springer, 2017. p. 167–189.

BONACCORSO, G. **Machine learning algorithms**. [S.l.]: Packt Publishing Ltd, 2017.

CHAABANE, L. Resolving multiple sequence alignment problem using an intelligent cooperative algorithm. In: IEEE. **2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)**. [S.l.], 2018. p. 1–3.

CHAABANE, L. et al. Particle swarm optimization with tabu search algorithm (pso-ts) applied to multiple sequence alignment problem. In: **Advances in Multidisciplinary Medical Technologies Engineering, Modeling and Findings**. [S.l.]: Springer, 2021. p. 103–114.

CHEN, W. et al. Multiple sequence alignment algorithm based on a dispersion graph and ant colony algorithm. **Journal of computational chemistry**, Wiley Online Library, v. 30, n. 13, p. 2031–2038, 2009.

CHOWDHURY, B.; GARAI, G. A review on multiple sequence alignment from the perspective of genetic algorithm. **Genomics**, Elsevier, v. 109, n. 5-6, p. 419–431, 2017.

CLIFTON, J.; LABER, E. Q-learning: theory and applications. **Annual Review of Statistics and Its Application**, Annual Reviews, v. 7, p. 279–301, 2020.

CORREA, J. M. et al. Parallel simulated annealing for fragment based sequence alignment. In: IEEE. **2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum**. [S.l.], 2012. p. 641–648.

DÁNGELO, G.; PALMIERI, F. Gga: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems. **Information Sciences**, Elsevier, v. 547, p. 136–162, 2021.

DANN, C. et al. Guarantees for epsilon-greedy reinforcement learning with function approximation. In: PMLR. **International conference on machine learning**. [S.l.], 2022. p. 4666–4689.

DOKEROGLU, T. et al. A survey on new generation metaheuristic algorithms. **Computers & Industrial Engineering**, Elsevier, v. 137, p. 106040, 2019.

DORIGO, M. Optimization, learning and natural algorithms. **Ph. D. Thesis, Politecnico di Milano**, 1992.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. **IEEE computational intelligence magazine**, IEEE, v. 1, n. 4, p. 28–39, 2007.

DORIGO, M.; CARO, G. D.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. **Artificial life**, MIT Press, v. 5, n. 2, p. 137–172, 1999.

DTE. **What Is tRNA? What Is tRNA Structure?** 2024. Disponível em: https://www.dnatestingexperts.com/what-is-trna-what-is-trna-structure/. Acesso em 25/03/2025.

DUNN, C. W.; LUO, X.; WU, Z. Phylogenetic analysis of gene expression. **Integrative and comparative biology**, Oxford University Press, v. 53, n. 5, p. 847–856, 2013.

DUNN, O. J. Multiple comparisons among means. **Journal of the American statistical association**, Taylor & Francis, v. 56, n. 293, p. 52–64, 1961.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: IEEE. **MHŚ95. Proceedings of the sixth international symposium on micro machine and human science**. [S.l.], 1995. p. 39–43.

EDGAR, R. C. Muscle: multiple sequence alignment with high accuracy and high throughput. **Nucleic acids research**, Oxford University Press, v. 32, n. 5, p. 1792–1797, 2004.

EDGAR, R. C.; BATZOGLOU, S. Multiple sequence alignment. **Current opinion in structural biology**, Elsevier, v. 16, n. 3, p. 368–373, 2006.

FENG, D.-F.; DOOLITTLE, R. F. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. **Journal of molecular evolution**, Springer, v. 25, p. 351–360, 1987.

FURLANETTO, G. C.; GOMES, V. Z.; BREVE, F. A. Artificial bee colony algorithm for feature selection in fraud detection process. In: SPRINGER. **International Conference on Computational Science and Its Applications**. [S.l.], 2023. p. 535–549.

GOMES, V. Z. et al. A hybrid genetic algorithm using progressive alignment and consistency based approach for multiple sequence alignments. In: **Proceedings of the 24rd International Conference on Enterprise Information Systems (ICEIS 2022)**. [S.l.: s.n.], 2022. p. 167–174.

GONDRO, C.; KINGHORN, B. P. A simple genetic algorithm for multiple sequence alignment. **Genetics and Molecular Research**, Citeseer, v. 6, n. 4, p. 964–982, 2007.

HOGEWEG, P.; HESPER, B. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. **Journal of molecular evolution**, Springer, v. 20, p. 175–186, 1984.

HOLLAND, J. H. **Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.** [S.l.]: U Michigan Press, 1975.

HU, G. et al. Forecasting energy consumption of long-distance oil products pipeline based on improved fruit fly optimization algorithm and support vector regression. **Energy**, Elsevier, v. 224, p. 120153, 2021.

HU, T. et al. Next-generation sequencing technologies: An overview. **Human Immunology**, Elsevier, v. 82, n. 11, p. 801–811, 2021.

HUSSAIN, A.; MUHAMMAD, Y. S. Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. **Complex & intelligent systems**, Springer, v. 6, n. 1, p. 1–14, 2020.

JAFARI, R.; JAVIDI, M. M.; RAFSANJANI, M. K. Using deep reinforcement learning approach for solving the multiple sequence alignment problem. **SN Applied Sciences**, Springer, v. 1, n. 6, p. 1–12, 2019.

JAIN, M. et al. An overview of variants and advancements of pso algorithm. **Applied Sciences**, MDPI, v. 12, n. 17, p. 8392, 2022.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, Springer, v. 31, n. 3, p. 685–695, 2021.

JAYAPRIYA, J.; AROCK, M. A parallel gwo technique for aligning multiple molecular sequences. In: IEEE. **2015 international conference on advances in computing, communications and informatics (ICACCI)**. [S.l.], 2015. p. 210–215.

KARABOGA, D.; AKAY, B. A comparative study of artificial bee colony algorithm. **Applied mathematics and computation**, Elsevier, v. 214, n. 1, p. 108–132, 2009.

KARABOGA, D. et al. **An idea based on honey bee swarm for numerical optimization**. [S.l.], 2005.

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. **Multimedia Tools and Applications**, Springer, v. 80, n. 5, p. 8091–8126, 2021.

KATOH, K. et al. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. **Nucleic acids research**, Oxford University Press, v. 30, n. 14, p. 3059–3066, 2002.

KAYA, M.; KAYA, B.; ALHAJJ, R. A novel multi-objective genetic algorithm for multiple sequence alignment. **International Journal of Data Mining and Bioinformatics**, Inderscience Publishers (IEL), v. 14, n. 2, p. 139–158, 2016.

KELLER, O. et al. A novel hybrid gene prediction method employing protein multiple sequence alignments. **Bioinformatics**, Oxford University Press, v. 27, n. 6, p. 757–763, 2011.

KHAN, S. A.; HE, D.; VALVERDE, J. C. **Pattern Recognition in Bioinformatics**. [S.l.]: Hindawi, 2016.

KHURI, S. A bioinformatics track in computer science. In: **Proceedings of the 39th SIGCSE technical symposium on Computer science education**. [S.l.: s.n.], 2008. p. 508–512.

KIM, J.; PRAMANIK, S.; CHUNG, M. J. Multiple sequence alignment using simulated annealing. **Bioinformatics**, Oxford University Press, v. 10, n. 4, p. 419–426, 1994.

KLEPEIS, J.; PIEJA, M.; FLOUDAS, C. Hybrid global optimization algorithms for protein structure prediction: Alternating hybrids. **Biophysical journal**, Elsevier, v. 84, n. 2, p. 869–882, 2003.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, Taylor & Francis, v. 47, n. 260, p. 583–621, 1952.

KUMAR, A. Encoding schemes in genetic algorithm. **International Journal of Advanced Research in IT and Engineering**, v. 2, n. 3, p. 1–7, 2013.

KUMAR, S.; SHARMA, V. K.; KUMARI, R. A novel hybrid crossover based artificial bee colony algorithm for optimization problem. **arXiv preprint arXiv:1407.5574**, 2014.

KUMAR, S.; STECHER, G.; TAMURA, K. Mega7: molecular evolutionary genetics analysis version 7.0 for bigger datasets. **Molecular biology and evolution**, Society for Molecular Biology and Evolution, v. 33, n. 7, p. 1870–1874, 2016.

LALWANI, S.; SHARMA, H. Multi-objective three level parallel pso algorithm for structural alignment of complex rna sequences. **Evolutionary Intelligence**, Springer, p. 1–9, 2019.

LASSMANN, T. **Kalign 3: multiple sequence alignment of large datasets**. [S.l.]: Oxford University Press, 2020.

LEE, Z.-J. et al. Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. **Applied Soft Computing**, Elsevier, v. 8, n. 1, p. 55–78, 2008.

LEI, X. et al. Identification of dynamic protein complexes based on fruit fly optimization algorithm. **Knowledge-Based Systems**, Elsevier, v. 105, p. 270–277, 2016.

LEMOS, M.; aO, M. V. S. P. d. A.; CASANOVA, M. A. **Padrões em Biossequências**. [S.l.], 2003.

LEMOS, M.; CASANOVA, M. A. **Algoritmos para Análises de Sequências**. [S.l.], 2000.

LIM, S. M. et al. Crossover and mutation operators of genetic algorithms. **International journal of machine learning and computing**, v. 7, n. 1, p. 9–12, 2017.

LÖYTYNOJA, A. Phylogeny-aware alignment with prank. **Multiple sequence alignment methods**, Springer, p. 155–170, 2014.

LUO, J.; ZHANG, L.; LIANG, C. A multigroup parallel genetic algorithm for multiple sequence alignment. In: SPRINGER. **International Conference on Artificial Intelligence and Computational Intelligence**. [S.l.], 2011. p. 308–316.

MACALINO, S. J. Y. et al. Role of computer-aided drug design in modern drug discovery. **Archives of pharmacal research**, Springer, v. 38, p. 1686–1701, 2015.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**, v. 9, n. 1, p. 381–386, 2020.

MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, JSTOR, p. 50–60, 1947.

MANN, M.; HENDRICKSON, R. C.; PANDEY, A. Analysis of proteins and proteomes by mass spectrometry. **Annual review of biochemistry**, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 70, n. 1, p. 437–473, 2001.

MCKIGHT, P. E.; NAJAB, J. Kruskal-wallis test. **The corsini encyclopedia of psychology**, Wiley Online Library, p. 1–1, 2010.

MCKNIGHT, P. E.; NAJAB, J. Mann-whitney u test. **The Corsini encyclopedia of psychology**, Wiley Online Library, p. 1–1, 2010.

MEU DNA. **A história do DNA: Parte 2**. 2021. Disponível em: https://blog.meudna.com/a-historia-do-dna-parte-2/. Acesso em 25/03/2025.

MISHRA, A.; SOAM, S. S.; TRIPATHI, B. K. An optimization approach for multiple sequence alignment using divide-conquer and genetic algorithm. **International Journal of Advanced Computer Science and Applications**, Science and Information (SAI) Organization Limited, v. 12, n. 4, 2021.

MISTELI, T. Beyond the sequence: cellular organization of genome function. **Cell**, Elsevier, v. 128, n. 4, p. 787–800, 2007.

MITIĆ, M. et al. Chaotic fruit fly optimization algorithm. **Knowledge-based systems**, Elsevier, v. 89, p. 446–458, 2015.

MOSS, J.; JOHNSON, C. G. An ant colony algorithm for multiple sequence alignment in bioinformatics. In: SPRINGER. **Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Roanne, France, 2003**. [S.l.], 2003. p. 182–186.

NASTESKI, V. An overview of the supervised machine learning methods. **Horizons. b**, v. 4, n. 51-62, p. 56, 2017.

NEEDLEMAN, S. B.; WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. **Journal of molecular biology**, Elsevier, v. 48, n. 3, p. 443–453, 1970.

NEUMANN, F.; WITT, C. Combinatorial optimization and computational complexity. In: **Bioinspired computation in combinatorial optimization**. [S.l.]: Springer, 2010. p. 9–19.

NINTENDO. **Animal Crossing New Horizons [Jogo Eletrônico]**. [S.l.]: Nintendo, 2020.

NOTREDAME, C.; HIGGINS, D. G. Saga: sequence alignment by genetic algorithm. **Nucleic acids research**, Oxford University Press, v. 24, n. 8, p. 1515–1524, 1996.

NOTREDAME, C.; HOLM, L.; HIGGINS, D. G. Coffee: an objective function for multiple sequence alignments. **Bioinformatics (Oxford, England)**, v. 14, n. 5, p. 407–422, 1998.

NUTE, M.; SALEH, E.; WARNOW, T. Evaluating statistical multiple sequence alignment in comparison to other alignment methods on protein data sets. **Systematic biology**, Oxford University Press, v. 68, n. 3, p. 396–411, 2019.

ORZECHOWSKI, P.; BORYCZKO, K. Parallel approach for visual clustering of protein databases. **Computing and Informatics**, v. 29, n. 6+, p. 1221–1231, 2010.

PAN, Q.-K. et al. An improved fruit fly optimization algorithm for continuous function optimization problems. **Knowledge-Based Systems**, Elsevier, v. 62, p. 69–83, 2014.

PAN, W.-T. A new fruit fly optimization algorithm: taking the financial distress model as an example. **Knowledge-Based Systems**, Elsevier, v. 26, p. 69–74, 2012.

PAREEK, C. S.; SMOCZYNSKI, R.; TRETYN, A. Sequencing technologies and genome sequencing. **Journal of applied genetics**, Springer, v. 52, p. 413–435, 2011.

PIERRI, C. L.; PARISI, G.; PORCELLI, V. Computational approaches for protein function prediction: a combined strategy from multiple sequence alignment to molecular docking-based virtual screening. **Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics**, Elsevier, v. 1804, n. 9, p. 1695–1712, 2010.

RANJAN, R. K.; KUMAR, V. A systematic review on fruit fly optimization algorithm and its applications. **Artificial Intelligence Review**, Springer, v. 56, n. 11, p. 13015–13069, 2023.

RODRIGUEZ, P. F.; NINO, L. F.; ALONSO, O. M. Multiple sequence alignment using swarm intelligence. **International Journal of Computational Intelligence Research**, Research India Publications, v. 3, n. 2, p. 123–130, 2007.

RUBIO-LARGO, Á.; VEGA-RODRÍGUEZ, M. A.; GONZÁLEZ-ÁLVAREZ, D. L. Hybrid multiobjective artificial bee colony for multiple sequence alignment. **Applied Soft Computing**, Elsevier, v. 41, p. 157–168, 2016.

SANGER, F.; COULSON, A. R. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. **Journal of molecular biology**, Elsevier, v. 94, n. 3, p. 441–448, 1975.

SHAFEE, T.; LOWE, R. Eukaryotic and prokaryotic gene structure. **WikiJournal of Medicine**, v. 4, n. 1, p. 1–5, 2017.

SHAKYA, A. K.; PILLAI, G.; CHAKRABARTY, S. Reinforcement learning algorithms: A brief survey. **Expert Systems with Applications**, Elsevier, v. 231, p. 120495, 2023.

SIEVERS, F.; HIGGINS, D. G. Clustal omega for making accurate alignments of many protein sequences. **Protein Science**, Wiley Online Library, v. 27, n. 1, p. 135–145, 2018.

SMITH, T. F.; WATERMAN, M. S. et al. Identification of common molecular subsequences. **Journal of molecular biology**, Elsevier Science, v. 147, n. 1, p. 195–197, 1981.

THIS ONE VS THAT ONE. **Prokaryotes vs Eukaryotes**. 2017. Disponível em: https://thisonevsthatone.com/prokaryotes-vs-eukaryotes. Acesso em 25/03/2025.

THOMPSON, J. D.; HIGGINS, D. G.; GIBSON, T. J. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. **Nucleic acids research**, Oxford university press, v. 22, n. 22, p. 4673–4680, 1994.

THOMPSON, J. D. et al. Balibase 3.0: latest developments of the multiple sequence alignment benchmark. **Proteins: Structure, Function, and Bioinformatics**, Wiley Online Library, v. 61, n. 1, p. 127–136, 2005.

THOMSEN, R.; BOOMSMA, W. Multiple sequence alignment using saga: investigating the effects of operator scheduling, population seeding, and crossover operators. In: SPRINGER. **Workshops on applications of evolutionary computation**. [S.l.], 2004. p. 113–122.

TRIVEDI, R.; NAGARAJARAM, H. A. Substitution scoring matrices for proteins-an overview. **Protein Science**, Wiley Online Library, v. 29, n. 11, p. 2150–2163, 2020.

VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In: IEEE. **2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)**. [S.l.], 2016. p. 261–265.

WANG, L.; JIANG, T. On the complexity of multiple sequence alignment. **Journal of computational biology**, v. 1, n. 4, p. 337–348, 1994.

WATSON, J. D.; CRICK, F. H. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. **Nature**, Nature Publishing Group UK London, v. 171, n. 4356, p. 737–738, 1953.

WONG, Y.-Y. et al. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. **Soft Computing**, Springer, v. 7, n. 8, p. 506–515, 2003.

XIANG, X. et al. Ant colony with genetic algorithm based on planar graph for multiple sequence alignment. **Information Technology Journal**, Science Alert, v. 9, n. 2, p. 274–281, 2010.

XIONG, C.; LIAN, S. Structural damage identification based on improved fruit fly optimization algorithm. **KSCE Journal of Civil Engineering**, Springer, v. 25, p. 985–1007, 2021.

XU, F.; CHEN, Y. A method for multiple sequence alignment based on particle swarm optimization. In: SPRINGER. **Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence: 5th International Conference on Intelligent Computing, ICIC 2009 Ulsan, South Korea, September 16-19, 2009 Proceedings 5**. [S.l.], 2009. p. 965–973.

ZAFALON, G. et al. A parallel approach of coffee objective function to multiple sequence alignment. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2015. v. 633, n. 1, p. 012084.

ZAFALON, G. F. D. et al. A hybrid approach using progressive and genetic algorithms for improvements in multiple sequence alignments. In: **Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021)**. [S.l.: s.n.], 2021. p. 384–391.

ZAHA, A. et al. **Biologia molecular básica**. [S.l.]: Mercado Aberto, 2000. v. 3.

ZHANG, X.-D. Machine learning. In: **A Matrix Algebra Approach to Artificial Intelligence**. [S.l.]: Springer, 2020. p. 223–440.

ZHOU, Y. et al. Parallel ant colony optimization on multi-core simd cpus. **Future Generation Computer Systems**, Elsevier, v. 79, p. 473–487, 2018.

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Vitoria Zanon Gomes Data de Nascimento: 21/02/1999
Nacionalidade	Brasileira
Nome em citações bibliográficas	GOMES, VITORIA GOMES, V. Z.
Currículo Lattes	http://lattes.cnpq.br/3202598317451099
ORCID	https://orcid.org/0000-0003-4176-566X
Google Scholar	https://scholar.google.com/citations?user=fD3QTQcAAAAJ&hl=pt-BR&oi=sra
FORMAÇÃO ACADÊMICA	
2017/2020	Bacharela em Ciência da Computação Universidade Estadual Paulista "Júlio de Mesquita Filho"
2021/2025	Doutora em Ciência da Computação Universidade Estadual Paulista "Júlio de Mesquita Filho"
PRODUÇÃO BIBLIOGRÁFICA	
ZAFALON, GERALDO ; GOMES, VITORIA ; AMORIM, ANDERSON ; VALÊNCIO, CARLOS. A Hybrid Approach using Progressive and Genetic Algorithms for Improvements in Multiple Sequence Alignments. *In:* 23rd International Conference on Enterprise Information Systems, 2021, Online Streaming. **Proceedings of the 23rd International Conference on Enterprise Information Systems**, 2021. v. 2. p. 384-391. GOMES, VITORIA; ANDRADE, MATHEUS ; AMORIM, ANDERSON ; ZAFALON, GERALDO . A Hybrid Genetic Algorithm using Progressive Alignment and Consistency based Approach for Multiple Sequence Alignments. *In:* 24th International Conference on Enterprise Information Systems, 2022, Online Streaming. **Proceedings of the 24th International Conference on Enterprise Information Systems, 2022**., 2022. v. 2. p. 167-174	

NOGUEIRA DA CRUZ, ÁLVARO ; GOMES, VITORIA ; ANDRADE, MATHEUS ; AMORIM, ANDERSON ; VALÊNCIO, CARLOS ; VAUGHAN, GILBERTO ; ZAFALON, GERALDO. Optimization of SNP Search Based on Masks Using Graphics Processing Unit. *In:* 25th International Conference on Enterprise Information Systems, 2023, Prague. **Proceedings of the 25th International Conference on Enterprise Information Systems**, 2023. p. 134

LAMPA, IGOR ; GOMES, VITORIA ; ZAFALON, GERALDO. Recommendation Systems: A Deep Learning Oriented Perspective. *In:* 26th International Conference on Enterprise Information Systems, 2024, Angers. **Proceedings of the 26th International Conference on Enterprise Information Systems**, 2024. p. 682.

LOPES, ANDERSON ; GOMES, VITORIA ; ZAFALON, GERALDO. Optimizing Natural Language Processing Applications for Sentiment Analysis. *In:* 26th International Conference on Enterprise Information Systems, 2024, Angers. **Proceedings of the 26th International Conference on Enterprise Information Systems**, 2024. p. 698.

LIBERATO, LUIZ ; NOGUEIRA DA CRUZ, ÁLVARO ; AMORIM, ANDERSON ; GOMES, VITORIA ; RODRIGUES DA SILVEIRA, BRUNO ; PREVATO, GABRIEL ; CAVARÇAN, LUIZA ; VALÊNCIO, CARLOS ; ZAFALON, GERALDO. Pattern Recognition in Biosequences Using Artificial Immune System. *In:* 27th International Conference on Enterprise Information Systems, 2025, Porto. **Proceedings of the 27th International Conference on Enterprise Information Systems**, 2025. p. 797.

LINO DE FREITAS, MATHEUS ; ANDRADE, MATHEUS ; AMORIM, ANDERSON ; GOMES, VITORIA ; RODRIGUES DA SILVEIRA, BRUNO ; PREVATO, GABRIEL ; CAVARÇAN, LUIZA ; VALÊNCIO, CARLOS ; ZAFALON, GERALDO. Multiple Sequence Alignment Using Ant Colony Optimization with Chaotic Jump. *In:* 27th International Conference on Enterprise Information Systems, 2025, Porto. **Proceedings of the 27th International Conference on Enterprise Information Systems**, 2025. p. 237.

FURLANETTO, Gabriel Covello; GOMES, Vitoria Zanon; BREVE, Fabricio Aparecido. Artificial Bee Colony Algorithm for Feature Selection in Fraud Detection Process. *In:* **International Conference on Computational Science and Its Applications**. Cham: Springer Nature Switzerland, 2023. p. 535-549.

PARTICIPAÇÃO EM BANCAS E ORIENTAÇÕES

Bancas de trabalhos de conclusão

Orientações

PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS

3RD WOMEN IN BIOINFORMATICS & DATA SCIENCE LA CONFERENCE., 2022, (Online Streaming). Liponium: An algorithm for identification of discrete traits associated with Mycobacterium tuberculosis hetero-resistance and virulence. 2022. (Apresentação de Trabalho).

XXXII SEMAC - Semana da Computação da UNESP de São José do Rio Preto, 2022, (São José do Rio Preto - SP). Bioinformática. 2022. (Palestra).

XXXIV SEMAC - Semana da Computação da UNESP de São José do Rio Preto, 2024, (São José do Rio Preto - SP). Mini-curso: Alinhamento Múltiplo de Sequências com Python. 2024. (Minicurso).

XII SEFIS - Semana da Física, 2024, (São José do Rio Preto - SP). Mini-curso: Aplicações de Aprendizado de Máquina em Biologia Computacional. 2024. (Minicurso).