UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Matheus Lino de Freitas

# Alinhamento múltiplo de sequências utilizando Ant Colony Optimization e Chaotic Jump

**São José do Rio Preto**
**2021**

Matheus Lino de Freitas

# Alinhamento múltiplo de sequências utilizando Ant Colony Optimization e Chaotic Jump

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador:

Prof. Dr. Geraldo Francisco Donegá Zafalon

**São José do Rio Preto**
**2021**

F866a

Freitas, Matheus Lino de
Alinhamento múltiplo de sequências utilizando Ant Colony Optimization e Chaotic Jump / Matheus Lino de Freitas. -- São José do Rio Preto, 2021
77 f. : il., tabs.

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

1. Alinhamento Múltiplo de Sequências. 2. Chaotic Jump. 3. Ant Colony Optimization. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Matheus Lino de Freitas

# Alinhamento múltiplo de sequências utilizando Ant Colony Optimization e Chaotic Jump

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Prof. Dr. Geraldo Francisco Donegá Zafalon
UNESP – Câmpus de São José do Rio Preto
Orientador

Prof. Dra. Carina Alexandra Rondini
UNESP – Câmpus de São José do Rio Preto

Prof. Dr. Diego Renan Bruno
FATEC – Câmpus Catanduva

**São José do Rio Preto**
**8 de setembro de 2021**

*A Deus e a toda minha família.*

# Agradecimentos

Agradeço a Deus, pela sua força, emprestada em todos esses anos e por permitir que as melhores oportunidades chegassem até mim. Também agradeço imensamente ao professor e meu orientador Geraldo Francisco Donegá Zafalon, que teve a maior paciência do mundo nos momentos mais difíceis, aos quais eu quase desisti devidas as dificuldades para conciliar as agendas familiares, profissionais e acadêmicas, bem como por toda sua contribuição de excelência para que eu pudesse atingir o objetivo do trabalho. Agradeço a todos os professores do PPGCC, ao qual tive o maior prazer em conhecê-los, em especial para a Prof. Dra. Rogéria Cristiane Gratão de Souza e o Prof. Dr. Leandro Alves Neves. Agradeço aos companheiros e a todo o pessoal do laboratório de Inteligência Computacional: Álvaro Magri, Matheus Carreira, Luiz Paulo, Cléber Ivo, Guilherme Botazzo e também às outras pessoas que fizeram essa caminhada ficar mais leve e descontraída. Agradeço muito ao Anderson Rici Amorim, por toda sua contribuição ao meu trabalho, compartilhando sua experiência para eu produzir um texto melhor. Agradeço à minha esposa, Laiane, pelo seu incentivo e por sempre confiar em mim e ficar ao meu lado durante todo esse tempo, em que eu não fui tão presente quanto eu gostaria.

## *Resumo*

Alinhamento múltiplo de sequências é uma das técnicas mais relevantes no contexto de bioinformática. Os sequenciadores modernos produzem um grande volume de dados que são posteriormente analisados por biólogos, biomédicos e profissionais da área genética. Devido a esse grande volume, estratégias computacionais são necessárias para auxiliar na análise dos dados, como por exemplo, os alinhamentos de sequências. A tarefa de alinhar sequências é um desafio computacional e biológico. Do ponto de vista biológico, os modelos são incompletos e não levam em consideração todos os aspectos estruturais e evolutivos das espécies. Além disso, do ponto de vista computacional, com os *hardwares* atuais, soluções exatas podem não ser obtidas em um tempo hábil. A alternativa prática é a utilização de heurísticas e modelos probabilísticos para se obter resultados com significância biológica, dentro de um tempo factível. Entretanto, heurísticas possuem a característica de se fixarem em pontos de máximo ou mínimo local e, deste modo, as soluções tornam-se sub-ótimas. Para amenizar esse problema recorre-se à utilização de estratégias híbridas e aplicação de estado caótico no algoritmo para deslocar a solução de um ponto no espaço de busca para outro. Portanto, o presente trabalho desenvolveu-se um novo método que, por meio de uma combinação entre as ferramentas KAlign e Clustal Ômega, produz um alinhamento inicial e efetua seu refinamento com otimização por colônia de formigas e *chaotic jump*. Houve a obtenção de 100% de melhores resultados quando comparados com a ferramenta MSA-GA e pelo menos 50% de melhores resultados quando comparados com as ferramentas KAlign e Clustal Ômega, para todas as famílias do *benchmark* utilizadas.



**Palavras-chave:** Alinhamento Múltiplo de Sequências. Colônia de Formigas. Abordagens Híbridas. *Chaotic Jump*.

*Abstract*

Multiple sequence alignment is one of the most relevant techniques in the bioinformatics field. The next generation sequencing technologies produces a large volume of data that is later analised by biologists, biomedicals and geneticists. Due to this huge volume, computational effort are necessary to aid in the data analisys, as an example, for sequence alignment. The sequence alignment task is a both a computational and biological challenge. From the biological perspective, the abstract models are incomplete and it does not take into consideration every structural and evolutionary aspect of the species. Besides, from the computational perspective, with currently available hardwares, exact solutions cannot be found in reasonable time. The practical alternative is to use heuristics and probabilistic models to reach results with biological meaning in a feasible time. However, heuristics has a particular characteristic of falling in local maxima or local minima and therefore the solutions that are found could be suboptimal. In order to ease this problem, one can appel to the usage of hybrid strategies and the application of chaotic state in the algorithm to jump the solution in the search space from one point to another. Thus, this works aims to intruduce a novel method that combines the KAlign and Clustal Omega tools in order to produce a seed alignment that will later be refined by Ant Colony Optimization and Chaotic Jump. The results showed us that for every test the ACO produced better alignments than the MSA-GA tool and at least for 50% of tests the proposed method was able to improved the initial alignments produced by the KAlign and Clustal Omega tools.

**Keywords:** Multiple Sequence Alignment. Ant Colony Optimization. Hybrid Approaches. Chaotic Jump.

# Lista de Ilustrações

# Lista de Tabelas

# Lista de Abreviações

*ACO* *Ant Colony Optimization*

*ADN* Ácido Desoxirribonucleico

*AMS* Alinhamento múltiplo de sequências

*ARN* Ácido Ribonucleico

*BLOSUM* *Blocks of Amino Acid Substitution Matrix*

*PAM* *Percent Accepted Mutation*

*TRF* Transformada Rápida de Fourier

*WSP* *Weighted Sum of Pairs*

# Sumário

# Capítulo 1

# Introdução

## 1.1 Considerações Iniciais

Após a descoberta da estrutura química da molécula de ácido desoxirribonucleico (ADN) em 1953, do código genético e do fluxo da informação biológica das moléculas, ocorreu o surgimento de uma nova área, a bioinformática. Essa área envolve diversas linhas de conhecimento e tem o papel fundamental de prover soluções computacionais à diversos problemas biológicos reais (PROSDOCIMI et al., 2002). Após mais de 30 anos essa área ganhou cada vez mais expressividade, graças à evolução dos *hardwares* e do crescimento do número de informações disponíveis, mas existem ainda muitos problemas a serem enfrentados (ZAFALON, 2009). Por consequência, os alinhamentos de sequências tornaram-se relevantes e necessários para a análise dessa grande quantidade de informações (SIEVERS; HIGGINS, 2014). Com isso, o presente trabalho propõe uma contribuição para um conceito muito utilizado em bioinformática, que é o alinhamento múltiplo de sequências.

A tarefa de alinhar múltiplas sequências é de grande utilidade por biólogos, pois o resultado do alinhamento é base para análises e construções de modelos que auxiliam a compreender a evolução das espécies, o funcionamento das proteínas dos organismos, o entendimento do comportamento de doenças, dentre outras aplicações (THOMSEN;

BOOMSMA, 2004).

Existem diversas estratégias para efetuar o alinhamento de múltiplas sequências e cada uma dessas estratégias possuem certas vantagens e desvantagens. Estas podem ser tanto do ponto de vista computacional, como tempo de processamento, consumo de memória do computador, utilização frequente de discos rígidos, quanto do ponto de vista biológico, como a qualidade do alinhamento produzido, o objetivo biológico no qual cada estratégia prioriza e como as informações são apresentadas aos biólogos.

Portanto, procura-se desenvolver métodos e abordagens que reduzam o tempo de processamento dos algoritmos e das ferramentas, sem prejudicar a qualidade dos resultados. Nesse sentido, a utilização de técnicas, cujas aplicações em outras áreas apresentaram bons resultados, é uma justificativa plausível.

## 1.2 Motivação e Escopo

O alinhamento múltiplo de sequências é uma das áreas da bioinformática de maior importância. Por meio dessa técnica, os biólogos e pesquisadores dessa área de estudo podem entender a evolução de uma espécie, entender o comportamento e até encontrar cura para doenças.

Devida a existência de múltiplos métodos e ferramentas no contexto de alinhamento múltiplo de sequências é necessário que pesquisadores da área sejam instruídos em conceitos biológicos. O entendimento sobre sequências de aminoácidos e nucleotídeos, bem como um sólido entendimento do dogma central da biologia, são importantes para compreensão dessas ferramentas e métodos.

Além disso, uma visão geral dos principais métodos e ferramentas contribuem para a construção de abordagens mais sólidas. Pois é necessário efetuar comparações com os métodos e ferramentas existentes para aferir a qualidade de novos métodos e novas ferramentas bem como para melhorar os existentes.

Diante desses desafios, surgiu a hipótese que a combinação de métodos ou ferra-

mentas existentes, que particularmente possuem pontos positivos e negativos com uma abordagem caótica, podem auxiliar a exploração do espaço de busca em uma abordagem heurística, evitando soluções máximos ou mínimos locais e apresentar resultados biologicamente mais adequados.

## 1.3 Justificativa

Como o alinhamento de sequência é da classe de problemas denominados *Non-Deterministic Polinomial-Time Hard* ou NP-Difícil, as abordagens exatas e até mesmo aproximadas não são utilizadas para grandes volumes de sequências. Recorrem-se às heurísticas e modelos estocásticos, pois essas abordagens podem encontrar soluções com boa qualidade biológica dentro de um tempo factível.

Mesmo as heurísticas, podem não convergir para um resultado considerado aceitável pois podem estagnar em pontos de máximo ou mínimo local. Portanto, além de um bom modelo que ajude a prevenir essas situações, abordagens caóticas, por exemplo, também são aplicadas para esse fim na literatura.

Ademais, soluções híbridas são outra manobra utilizada para melhorar o resultado esperado. O trabalho de (LEE et al., 2008) propõe uma abordagem híbrida utilizando Algoritmo Genético para construção do alinhamento e Otimização por colônia de formigas para um refinamento. Outra linha de solução híbrida é a combinação de ferramentas, como pode ser observada no trabalho de (RUBIO-LARGO; VEGA-RODRÍGUEZ; GONZÁLEZ-ÁLVAREZ, 2016) que utiliza a ferramenta KAlign para efetuar um realinhamento de fragmentos das sequências.

O presente trabalho leva essas questões práticas em consideração e propõe uma solução para alinhamento múltiplo de sequências utilizando uma solução híbrida com duas ferramentas: KAlin e Clustal Ômega em conjunto com um refinamento de otimização por colônia de formigas combinado com *chaotic jump* visando reduzir a probabilidade de atingir máximos locais e atingir resultados de maior qualidade.

## 1.4 Objetivos

O objetivo deste trabalho é o desenvolvimento de um método para alinhamento múltiplo de sequências, utilizando a meta-heurística *Ant Colony Optimization* e buscando obter resultados que sejam biologicamente relevantes. Para esse fim, estratégias híbridas foram empregadas, como a combinação de ferramentas para geração de um alinhamento base, refinamento do alinhamento por meio da meta-heurística citada e aplicação do conceito de *chaotic jump* combinado com realinhamento parcial das sequências, se o algoritmo adentrar um ponto de máximo local.

Finalmente, a ideia é que o método proposto contribua para a área de estudo em razão da qualidade do alinhamento múltiplo produzido.

# Capítulo 2

# Fundamentação Teórica

Neste capítulo, são apresentados os fundamentos da biologia molecular, os conceitos fundamentais da área de bioinformática e algumas estratégias para alinhamentos de sequências.

## 2.1 Biologia molecular

Na biologia molecular procura-se compreender as relações entre os vários sistemas da célula, quais são seus constituintes moleculares e quais processos ocorrem nesse meio (ZAHA; FERREIRA; PASSAGLIA, 2014). Alguns fundamentos mínimos de biologia molecular que serão apresentados nas próximas subseções são: a organização celular, o que são os ácidos nucleicos e quais suas funções e importância, como funcionam os processos de codificação genética e síntese proteica e filogenia. Por meio desses fundamentos é possível entender diversos métodos e técnicas da bioinformática, dentre eles a proposta do trabalho atual.

### 2.1.1 Organização celular

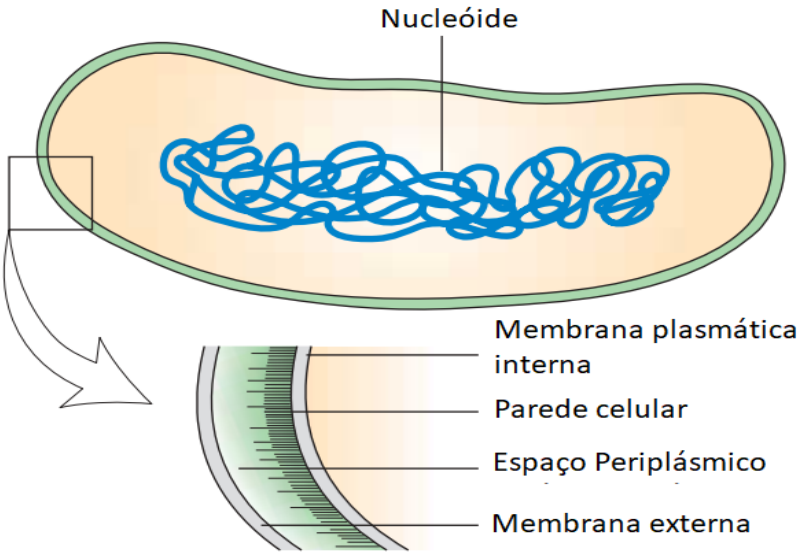
A célula é uma pequena estrutura que constitui toda matéria viva e, embora seja a unidade mais básica de vida, são diversas e complexas. Mesmo com essa diversidade,

ao observar a nível molecular, percebe-se a existência de características similares entre elas, como: a arquitetura de suas membranas, processos metabólicos, replicação de ADN, síntese proteica e produção de energia química (ZAHA; FERREIRA; PASSAGLIA, 2014).

Pode-se classificar as células em dois tipos: procarióticas e eucarióticas (KAISER et al., 2007). As células procarióticas, conforme representadas na Figura 2.1, consistem em um único compartimento de matéria envolvido por uma membrana plasmática, não possuem um núcleo definido e possuem uma organização interna mais simples que as eucarióticas (KAISER et al., 2007). Os organismos que constituem esse grupo são organismos unicelulares, que podem se associar a outros organismos para formar colônias. As bactérias, por exemplo, são seres procarióticos (ZAHA; FERREIRA; PASSAGLIA, 2014).

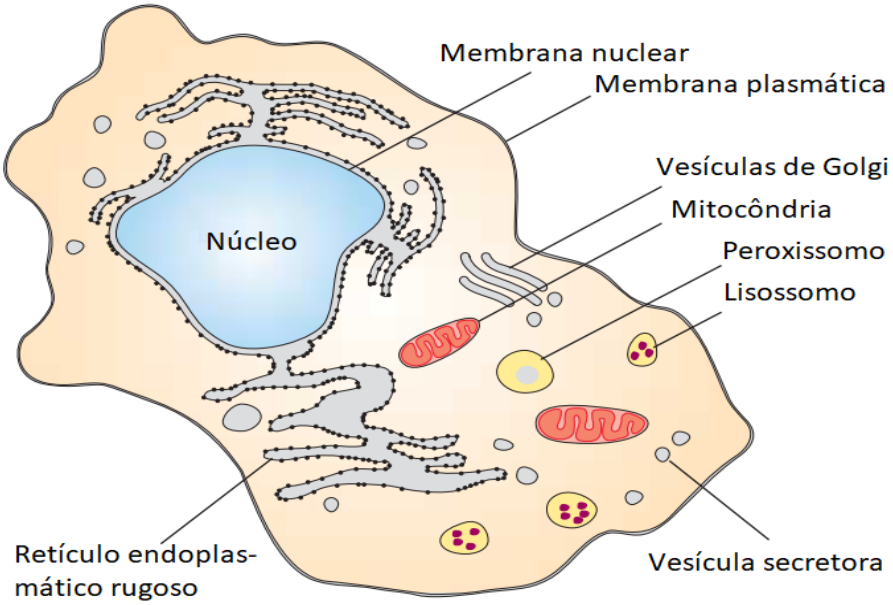
As células eucarióticas, representadas na Figura 2.2, possuem um envoltório nuclear bem definido e uma série de outros compartimentos internos chamados organelas. Entre o núcleo e a membrana plasmática existe um meio aquoso, repleto de citosol, denominado citoplasma no qual situam-se as organelas. Os organismos desse grupo incluem todos os membros do reino animal e vegetal, incluem também organismos do reino fungi e protista e podem se apresentar tanto como multicelulares quanto unicelulares (KAISER et al., 2007).

Figura 2.1: Célula procariótica.




Fonte: Adaptado de KAISER et al. (2007)

Figura 2.2: Célula eucariótica. Adaptado de KAISER et al. (2007)




Fonte: Adaptado de (KAISER et al., 2007)

Uma característica importante da célula é sua capacidade de se reproduzir. Os mamíferos, por exemplo, originam-se de uma única célula chamada zigoto, que contém a

informação necessária para construção de todas as outras células no organismo (aproximadamente 100 trilhões de células (KAISER et al., 2007)). Essas informações estão, em todas as células, num mesmo código químico: o ADN (ALBERTS et al., 2010).

### 2.1.2 Proteínas e aminoácidos

Os constituintes moleculares da célula são, dentre outros elementos químicos: água, íons, carboidratos, lipídios e duas importantes substâncias moleculares distintas por tamanho e organização, denominadas moléculas pequenas e macromoléculas. As moléculas pequenas levam esse nome devido ao seu número de átomos, pois são compostas por pouco mais de 50 átomos. As macromoléculas, ou polímeros biológicos, são maiores, daí o nome macromoléculas e são formadas por cópias de moléculas pequenas, ligadas por meio de ligação covalente (ZAHA; FERREIRA; PASSAGLIA, 2014).

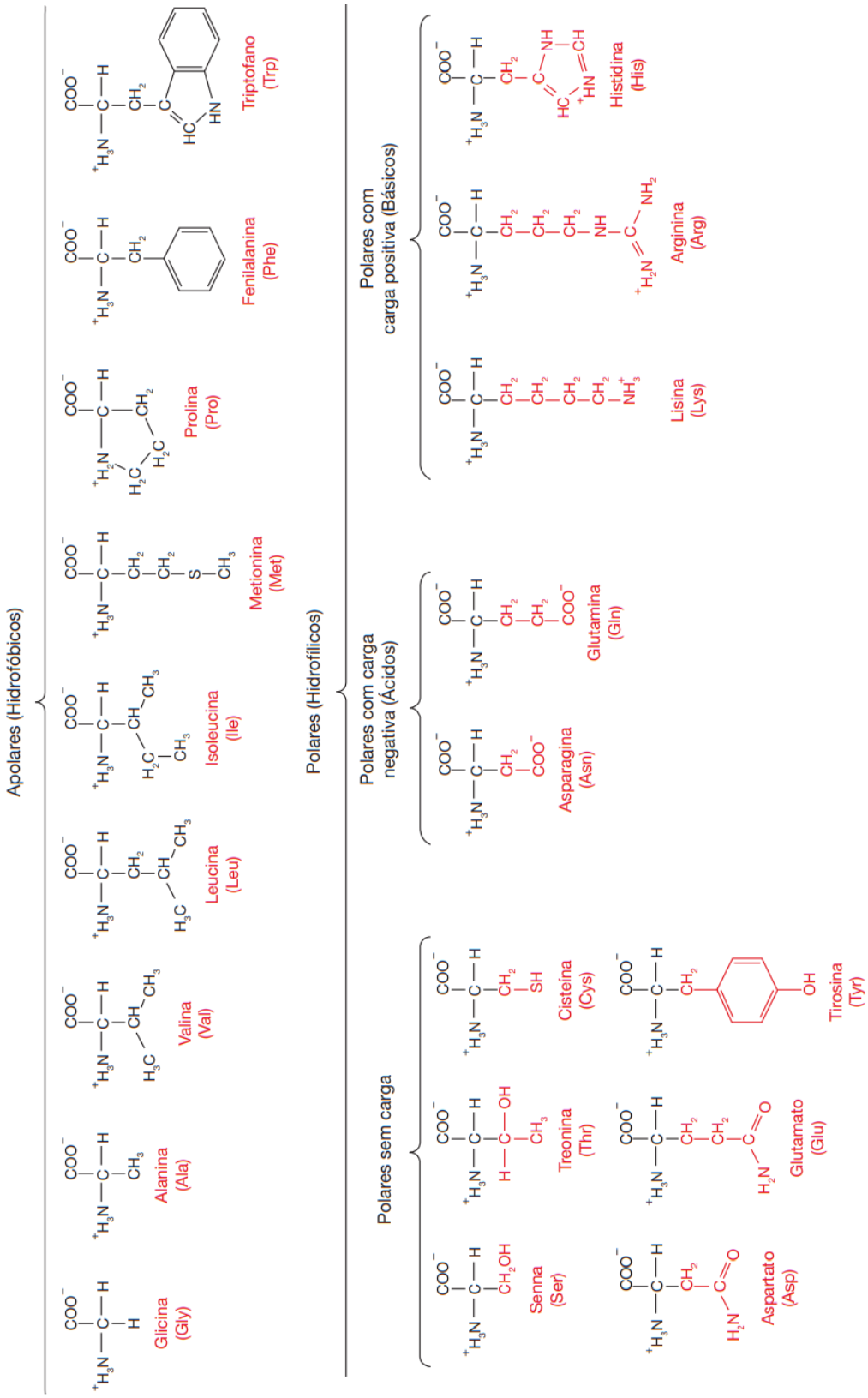
As moléculas formadoras das macromoléculas levam o nome de monômeros ou resíduos. Basicamente, existem quatro tipos de monômeros (ZAHA; FERREIRA; PASSAGLIA, 2014):

- Os nucleotídeos, formadores dos ácidos nucleicos.
- Os aminoácidos, formadores das proteínas.
- Os açúcares, formadores dos carboidratos que são responsáveis por diversas funções celulares e são também a principal fonte de energia celular.
- E finalmente, os ácidos Graxos, formadores dos lipídeos que diferente das substâncias anteriores, não são macromoléculas. Os ácidos graxos também constituem uma outra substância denominada fosfoacilgliceróis, que estão presentes na membrana celular.

As proteínas são macromoléculas formadas por uma longa cadeia não ramificada de aminoácidos. As proteínas também são chamadas de polipeptídios, devido ao fato

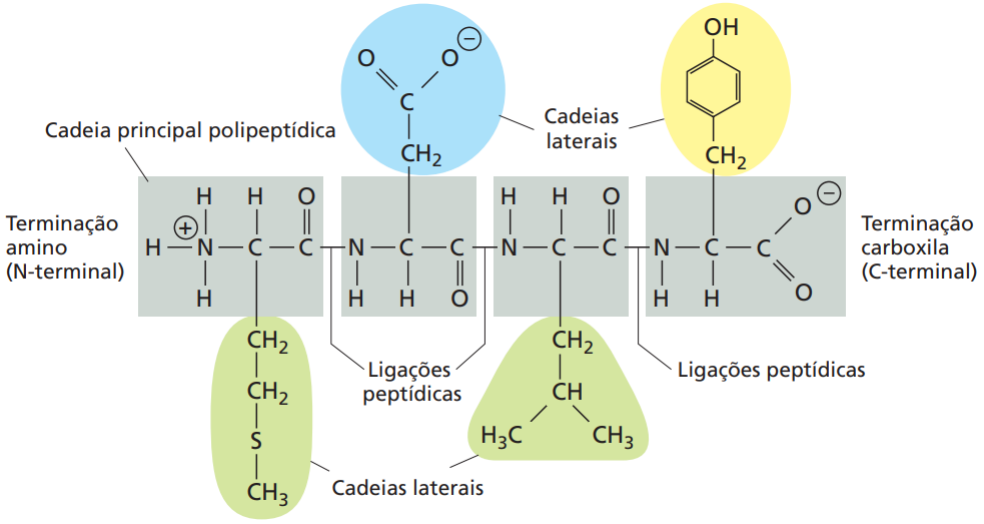
de que seus aminoácidos estão ligados uns aos outros por meio de ligações peptídicas covalentes (ALBERTS et al., 2010). Existem ao todo, vinte aminoácidos para formar todos e qualquer tipo de proteína, eles estão ligados à cadeia principal da proteína (cadeia central), mas não estão entre as ligações peptídicas e sim em cadeias laterais, conforme a Figura 2.4. As cadeias laterais podem ser apolares (hidrofólicos) ou polares (hidrofílicos), as polares podem possuir carga positiva, negativa ou nenhuma carga, conforme a Figura 2.3 (ALBERTS et al., 2010).

Figura 2.3: Estrutura química dos 20 aminoácidos formadores das proteínas.




Fonte: Adaptado de ZAHA; FERREIRA; PASSAGLIA (2014).

Figura 2.4: Os componentes de uma proteína.




Fonte: Adaptado de ALBERTS et al. (2010)

As proteínas podem ser diferenciadas em quatro níveis de organização estrutural, como pode ser observado na Figura 2.5 (ZAHA; FERREIRA; PASSAGLIA, 2014):

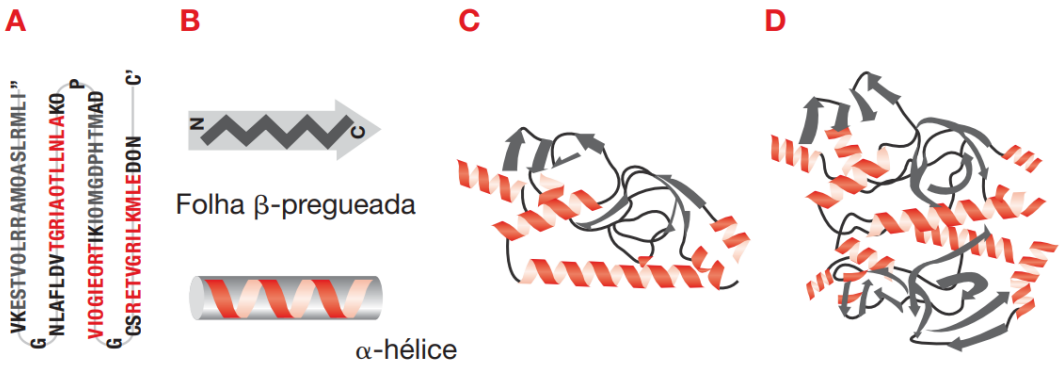
- Estrutura primária: Essa estrutura se refere a ordem em que os aminoácidos e são ligados para formar a cadeia peptídica. As ligações peptídicas e as pontes S-S, formadas entre os resíduos de cisteína, também estão localizadas nessa estrutura.

- Estrutura secundária: Este nível refere-se a organização espacial nos quais se encontram os aminoácidos próximos da cadeia central. Essa organização pode aparentar uma forma cilíndrica, onde a cadeia se enrola em torno de um eixo imaginário, ou como uma folha pregueada ou ainda, um formato globular.

- Estrutura terciária: Essa estrutura se refere a forma como a proteína está enovelada. Ela é estabelecida quando diferentes estruturas secundárias dispõem-se entre si, incluindo os das cadeias laterais e do grupo prostético.

- Estrutura quaternária: Algumas proteínas são multiméricas, isto é, possuem mais de um polipeptídeo, e portanto, apresentam um quarto nível estrutural.

Figura 2.5: Esquema representativo das estruturas das proteínas A) Estrutura primária. B) Estrutura secundária. C) Estrutura terciária e D) Estrutura quaternária.




Fonte: Adaptado de ZAHA; FERREIRA; PASSAGLIA (2014)

### 2.1.3 Ácidos nucleicos e nucleotídeos

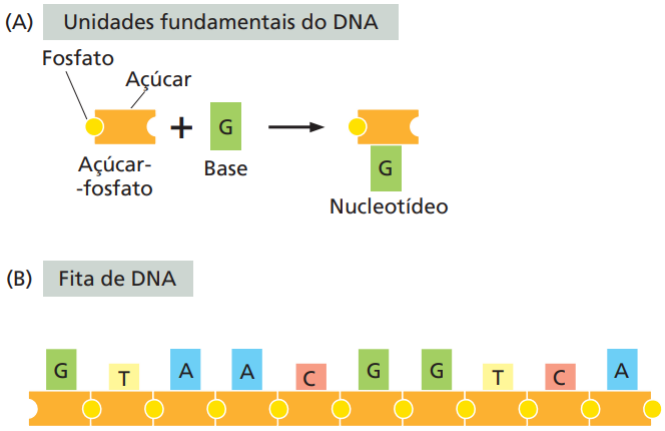
A proteína é resultado de uma expressão genética. Portanto, é o gene que determina a sequência de aminoácidos de cada proteína (ZAHA; FERREIRA; PASSAGLIA, 2014). O gene é constituído pelo ADN, que é o responsável pelo armazenamento de toda informação genética da célula. Um gene pode ser definido como um intervalo da sequência genética do ADN, delimitado por pontuações ou indicações presentes nessa mesma sequência (ALBERTS et al., 2010).

O ADN consiste de uma série de moléculas de açúcar-fosfato ligada à uma cadeia nitrogenada (denominada base), denominados nucleotídeos. As bases ligadas à essa cadeia são Adenina (representada pelo caractere A), Citosina (representada pelo caractere C), Timina (representada pelo caractere T) e Guanina (representada pelo caractere G), conforme Figura 2.6 (ALBERTS et al., 2010).

Segundo ALBERTS et al. (2010), uma molécula de ADN típica é formada por duas longas cadeias pareadas e não ramificadas, num formato de hélice (vide Figura 2.7). As bases dessa cadeia estão ligadas sempre seguindo uma regra rigorosa: A liga-se com T e C liga-se com G. As ligações entre as duas fitas, dada por um par de hidrogênios, são fracas do ponto de vista químico e isso faz com que, na separação

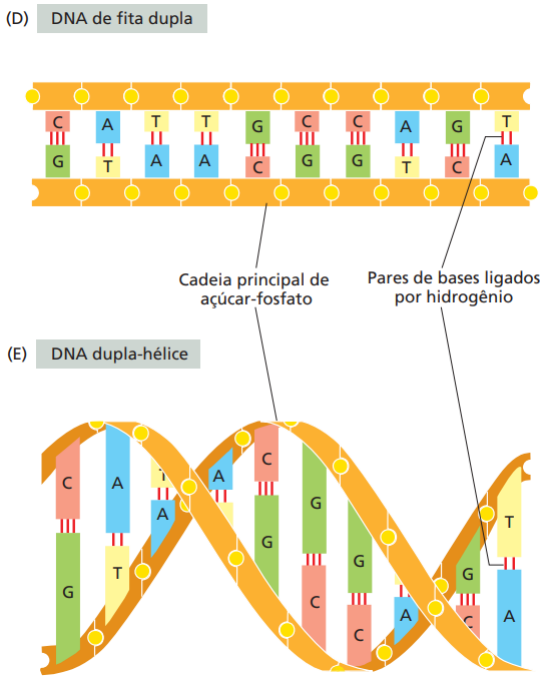
que ocorre durante o processo de replicação do ADN, a cadeia não seja danificada.

Figura 2.6: Composição do ADN.




Fonte: Adaptado de (ALBERTS et al., 2010)

Figura 2.7: Fita de ADN.



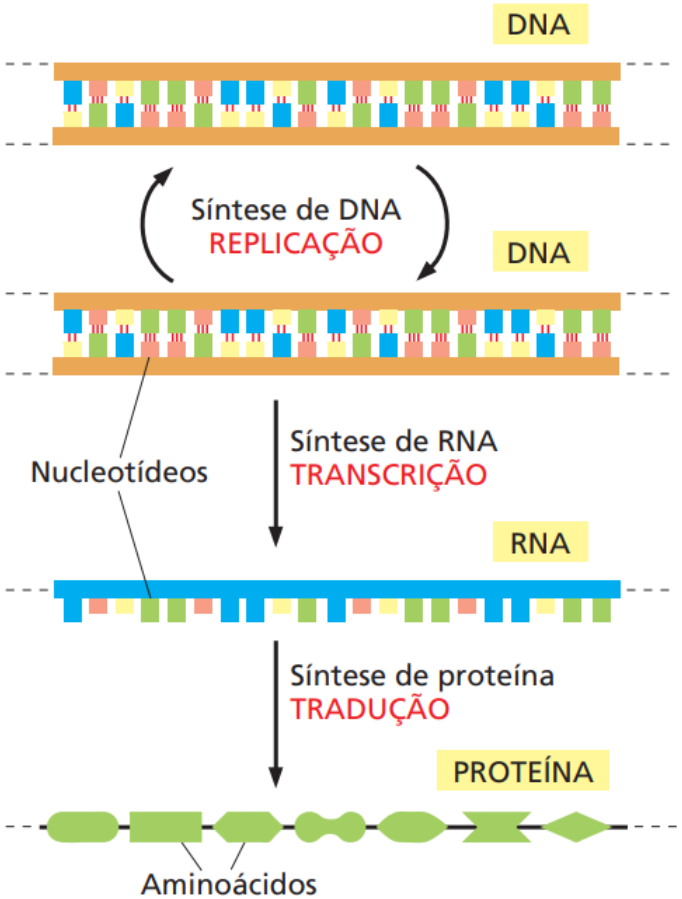

Fonte: Adaptado de (ALBERTS et al., 2010)

### 2.1.4 Síntese Proteica

Para ocorrer a síntese proteica, isto é, para que uma proteína seja construída é necessário que o gene correspondente dessa proteína se expresse (ALBERTS et al., 2010). Para que ocorra essa expressão gênica, são necessárias 2 etapas fundamentais: a transcrição e a tradução.

A síntese do ADN ou replicação, a transcrição e tradução, define um processo que é denominado o dogma central da biologia molecular, demostrado na Figura 2.8. Uma macromolécula importante nesse processo é o ácido ribonucleico, ou ARN. O ARN é o responsável por transportar as informações contidas no DNA para que ocorra a produção da molécula da proteína (ZAHA; FERREIRA; PASSAGLIA, 2014).

Figura 2.8: Do DNA à Proteína: O Dogma central da Biologia Molecular.




Fonte: (ALBERTS et al., 2010)

A molécula de ARN é muito similar à molécula do ADN, exceto que possui açúcares do tipo ribose ao invés de desoxirribose e que as bases de Timina são substituídas por bases de uracila. Nesse sentido, a informação contida no ADN, correspondente ao gene da proteína, é replicada fielmente para a molécula de ARN, com um alfabeto levemente diferente (ALBERTS et al., 2010). Por transportar as informações genéticas do ADN no processo da síntese proteica, o ARN leva o nome de ARN mensageiro, ou ARNm.

É importante esclarecer que o ARN também exerce outras funções dentro da célula e não serve apenas para a síntese proteica. De fato, o ARN desempenha diversas funções, como observa-se na Tabela 2.1.

Tabela 2.1: Tipos de ARNs e suas funções correspondentes. Adaptado de (ALBERTS et al., 2010)

| Tipo de ARN | Função |
|---|---|
| mRNAs | RNAs mensageiros; codificam proteínas. |
| rRNAs | RNAs ribossômicos; formam a estrutura básica do ribossomo e catalisam a síntese proteica. |
| tRNAs | RNAs transportadores; elementos essenciais para a síntese proteica, atuando como adaptadores entre o mRNA e os aminoácidos. |
| snRNAs | Pequenos RNAs nucleares; atuam em uma série de processos nucleares, incluindo o *splicing* do pré-mRNA. |
| snoRNAs | Pequenos RNAs nucleolares; ajudam a processar e modificar quimicamente os rRNAs. |
| miRNAs | Micro-RNAs; regulam a expressão gênica pelo bloqueio da tradução de mRNAs específicos e provocam a sua degradação. |
| siRNAs | Pequenos RNAs de interferência; desligam a expressão de genes pela degradação direta de mRNAs selecionados e pelo estabelecimento de estruturas de cromatina compacta. |
| piRNAs | RNAs que interagem com piwi; ligam-se a proteínas piwi e protegem a linhagem germinativa da ação de elementos transponíveis. |
| lncRNAs | RNAs não codificadores longos; muitos têm função de suporte estrutural; eles regulam diversos processos celulares, inclusive a inativação do cromossomo X. |

Após concluído o processo de transcrição do ADN, o ARN é movido do núcleo

para o citoplastma, onde o ribossomo, uma molécula complexa que é composta de ARNs e proteínas, efetua a segunda parte do processo que é a tradução. De modo geral, o processo de tradução consiste em traduzir as informações contidas no ARN para uma sequência de nucleotídeos. Um fato interessante é que existem apenas quatro tipos de nucleotídeos e vinte tipos de aminoácidos, portanto, uma tradução direta não é possível de ser feita sendo necessário um código genético para efetuar essa conversão (ALBERTS et al., 2010).

Esse código pode ser visto como um mapa ou um dicionário que relaciona um códon, que é uma trinca de nucleotídeos, para um aminoácido. Como pode ser observado, se existem quatro nucleotídeos e eles são lidos em trincas para codificar os aminoácidos e existem apenas vinte aminoácidos, então quarenta e quatro combinações são redundantes. De fato, alguns aminoácidos são codificados por diferentes combinações, como pode ser observado na Figura 2.9.

Figura 2.9: Código genético: Mapeamento entre códons e aminoácidos.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AGA | | | | | | | | | UUA | | | | | AGC | | | | |
| | AGG | | | | | | | | | UUG | | | | | AGU | | | | |
| GCA | CGA | | | | | | GGA | | | CUA | | | | CCA | UCA | ACA | | GUA | |
| GCC | CGC | | | | | | GGC | | AUA | CUC | | | | CCC | UCC | ACC | | GUC | UAA |
| GCG | CGG | GAC | AAC | UGC | GAA | CAA | GGG | CAC | AUC | CUG | AAA | | UUC | CCG | UCG | ACG | | GUG | UAG |
| GCU | CGU | GAU | AAU | UGU | GAG | CAG | GGU | CAU | AUU | CUU | AAG | AUG | UUU | CCU | UCU | ACU | UGG | GUU | UGA |
| Ala | Arg | Asp | Asn | Cys | Glu | Gln | Gly | His | Ile | Leu | Lys | Met | Phe | Pro | Ser | Thr | Trp | Tyr | Val | Término |
| A | R | D | N | C | E | Q | G | H | I | L | K | M | F | P | S | T | W | Y | V | |

Fonte: (ALBERTS et al., 2010)

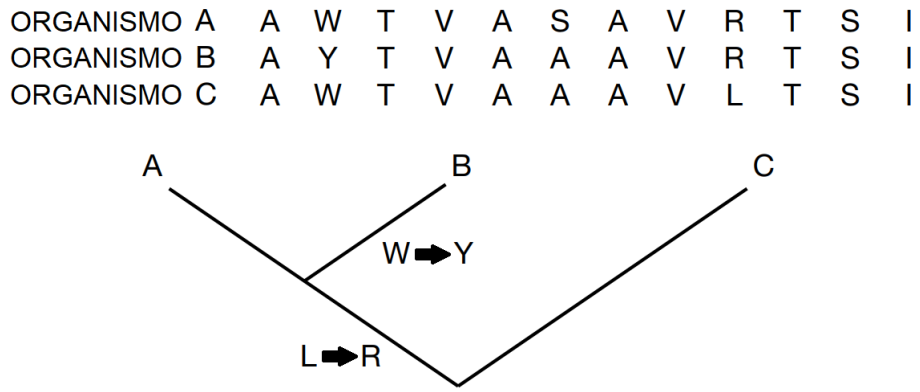
### 2.1.5 Filogenia

A filogenia é o estudo da evolução das espécies e seu objetivo é construir uma árvore filogenética, dado que parte-se da premissa que as espécies derivam de um ancestral comum. As primeiras abordagens desse estudo levavam em consideração os fenótipos dos indivíduos, mas como existem indivíduos que derivam de ancestrais diferentes e compartilham características fenotípicas, as classificações poderia ficar incorretas. Nesse sentido, uma melhor abordagem é basear-se no genótipos dos indi-

víduos (ZAFALON, 2009).

A abordagem de classificação por análise de genótipos envolve efetuar um alinhamento de múltiplas sequências e construir uma árvore filogenética por meio de análises das sequências alinhadas. Essa abordagem, além de computacionalmente intensiva, requer uma modelagem probabilística e nem sempre resultam em uma árvore que representa a evolução real, isso pode ocorrer por exemplo, quando as sequências alinhadas não são ortólogas. Sequências ortólogas são aquelas que descendem de um ancestral comum e que mantiveram a função proteica. Na Figura 2.10 observa-se esse método para construção da árvore filogenética.

Figura 2.10: Método de construção da arvore filogenética.




Fonte: Adaptado de (MOUNT, 2004)

## 2.2 Alinhamento de Sequências

A tarefa de alinhar sequências é muito importante no contexto da bioinformática e é utilizada por biólogos para diversas finalidades (SIMOSSIS; KLEINJUNG; HERINGA, 2003), como por exemplo:

- Inspeções visuais, para procurar pistas que possam revelar alguma significância biológica.
- Modelagem por homologia, para encontrar o modelo tridimensional da proteína.

- Reconstrução filogenética, para formação da árvore filogenética, contemplada na seção 2.1.5.

Uma sequência biológica é representada por uma cadeia de caracteres, cujo alfabeto é diferente para as cadeias de proteínas e as cadeias de aminoácidos. As sequências de proteínas possuem o alfabeto $\alpha$ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}, onde cada letra corresponde a um aminoácido. As sequências de nucleotídeos possuem o alfabeto $\alpha$' = {A, T, C, G}, onde cada letra corresponde a um nucleotídeo (ROSEN, 2017). A relação entre os caracteres e as moléculas pequenas correspondentes estão demonstradas nos Tabelas 2.3 e 2.2.

Tabela 2.2: Correspondência entre aminoácidos, sigla e sua representação em caracteres.

| Aminoácido | Abreviação | Caractere |
|---|---|---|
| Glicina ou Glicocola | Gly, Gli | G |
| Alanina | Ala | A |
| Leucina | Leu | L |
| Valina | Val | V |
| Isoleucina | Ile | I |
| Prolina | Pro | P |
| Fenilalanina | Phe ou Fen | F |
| Serina | Ser | S |
| Treonina | Thr, The | T |
| Cisteina | Cys, Cis | C |
| Tirosina | Tyr, Tir | Y |
| Asparagina | Asn | N |
| Glutamina | Gln | Q |
| Aspartato ou Ácido aspártico | Asp | D |
| Glutamato ou Ácido glutâmico | Glu | E |
| Arginina | Arg | R |
| Lisina | Lys, Lis | K |
| Histidina | His | H |
| Triptofano | Trp, Tri | W |
| Metionina | Met | M |

Tabela 2.3: Correspondência entre nucleotídeos e sua representação em caracteres.

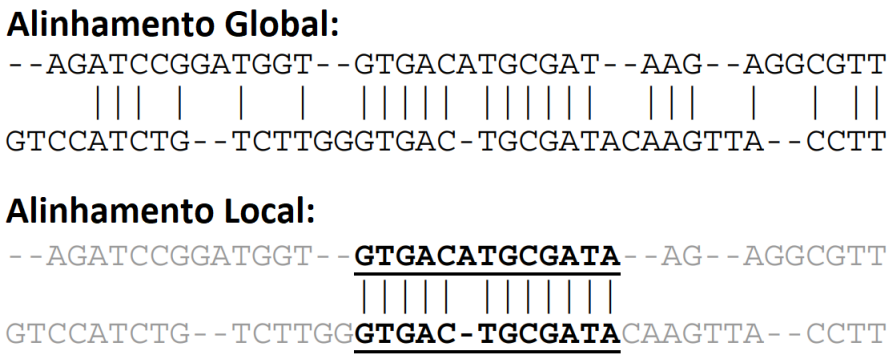
| Nucleotídeo | Caractere |
|---|---|
| Adenina | A |
| Timina | T |
| Citosina | C |
| Guanina | G |

De modo geral, o alinhamento de sequências consiste no posicionamento e comparação entre duas ou mais cadeias, possibilitando que inferências possam ser feitas a partir de suas regiões similares (ZAFALON, 2009). As técnicas de alinhamento de sequência podem ser divididas, basicamente, entre alinhamento par-a-par e alinhamento múltiplo (MOUNT, 2004). Além dessa divisão, existe um diferença importante entre alinhamento de aminoácidos, componentes da proteína e alinhamento de nucleotídeos, componentes do ADN (COHEN, 2004).

### 2.2.1 Alinhamento par-a-par

O alinhamento par-a-par, ocorre sempre entre duas sequências e pode ser feito de modo local ou global, dependendo da necessidade. Nas buscas de sequências em bases de dados, por exemplo, é frequente o uso de alinhamentos locais, pois é desejado encontrar algum fragmento da sequência alvo que possua uma região que se alinhe à mesma região de uma sequência de entrada. No caso de alinhamento global, procura-se efetuar o alinhamento de toda sequência e não simplesmente de uma região específica. Essa diferença está representada na Figura 2.11.

Figura 2.11: Alinhamento local x Alinhamento Global.




Fonte: Adaptado de (COULL et al., 2003)

Para efetuar um alinhamento par-a-par é necessário, primeiramente, definir uma função de pontuação. Essa função de pontuação irá penalizar o resultado final do alinhamento quando ocorrer *mismatches* ou descasamentos, e bonificar quando ocorrer *matches* ou casamentos. Os *matches* ou *mismatches* do alinhamento ocorrem entre os caracteres das duas sequência numa mesma posição.

Na prática, as sequências podem ser de tamanhos diferentes, por esse motivo, os *gaps* são inseridos entre os caracteres em regiões especificas para que elas fiquem do mesmo tamanho, como pode ser observado na Figura 2.11. A adição dos espaços também são levadas em consideração pela função de pontuação e em muitas vezes, para penalização.

Um exemplo de função de pontuação em que é atribuído (+2) para cada *match* e (-2) para *mismatch*, aplicada a um par de s e s', onde, s = {G,A,C,G,C,A,T,T,A,G} e s' = {G,A,T,C,G,G,A,A,T,A,G}, alinhadas da forma demonstrada no Tabela 2.4, resultaria em 14 pontos, pois:

$$9(+2)+2(-2)=14$$

Sendo que ocorreram 9 *matches* e 2 *mismatches*, 1 *mismatch* na terceira posição, entre o caractere T e um espaço e outro *mismatch* na oitava posição, entre os caracteres

T e A.

Tabela 2.4: Exemplo de alinhamento par-a-par. Adaptado de ZAFALON (2009)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| G | A | - | C | G | C | A | T | T | A | G |
| G | A | T | C | G | G | A | A | T | A | G |

O alinhamento entre um par de sequências é geralmente baseado no algoritmo de Programação Dinâmica. Com essa técnica, busca-se sempre o melhor alinhamento entre as duas cadeias (ZAFALON, 2009).

No algoritmo de programação dinâmica utiliza-se da função de pontuação para preencher uma matriz, na qual cada célula dessa matriz contém um valor correspondente à comparação entre um caractere da sequência $s$ e outra da sequência $s'$. Após o cálculo de toda a sequência, efetua-se no algoritmo um processo de *backtracking* para realizar o alinhamento ótimo.

Para alinhamento global, o algoritmo mais indicado é o de Needleman-Wunsh (NEEDLEMAN; WUNSCH, 1970), para o caso de alinhamento local, o de Smith-Waterman (SMITH; WATERMAN et al., 1981), pois ambos os algoritmos efetuam um alinhamento ótimo.

### 2.2.2 Alinhamento de Aminoácidos

Ao efetuar alinhamento de sequências de aminoácidos, cuidados extras devem ser tomados com relação à função de pontuação (COHEN, 2004). Um dos motivos desse cuidado é devido as mutações que ocorrem durante o processo de evolução, fazendo com que alguns aminoácidos sejam substituídos por outros, sem causar prejuízos à função da proteína (ALBERTS et al., 2010).

Para mitigar o problema de alinhamentos de aminoácidos que estão mais distantes da realidade biológica, criou-se algumas matrizes de dimensão 20x20 com base em substituições de aminoácidos para orientar as funções de pontuação. Nesse sentido, um peso é atribuído em cada célula dessa matriz para pontuar a substituição de um

aminoácido por outro ou por um espaço. Desse modo, as funções de pontuação podem levar em consideração essa matriz e computar um peso mais adequado em situações de *mismatches*.

As matrizes mais utilizadas são da família BLOSUM (*Blocks of Amino Acid Substitution Matrix*) e PAM (*Percent Accepted Mutation*) (ZAFALON, 2009). Dentro de uma mesma família, as matrizes podem ser diferenciadas por um sufixo numérico, como PAM250 ou BLOSUM62 e quanto maior o valor do sufixo mais leniente são os valores de substituição (COHEN, 2004). Essa leniência pode ser analogamente comparada com a pontuação de *matches* e *mistaches* dos aminoácidos. Uma pontuação mais leniente seria (+1) para *matches* e (0) para *mismatches*, enquanto um pontuação menos *leniente*, seria de (-1) para *mismatches* (COHEN, 2004).

## 2.3 Alinhamento Múltiplo de Sequências

O Alinhamento múltiplo de sequências, ou AMS, pode ser visto como uma matriz bidimensional, onde as linhas são as sequências e as colunas são os aminoácidos que estão alinhados em posições apropriadas entre arranjos de espaços (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

Para efetuar um AMS e extrair o máximo de informações desse alinhamento, são necessárias três etapas:

- Seleção de Sequências;
- Escolha de uma função de pontuação;
- Aplicação da função de pontuação na compilação do alinhamento.

A seleção de sequências é uma etapa importante no processo de alinhamento múltiplo. Se sequências que não possuem relação biológicas forem selecionadas, o resultado do alinhamento também não terá nenhum significado biológico. Portanto, o ideal

é que as sequências escolhidas sejam ortólogas. Nessa fase, técnicas de busca por homologia em bases de dados são aplicadas, para pontuar as sequências selecionadas e evitar a seleção de sequências sem relevância (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

A escolha da função de pontuação também é importante, mas essa escolha acompanha um problema complexo de ser resolvido. O ideal é que a função de pontuação seja formulada levando-se em consideração todo conhecimento biológico, contendo os aspectos evolucionários, estruturais e funcionais das proteínas envolvidas. Na prática, entretanto, essas informações não estão presentes ou não são possíveis modelá-las matematicamente e portanto recorre-se às matrizes de substituição de aminoácidos apresentadas na seção 2.2.2 (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

A última etapa é a aplicação da função de pontuação para compilar o alinhamento múltiplo de sequências. Diversas classes de algoritmos existem para efetuar essa compilação, as duas mais utilizadas são: alinhamento múltiplo progressivo e alinhamento múltiplo iterativo (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

### 2.3.1 Alinhamento Progressivo

O alinhamento progressivo é dividido em três fases: alinhamento inicial das sequências de modo par-a-par, construção de uma árvore guia e finalmente, o alinhamento das sequências em si. Na primeira fase, um alinhamento de todas as sequências, de modo par-a-par, é efetuado para preencher uma matriz de distâncias, essas distâncias são utilizadas no segundo passo para a construção da árvore guia.

A partir da matriz de distâncias, uma árvore é construída por meio de um algoritmo de agrupamento hierárquico, essa árvore é chamada árvore guia. O resultado do agrupamento gerará portanto, um dendrograma, que será processado em ordem, das folhas até a raiz para efetuar o alinhamento em si. Nessa etapa, vários algoritmos de agrupamento podem ser utilizados e o mais comum é o algoritmo de *Neighbor-Joining*

(SAITOU; NEI, 1987).

Uma característica desse algoritmo é que ele utiliza uma estratégia gulosa para o alinhamento, pois uma vez que uma sequência é alinhada e espaços são inseridos nela, esses espaços serão mantidos até o final da execução. Em muitas situações essa característica pode ser um problema, pois algumas sequências podem conter melhores alinhamentos óbvios, se fossem analisadas visualmente, mas por terem sido alinhadas prematuramente, o alinhamento nunca é corrigido (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

Para evitar esse problema, uma outra classe de algoritmos pode ser utilizada, que é a estratégia iterativa. Essa estratégia leva em consideração informações de uma iteração anterior para a próxima, permitindo que os alinhamentos já efetuados possam ser revisados. Essa abordagem não segue uma regra formal, como a técnica de alinhamento progressivo, ficando livre para utilização de qualquer meta-heurística que possa ser aplicada (SIMOSSIS; KLEINJUNG; HERINGA, 2003).

Diversos métodos são aplicados seguindo a estratégia de alinhamento iterativo, como por exemplo: Algoritmos Genéticos (NOTREDAME; HIGGINS, 1996) (NOTREDAME; O'BRIEN; HIGGINS, 1997) (LEE et al., 2008), Recozimento Simulado (SCHWARTZ; PACHTER, 2007) (ISHIKAWA et al., 1993) (KIM; PRAMANIK; CHUNG, 1994), Colônia Artificial de Abelhas (RUBIO-LARGO; VEGA-RODRÍGUEZ; GONZÁLEZ-ÁLVAREZ, 2016), Busca Tabu (RIAZ; WANG; LI, 2004), dentre outros.

### 2.3.2 Recozimento Simulado

A meta-heurística de recozimento simulado (KIRKPATRICK; GELLATT; VECCHI, 1982) é inspirada no processo metalúrgico de endurecimento de metais. Basicamente, ela consiste em definir um espaço de busca $X = \{x_1 ... x_n\}$ e uma função custo $f(x) \Rightarrow \mathbb{R}$ e por meio de regras de decaimento de uma temperatura $\mathcal{T}$ atingir um estado

de equilíbrio ou ponto de mínimo local.

A cada iteração, é calculada a probabilidade de um novo estado $X_{novo}$ ser aceito e substituir o estado atual $X_{otimo}$. O cálculo da probabilidade permite, em algumas situações, que estados com maior custo substituam estados com menor custo durante o inicio do algoritmo, para explorar mais o espaço de busca, mas ao decorrer das iterações, essa probabilidade se torna cada vez menor.

Em bioinformática, essa técnica foi empregada no problema de alinhamento múltiplo de sequências (KIM; PRAMANIK; CHUNG, 1994), a paralelização do algoritmo também foi explorada (ISHIKAWA et al., 1993). YAO et al. 2015 também emprega essa abordagem para efetuar alinhamentos baseados em uma sequência consenso previamente definida.

### 2.3.3 Algoritmos Genéticos

A técnica de algoritmos genéticos (SCHWEFEL; RUDOLPH, 1995) também é aplicada no alinhamento múltiplo de sequências. Simplificadamente, ela define uma população de indivíduos que representam as solução possíveis aos quais são aplicados alguns operadores, como seleção, mutação e recombinação e por meio de uma função objetivo, os indivíduos são comparados e eliminados ou promovidos para uma próxima geração, visando encontrar uma solução ótima.

Quando aplicado à alinhamento múltiplo de sequências, os indivíduos representam os alinhamentos e a função objetivo leva em consideração a similaridade entre as sequências envolvidas. Os indivíduos são manipulados pelos operadores a cada iteração e a função objetivo é aferida. Esse ciclo é repetido por um número parametrizado de vezes ou até atingir alguma condição de parada.

As ferramentas SAGA (NOTREDAME; HIGGINS, 1996), RAGA (NOTREDAME; O'BRIEN; HIGGINS, 1997) e MSA-GA (GONDRO; KINGHORN, 2007) implementam a técnica de algoritmos genéticos. Segundo GONDRO; KINGHORN (2007) a

ferramenta SAGA apresenta resultados melhores do que a ferramenta CLUSTALW.

Uma característica dos algoritmos genéticos, conforme ressaltado por LEE et al. (2008) é que devido à sua natureza, é comum que as soluções convirjam para um ponto de máximo ou mínimo local e por esse motivo, as pesquisas voltam-se para o refinamento dos operadores.

As evoluções dos operadores geralmente são aplicadas diretamente nas ferramentas, AMORIM et al. implementou a função objetivo COFFEE *(Consistency based Objective Function For alignmEnt Evaluation)* na ferramenta MSA-GA e obteve melhores resultados para conjuntos de sequências pouco similares. ZHU; HE; JIA (ZHU; HE; JIA) propuseram alterações no algoritmo genético no processo de geração de novas populações e introduziram um novo operador, além disso, modificaram a função objetivo para se tornar uma função multiobjetivo baseada em decomposição e obtiveram alinhamentos de boa qualidade, superando outras ferramentas. AMORIM et al. (2016) implementaram uma versão paralela do algoritmo, visando reduzir o tempo de execução.

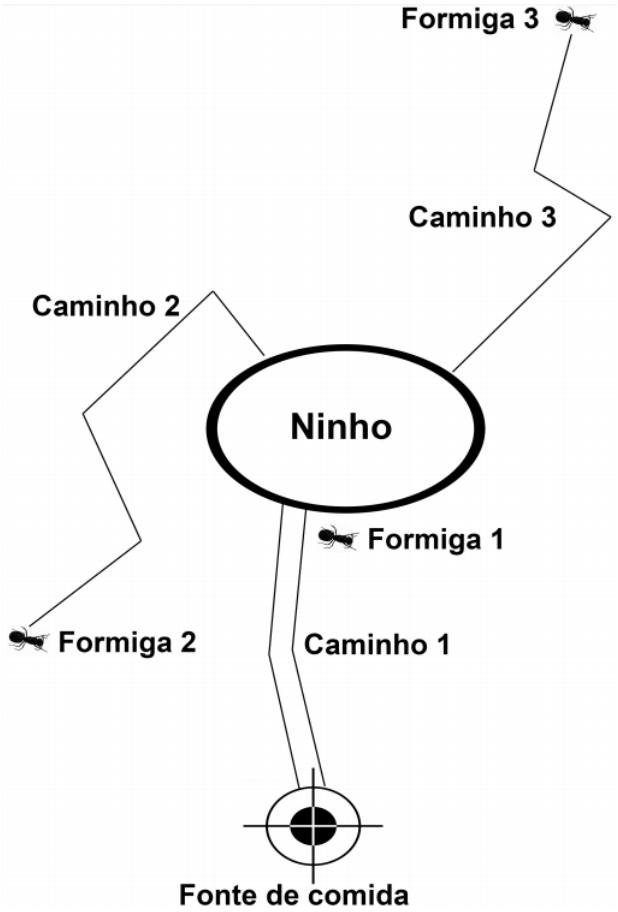
### 2.3.4 Otimização por Colônia de Formigas

O Algoritmo de otimização por colônia de formigas (DORIGO; GAMBARDELLA, 1997), ou OCF, é um algoritmo bioinspirado no comportamento das formigas. Observa-se que as formigas, que são praticamente cegas, procuram sempre o caminho mais curto entre a fonte de alimentos e o ninho. A comunicação entre elas ocorre por meio de uma trilha de feromônio depositado pelo caminho que elas percorrem, de modo que, conforme mais formigas percorrem uma trilha, maior o depósito de feromônio nessa trilha. Dessa mesma forma, em trilhas com distâncias menores, a chance do feromônio evaporar em decorrência do tempo também é menor do que em trilhas mais longas.

No cenário observado pela Figura 2.12 existem três formigas, que inicialmente

procuram por fontes de alimento aleatoriamente e cada uma delas deixa um rastro de feromônio pelo caminho. Quando a formiga 1, que encontrou uma fonte de alimentos, voltar ao ninho e passar pelo mesmo caminho, ela deixará mais feromônio e portanto, quando um segundo grupo de formigas deixar o ninho para procurar alimentos o caminho da formiga 1 terá maior chance de ser escolhido (ZAFALON, 2009).

Figura 2.12: Cenário de busca de comida em uma colônia de formigas.




Fonte: Adaptado de (AMORIM, 2017)

A meta-heurística de otimização por colônia de formigas pode ser aplicada em bioinformática de diversas maneiras. Para acelerar a busca por *motiffs*, que são regiões biologicamente relevantes nas sequências (MOSS; JOHNSON, 2003) . LEE et al. (2008) utilizaram para efetuar o alinhamento das sequências em si. AMORIM (2017) utilizou uma abordagem híbrida baseada em colônia de formigas e aplicou na

ferramenta MSA-GA, para refinar o resultado produzido pela ferramenta.

## 2.4 Ferramentas para AMS

### 2.4.1 Clustal Ômega

A ferramenta Clustal Ômega (SIEVERS; HIGGINS, 2014) é a mais recente ferramenta lançada na família Clustal. Essa versão foi criada principalmente para permitir o processamento de grandes quantidades de sequências, o que nas versões anteriores não era possível sem prejudicar o resultado dos alinhamentos (SIEVERS et al., 2011).

A Clustal Ômega baseia-se na estratégia progressiva e para construir a árvore guia, utiliza o algoritmo *mBed*. As versões anteriores não utilizavam esse algoritmo e essa foi a principal diferença para permitir o processamento de grandes volumes de dados, visto que a etapa da construção da árvore filogenética é a etapa mais custosa do processo. Ao final do processo, a ferramenta também efetua um refinamento do alinhamento produzido, para melhorar a significância biológica dos resultados, esse processo utiliza o algoritmo *HHalign*, baseado em modelos ocultos de Markov (SIEVERS et al., 2011).

### 2.4.2 DIALIGN

A ferramenta DIALIGN pode ser utilizada tanto para alinhamento de sequências de ADN quanto de sequências de proteínas. Além disso, ela possibilita efetuar alinhamentos par-a-par e alinhamentos múltiplos (MORGENSTERN et al., 1998).

Ao invés de inserir *gaps* e efetuar o alinhamento utilizando o algoritmo de programação dinâmica, a ferramenta utiliza um método de diagonais, que produz bons resultados para alinhamentos locais. Segundo MORGENSTERN et al. (1998) o algoritmo de Smith-Waterman apresenta problemas quando as regiões semelhantes estão intercaladas com regiões não relacionadas e sua proposta produz resultados melhores

para essas situações.

SCHMOLLINGER et al. (2004) propuseram uma versão paralelizada do algorítmo que melhorou seu desempenho. Além disso, SUBRAMANIAN; KAUFMANN; MORGENSTERN (2008) agregaram a estratégia de alinhamento progressivo junto às estratégias anteriores o que aumentou a qualidade dos alinhamentos gerados.

### 2.4.3 MAFFT

A ferramenta MAFFT baseia-se na Transformada Rápida de Fourier (TRF) para efetuar o alinhamento múltiplo de ADN e proteínas. A ferramenta possui duas características que se destacam das demais: a utilização de TRF para detecção de regiões homólogas e uma abordagem eficiente de pontuação das sequências para o alinhamento múltiplo (KATOH et al., 2002).

Uma grande vantagem da MAFFT é que seu tempo de processamento supera consideravelmente outras ferramentas e além disso produz um alinhamento resultante de alta qualidade do ponto de vista biológico. Por outro lado, quando as sequências selecionadas são pouco relacionadas ou nelas existem grandes trechos com inserção, o resultado é degradado (KATOH; STANDLEY, 2013).

Para atender as necessidades de processamento de maiores volumes de dados (KATOH; TOH, 2010) modificaram a ferramenta para explorar o conceito de *multi-threading*. Além disso, o algoritmo não apresentava resultados esperados quando o número de sequências aproximava de 50.000 sequências, por isso, NAKAMURA et al. (2018) propuseram outro método para processar com eficiência esse volume de dados. Nesse método, há um requisito de maior capacidade dos discos, pois as informações são armazenadas em arquivos temporários e as operações de entrada e saída podem gerar perdas de desempenho casos as sequências possuam sejam pequenas, com aproximadamente vinte e três aminoácidos.

### 2.4.4 SAGA

A ferramenta SAGA utiliza-se do método iterativo baseado em algoritmos genéticos. Além disso, o algoritmo de programação dinâmica também é utilizando, o que classifica a ferramenta como uma ferramenta híbrida.

A função objetivo do algoritmo genético aplicada na ferramenta é a soma dos pesos dos pares e a função *fitness* é a qualidade do alinhamento. Inicialmente, uma população aleatória de alinhamentos é gerada, essa geração é chamada *Generation Zero*. Os melhores alinhamentos são utilizados para gerar descendentes, esses alinhamentos são gerados aplicando operadores de *cross-over* dos indivíduos ancestrais ou mutação de um único indivíduo. Como critério de parada, utiliza-se a estabilização do aumento da aptidão (SIMOSSIS; KLEINJUNG; HERINGA, 2003), (NOTREDAME; HIGGINS, 1996).

A ferramenta possui vinte de dois operadores de mutação e recombinação genética, além de uma escalonador para selecionar quais operadores serão utilizados. Entretanto, THOMSEN; BOOMSMA (2004) demonstraram que essa grande variedade de operadores e escalonador não resulta em melhores resultados.

## 2.5 *Chaotic Jump*

Caos é um fenômeno não linear que ocorre em sistemas não-lineares. De modo geral, um sistema deve possuir três características para que ocorra o caos:

- O sistema é determinístico: Segue uma regra, uma função, equação ou uma lei que pode determinar ou especificar o resultado. Embora muitos sistemas podem aparentar aleatoriedade, um estudo mais aprofundado desses sistemas pode revelar que são de fato, caóticos e não aleatórios (WILLIAMS, 1997).

- Não linear: Significa que a saída desse sistema não é proporcional à entrada e que a mudança em uma variável não produz uma mudança proporcional em

outras variáveis relacionadas (WILLIAMS, 1997).

- Dinâmico: Um sistema dinâmico é um sistema que se move ou que evolui ao longo do tempo. Os sistemas dinâmicos se enquadram em duas categorias: conservativos, isto é, não ocorre fricção e portanto, não perde energia ao longo do tempo ou, dissipativos, quando um sistema perde energia ao longo do tempo e portanto se aproxima de uma condição assintótica ou limitante (WILLIAMS, 1997).

O estado de um sistema caótico é denominado espaço-fase (*phase space*) ou espaço-estado (*space state*), esse espaço é uma abstração matemática em que as coordenadas são utilizadas para demonstrar a fase ou o estado do sistema (WILLIAMS, 1997). Um sistema com apenas duas variáveis pode ser representado por um plano cartesiano, por exemplo. Ao analisar graficamente o plano cartesiano de um sistema caótico, não é incomum encontrar estados como observado pela Figura 2.13.

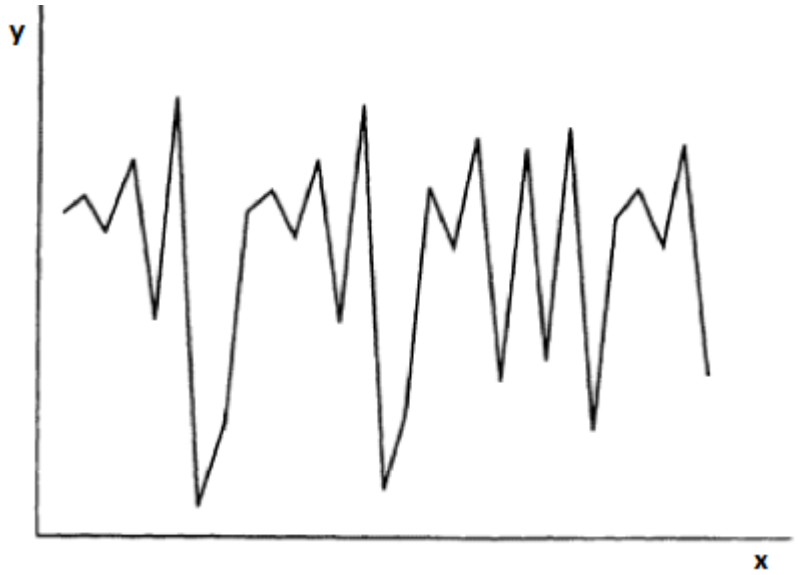
Além das características já citadas, observa-se que o sistemas caóticos são altamente sensíveis às condições iniciais e que pequenas modificações nas condições iniciais resultam em saídas muito diferentes (KAUR; ARORA, 2018). Essa sensibilidade pode ser demonstrada por meio de uma simples equação logística, como por exemplo, a equação 2.1:

$$x_{n+1} = \alpha x_n(1 - x_n) \tag{2.1}$$

A aplicação de *chaotic jump*, pode ser simplificada pela aplicação de uma função de mapa, como a equação logística demonstrada, para orientar a seleção de parâmetros ou conduzir a busca através do espaço. Nesse sentido, ao invés de o algoritmo convergir linearmente, o seu comportamento evolutivo será similar a saltos pelo espaço de busca. Esse comportamento, de forma controlada (KAUR; ARORA, 2018), evita o problema de fixar-se em mínimos ou máximos locais e apresentam bons resul-

tados, principalmente em algoritmos coletivos, como pode ser observado na literatura (ALATAS, 2010), (XU; DUAN; LIU, 2010), (ZAWBAA; EMARY; GROSAN, 2016), (KAUR; ARORA, 2018). As funções de mapa possuem as três características de um estado caótico, elas são determinísticas, pois para uma mesma entrada, sempre resultará na mesma saída, não são lineares, pois conforme o valor de entrada aumenta, a saída não é linear à essa entrada e são dinâmicas, pois o resultado não entra em um estado de exaustão, os valores sempre são modificados ao longo do intervalo caótico.

Figura 2.13: Plano cartesiano de um sistema caótico.




Fonte: Adaptado de (WILLIAMS, 1997)

## 2.6 Trabalhos Correlatos

Nas subseções 2.6.1, 2.6.2, 2.6.3 e 2.6.4 descrevem-se métodos que estão relacionados ao trabalho atual, desde abordagens feitas anteriormente e que podem ser melhoradas até aplicações em outras áreas da bioinformática que elucidam as vantagens do método proposto.

### 2.6.1 *An ant colony algorithm for multiple sequence alignment in bioinformatics*

O trabalho de (MOSS; JOHNSON, 2003) produz um sistema chamado *AntAlign* em que a otimização por colônia de formigas é aplicada no problema de alinhamento múltiplo de sequências. A ideia principal desse sistema é distribuir subsequências para formigas e fazê-las mover-se em um intervalo associado para cada sequência, permitindo que a trilha de feromônios se torne mais forte quando ocorrer um *match* entre a posição atual da formiga nas outras sequências.

O sistema possui uma classe gerenciadora da trilha, denominada *ITrail*, essa classe é responsável pelo caminho pelo qual as formigas irão percorrer. Desse modo, ela controla o depósito de feromônio das formigas e o intervalo no qual os feromônios serão depositados. A classe define três funções:

- A primeira função é responsável por selecionar uma trilha para a formiga quando uma nova instância é construída, essa função leva em consideração a subsequência gerada para a nova formiga e os caminhos já conhecidos.
- A segunda função é responsável por adicionar ou atualizar a contagem de feromônio nas trilhas.
- A terceira função é responsável por aplicar a evaporação do feromônio ao final de cada ciclo.

Durante o progresso do algoritmo, uma representação da sequência consenso é construída, essa representação, denominada *PTrail*, norteia a evolução desse método de alinhamento múltiplo das sequências. Uma sequência consenso, em alinhamento múltiplo de sequências, é formada pelos resíduos que mais possuem correspondência em todas as sequências em cada posição. Durante o processamento, a sequência consenso é utilizada pelas formigas para comparar com a subsequência que a formiga possui e determinar se o caminho que ela está percorrendo levará a uma solução melhor.

Ao final do ciclo de cada formiga, após a obtenção de um score para o alinhamento ocorrido entre a subsequência consenso e a trilha selecionada pela formiga, ambas as trilhas de feromônios da sequência consenso e da própria formiga são atualizadas para permitir a evolução até uma solução global. No Algoritmo 1 exibe-se o pseudo-código desse algoritmo e a Tabela 2.5 apresentam-se os resultados dos alinhamentos produzidos:

**Algorithm 1** Pseudocódigo do algoritmo Ant Colony Optimization aplicado à Alinhamento Múltiplo de sequências. Adaptado de (MOSS; JOHNSON, 2003)

```
1:  while Iterações não atingir limite do
2:      cargoLength =startingLength + (cycleNumber∗rateOfLengthIncrease)
3:      for sequencia = 1,2,...,N do
4:          Gerar o número requerido de formigas para a sequência atual
5:          for ant = 1,2,...,N do
6:              Encontra matches entre formiga atual e Trilha Consenso (PTrails)
7:              Determina probabilidades
8:              Determina o caminho da trilha
9:              Gera o alignmentScore para o caminho escolhido
10:             if alignmentScore > 0 then
11:                 Substitui nova Trilha (PTrail) como consenso
12:                 Define nova Trilha (PTrail) como trilha atual
13:             end if
14:         end for
15:     end for
16: end while
```

Tabela 2.5: Resultados do alinhamento produzido

| Conjunto | Score | Hits | Misses | Tempo Execução (segundos) |
|---|---|---|---|---|
| Sequencias idênticas | 50 | 24342 | 35658 | 305,77 |
| Sequencias aleatórias | 00 | 25348 | 34652 | 335,36 |
| Sequencias mais conservadas | 10 | 21273 | 38727 | 1016,45 |
| Sequencias menos conservadas | 00 | 18579 | 41421 | 955,49 |

O autor apresentou resultados para sequências idênticas, aleatórias, bem conservadas e pouco conservadas e, embora a solução produza resultados para todos os casos,

alguns pontos são apontados pelo próprio autor como não ideais. Um caso específico, para ilustrar esse argumento, é o fato de que, mesmo no alinhamento de sequências idênticas, ocorreu descasamento em um aminoácido. Outro ponto relevante, é que apenas 6 sequências foram utilizadas para comparação e essa quantidade é muito inferior à quantidade de sequências para aplicações atuais, que podem atingir a casa de milhares de sequências.

### 2.6.2 Alinhamento múltiplo de sequências utilizando algoritmo genético multifunção e colônia de formigas

A proposta de (AMORIM, 2017) é construir um escalonador automático de funções, junto à ferramenta MSA-GA (GONDRO; KINGHORN, 2007), que selecionará entre a função objetivo COFFEE (Consistency based Objective Function For alignmEnt Evaluation) ou WSP (Weighted Sum-of-Pairs), com base no percentual médio de identidade das sequências. O valor do limiar, para escolher uma função ou a outra, sugerido pelo autor é de 35%, entretanto, esse valor é parametrizável na ferramenta.

Além da contribuição já citada, (AMORIM, 2017) também implementou uma etapa de pós-processamento, com base em otimização por colônia de formigas, executado após a conclusão do alinhamento pela ferramenta MSA-GA. O método proposto consiste em construir um grid simplificado, pelo qual as formigas se deslocam. Nesse grid, as formigas possuem três opções, tomar a linha central, superior ou inferir e com isso, os resíduos do alinhamento resultante podem ser mantidos ou substituídos por um *gap*. A Figura 2.14 elucida a decisão da formiga, na etapa de deslocamento.

Figura 2.14: Grid simplificado


Fonte: (AMORIM, 2017)

### 2.6.3 *Chaotic step length artificial bee colony algorithms for protein structure prediction*

Uma área importante da bioinformática, além de reconhecimento de padrões e alinhamento múltiplo de sequências é a área de predição da estrutura proteica. Originalmente, técnicas de ressonância magnética nuclear e cristalografia de raios X eram utilizadas para determinar a estrutura da proteína. Entretanto, essas abordagens requerem um aparato laboratorial custoso e também consomem muito tempo. O trabalho de (SAXENA et al., 2020) versa na linha de predição da estrutura proteica através de um modelo físico computacional. Em geral, os modelos físicos de predição de proteína são construídos em duas fases: inicialmente, um modelo com energia desconhecida é criado e sobre esse modelo são aplicadas funções de otimização para minimizar a energia livre da proteína.

Para construir e otimizar os modelos de energia livre, meta-heurísticas são frequentemente utilizadas, uma dessas meta-heurísticas é a *Artificial Bee Colony*. No trabalho de (SAXENA et al., 2020), valida-se a hipótese da aplicar funções caóticas em uma etapa do algoritmo *Artificial Bee Colony* (ABC) para obter melhores resultados.

Originalmente, o algoritmo ABC atualiza a velocidade das abelhas por meio da fórmula 2.2, em que $\Phi$ é um número gerado aleatoriamente no intervalo $[-1,1]$. A proposição de Saxena é substituir o valor aleatório por um método denominado *Cha-*

*otic Length Separator*. A função modificada é representada conforme Fórmula 2.3.

$$\upsilon_{ij} = x_{ij} + \phi(x_{best\,j} - x_{ij}) \tag{2.2}$$

$$\upsilon_{ij} = x_{ij} + CLS_t(x_{best\,j} - x_{ij}) \tag{2.3}$$

Para calcular o *Chaotic Length Separator*, dez funções de mapa caóticos foram escolhidas. Esses mapas foram denominados *Enhance Chaotic Artificial Bee colony* e para cada função foi atribuído um indice. Algumas delas são: *Chebyshev chaotic* (ECABC1), *Sinusoidal chaotic* (ECABC9) e *Tent chaotic* (ECABC10). Após a computação das funções, o resultado é normalizado por meio da função 2.4, no intervalo $[0.2, 1^{e-10}]$ onde, $t$ denota a iteração atual e $T$ o número máximo de iterações. Por fim, o resultado é aplicado no algoritmo ABC para continuar a busca. A Tabela 2.6 exibe um comparativo do resultado da aplicação do algoritmo original *Artificial Bee colony* e do algoritmo proposto *Enhanced Chaotic Artificial Bee Colony* utilizando a função de mapa caótico ECABC10.

$$N_m(t) = N_m^{max} - \left(\frac{N_m^{max} - N_m^{min}}{T}\right) * t \tag{2.4}$$

Tabela 2.6: Resultados da otimização de dobra da proteína

| Sequencia | Média ABC Original | Média ECABC10 | Desvio ABC Original | Desvio ECABC10 |
|---|---|---|---|---|
| Asmall1 | -0.64937 | -0.64937 | 2.51E-06 | 2.08E-06 |
| As1 | -1.5837 | **-1.58413** | 0.003966 | 0.00432 |
| Am1 | -1.28362 | **-1.39973** | 0.287315 | 0.221548 |
| Rs1 | -1.34887 | **-1.38314** | 0.188762 | 0.23637 |

Os resultados apontados pelo estudo indicam melhor eficácia na conversão do algoritmo ABC com a utilização dos mapas caóticos. Além disso, essa abordagem, com

dez funções distintas, permitiu explorar e ressaltar as diferenças e o comportamento do algoritmo com sua utilização de cada uma delas.

### 2.6.4 *Multiple Sequence Alignment Based on Chaotic PSO*

O trabalho de (LEI; SUN; MA, 2009) apresenta uma abordagem de alinhamento múltiplo de sequências, combinando *chaotic jump* com *Particle Swarm Optimization*. Os autores propõem essa solução para lidar com o problema de convergência prematura, que frequentemente significa que o algoritmo atingiu um ponto de máximo ou mínimo local.

Basicamente, o método proposto percebe a convergência prematura observando dois pontos, primeiro se a distância média das partículas é menor que um *threashold* e segundo se a variância da função de satisfação das partículas é menor do que outro parâmetro de *threshold*.

Ao identificar a situação de convergência prematura, o algoritmo utiliza uma função de mapa logístico para gerar valores no intervalo caótico entre 0 e 1. Em seguida, esses valores são mapeados para posições inteiras correspondente à *gaps* que serão inseridas na sequência alvo. Em situações onde já exista um gap na posição determinada, uma posição aleatória é escolhida, o *gap* anterior é mantido e o novo *gap* é inserido nessa posição.

O algoritmo apresentou soluções melhores para um conjunto de sequências de *Ribonuclease* extraído do *dataset* BAliBase, que são sequências com identidade acima de 35%. Para outras sequências, com identidades menores, como por exemplo *Cytochrome c*, os resultados não são muito melhores, mas ainda assim, a diferença entre os *scores* mínimos e máximos é pequena, o que denota robustez na solução.

Tabela 2.7: Resultados da otimização por PSO caótico

| Identidade | Sequencia | Média | Máximo | Mínimo |
|---|---|---|---|---|
| < 25% | SH3 | 0.5971 | 0.9461 | 0.4559 |
| | twitchin | 0.4721 | 0.5248 | 0.4215 |
| 20% - 40% | SH2 | 0.6235 | 0.6807 | 0.5723 |
| | Cytochrome c | 0.5346 | 0.5731 | 0.4822 |
| > 35% | Ribonuclease | 0.8256 | 0.8761 | 0.7778 |
| | immunophilin c | 0.5146 | 0.5739 | 0.4611 |

### 2.6.5 Considerações sobre os trabalhos correlatos

Como se observam na Tabela 2.8, a aplicação de *chaotic jump* em conjunto com outras técnicas apresentam resultados satisfatórios para o problema de alinhamento múltiplo de sequências, bem como para outros problemas de bioinformática. Por outro lado, é notório que a aplicação de *Ant Colony Optimization* para construção do alinhamento múltiplo de sequências em si, não apresenta resultados satisfatórios. Percebe-se que essa meta-heurística pode ser melhor aproveitada para o refinamento do alinhamento efetuado por outras ferramentas.

Além disso, a utilização de estratégias híbridas e combinação de técnicas se mostram como soluções viáveis e apresentam resultados mais relevantes, especialmente em abordagens que lançam mão de meta-heurísticas. Do ponto biológico, em específico no alinhamento múltiplo de sequência, significados mais relevantes são aquele que mais satisfazem a função de satisfação que foi discorrida na sessão 2.3.

Tabela 2.8: Comparativo entre métodos apresentados nesta seção

| Trabalho | Problema | Método | Aplicou Estado Caótico | Qualidade dos Resultados |
|---|---|---|---|---|
| *An ant colony algorithm for multiple sequence alignment in bioinformatics* | Construção do Alinhamento | ACO | Não | Não obteve bons resultados |
| Alinhamento múltiplo de sequências utilizando algoritmo genético multifunção e colônia de formigas | Construção e Refinamento | GA e ACO | Não | Obteve melhores resultados |
| *Chaotic step length artificial bee colony algorithms for protein structure prediction* | Dobramento de proteínas | ABC | Sim | Obteve melhores resultados |
| *Multiple Sequence Alignment Based on Chaotic PSO* | Construção do Alinhamento | PSO | Sim | Obteve melhores resultados |

# Capítulo 3

# Metodologia

O presente trabalho versa no refinamento do alinhamento múltiplo de sequências por meio de uma hibridização de técnicas, visando obter resultados mais relevantes biologicamente. Para se obter esse objetivo, diversas abordagens foram implementadas e a principal delas, que é o cerne do método proposto, sendo esta o algoritmo de otimização por colônia de formigas. É por meio desse método que as outras abordagens, como o realinhamento de fragmentos das sequências e a aplicação do método caótico são aplicadas.

Este capítulo foi subdividido em 5 partes: Visão geral do método, Otimização por colônia de formigas, Alinhamento múltiplo de sequências, Realinhamento de subsequências e por fim, *Chaotic Jump*. Na visão geral apresenta-se um pseudocódigo do método para facilitar a compreensão. Na seção de Otimização por colônia de formigas, será detalhado o algoritmo de otimização por colônia de formigas, sua modelagem para o problema de alinhamento de sequências e os parâmetros utilizados. Na seção de Alinhamento múltiplo de sequências, será elucidado qual estratégia de alinhamento múltiplo será utilizada e como a pontuação desse alinhamento será calculada. Na seção de Realinhamento de subsequências, será elucidado o processo de realinhamento e seu objetivo com relação ao método como um todo. Na última etapa, serão demonstradas as funções e parâmetros relacionados ao *Chaotic Jump* e em qual ponto do método ele

será usado.

## 3.1 Visão geral do método

O método proposto consiste em aplicar Otimização por colônia de formigas para refinar o alinhamento múltiplo de sequências construído pelas ferramentas KAlign e Clustal Ômega. Para se obter esse objetivo, este método captura o alinhamento inicial produzido e executar efetuar o refinamento utilizando a otimização ACO, recomputar o alinhamento par-a-par e efetuar um novo alinhamento pela técnica *Center Star*.

Quando o alinhamento resultante do refinamento for melhor do que o anterior, esse alinhamento é armazenado como possível solução final, caso contrário, um contador é incrementado. Caso o contador obtiver um threshold pré-determinado, o algoritmo entra em uma fase de realinhamento. Nessa fase, uma posição das sequencias é escolhida utilizando sorteio caótico, onde é aplicado o *chaotic jump* num valor inicialmente aleatório. Somente o primeiro sorteio valores aleatórios são utilizados, pois nas próximas ocorrências, o valor é retroalimentado na própria formula caótica para recalibrar a posição inicial da recombinação.

A etapa de realinhamento seleciona a posição inicial e o tamanho que será recalibrado e efetua um alinhamento múltiplo de apenas esse pedaço das sequencias e depois insere-as novamente nas posições originais. Após esse realinhamento, uma nova iteração será efetuada com as sequencias modificadas para aferir novamente a qualidade do alinhamento efetuado. Esse ciclo se repete até que o limite de iterações seja atingido. No Algoritmo 2 ilustram-se essas etapas que foram descritas nos parágrafos anteriores.

**Algorithm 2** Visão geral do método proposto

```
1: for ant = 1,2,...,N do
2:     for iteration = 1,2,...,N do
3:         Gerar alinhamento múltiplo inicial (ClustalW e KAlign)
4:         Computar os alinhamentos par-a-par
5:         Executar Center Star para Alinhamento múltiplo de sequências
6:         Calcular o score do alinhamento múltiplo de sequências
7:         if Alinhamento múltiplo melhor que o anterior then
8:             Armazenar melhor alinhamento
9:         else
10:            Computar execução sem melhorar
11:            if Atingiu limite de execuções sem melhorar then
12:                Realinhar fragmentos de todas as sequências
13:            end if
14:        end if
15:    end for
16: end for
```

## 3.2 Otimização por colônia de formigas

A abordagem do método proposto, especificamente no tocante à otimização por colônia de formigas baseia-se no trabalho de (AMORIM, 2017). Em seu trabalho, o alinhamento múltiplo de sequências é melhorado por meio de mutações nas sequências pré-alinhadas utilizando um *grid* simplificado para deslocar as formigas e atualizar a trilha de feromônio por esse *grid*. Conforme as formigas vão selecionando os caminhos pelo *grid*, elas também vão depositando os feromônios na trilha escolhida, de modo que a trilha que gerar melhor resultado biológico será preferida e trilhas com menores resultados serão descartadas.

Durante o deslocamento pela trilha, quando a formiga toma um caminho no *grid* da linha superior ou linha inferir, um *gap* é inserido na linha não escolhida, se a formiga seguir pela linha do meio, nenhum gap é inserido e as duas sequências não são modificadas. Por causa das mutações, as sequências resultantes são previamente redimensionadas para um tamanho de (2 * *maior*) posições, em que *maior* corresponde à sequência de maior tamanho e os espaços vazios são preenchidos com gaps. Esse procedimento é feito apenas para normalizar o tamanho das sequências e permitir com

que os realinhamentos caibam nas sequências em suas posições iniciais.

No presente trabalho, algumas abordagens foram modificadas na tentativa de obter um alinhamento com maior significância biológica. Primeiro, as regras de transições do método atual são diferentes do trabalho original de (AMORIM, 2017), além disso, o método proposto também aplica o *chaotic jump* para selecionar posições da sequência para efetuar um realinhamento, quando atingir um determinado número de iterações sem melhora na qualidade do alinhamento múltiplo.

No início do processo, as formigas utilizam as ferramentas ClustalW e KAlign para produzir um alinhamento básico inicial. Essa ferramentas foram escolhidas pois o tempo de resposta delas está na casa de milissegundos e a qualidade do alinhamento produzido é razoável.

Depois da etapa de construção do alinhamento inicial, é efetuado o alinhamento par-a-par de cada sequência, utilizando a meta-heurística *ant colony optimization* para convergir em alinhamentos biologicamente mais relevantes.

No tocante às transições, o método atual não compartilhará uma mesma trilha para todas as sequências. Na presente abordagem, existe uma trilha que será computada separadamente para cada sequência inicial do alinhamento par-a-par. Na prática, isso corresponde a uma dimensão a mais na matriz de feromônios, que será indexada pelo número da sequência inicial do alinhamento par-a-par.

A equação 3.1 denota a probabilidade de transição para os *grids* do alinhamento par-a-par. A dimensão $i = (0,1,2)$ corresponde ao caminho que a formiga escolherá, 0 para a linha superior, 1 para linha do meio e 2 para linha inferior. A dimensão $j = 1..N_{esimo}$ corresponde ao resíduo ou aminoácido da sequência e a dimensão $k = 1..T$ corresponde ao número da sequência inicial do alinhamento par-a-par, em que $T$ é o número total de sequências. O valor de $\eta$ é fixado em 1, visto que, como o *grid* simplificado possui apenas 3 opções, esse valor não interfere na convergência. O valor de $\alpha$ e $\beta$ são parâmetros do método e seus valores serão definidos pelo usuário. Todos

os valores dos parâmetros e seus critérios de escolha serão apresentados no Capítulo 4.

$$P_{ijk}^{z} = \frac{\tau_{ijk}(t)^{\alpha}\eta_{ijk}^{\beta}}{\sum \tau_{ijk}(t)^{\alpha}\eta_{ijk}^{\beta}} \tag{3.1}$$

Durante o processo de deslocamento das formigas pelas trilhas, o algoritmo utiliza a equação 3.1 para calcular o valor probabilístico $q$ da formiga escolher qualquer um dos três caminhos possíveis. Nessa etapa, é necessário satisfazer a restrição $q \leq q_0$ para que a decisão das trilhas seja tomada com base no valor de $q$. Caso $q > 0$, a trilha é selecionada aleatoriamente. Caso as sequências sejam de tamanhos diferentes e uma delas chegar ao fim, a sequência menor será completada com *gaps* até que estejam do mesmo tamanho.

Após o alinhamento de cada par, é computado um *score* para o alinhamento. Esse *score* é utilizado para atualizar a intensidade da trilha de feromônios, que serve de guia para as formigas decidirem entre um caminho ou outro. A equação que representa a evaporação de feromônios da trilha, pode ser observada em 3.2. Nessa equação, $p$ é um parâmetro no intervalo $[0,1]$ e representa a taxa de evaporação. O valor de $\Delta\tau_{ijk}$ é definido pela equação 3.3, em que, $Q$ e $MaxScore$ são parâmetros para o algoritmo e $Score^{z}$ é a pontuação calculada para o par de sequências recém alinhados e é definida pela equação 3.4.

$$\tau_{ijk}(t+1) = (1-p) * \tau_{ijk}(t) + \Delta\tau_{ijk} \tag{3.2}$$

$$\begin{aligned} &\Delta(\tau_{ijk}) = Q * (Score^{z} + MaxScore)/2 \Rightarrow \text{se a formiga z passar pelo grid (i,j,k)} \\ &\Delta(\tau_{ijk}) = 0 \Rightarrow \text{se a formiga z não passar pelo grid (i,j,k)} \end{aligned} \tag{3.3}$$

$$Score(a,b) = \sum_{i=1}^{\text{len}} \alpha(A'[i], B'[i]) \tag{3.4}$$

Na equação de pontuação (3.4), $a$ e $b$ são as duas sequências que formam um par para o alinhamento e $A'$ e $B'$ são as sequências modificadas, com possíveis gaps além dos resíduos originais. Note que, como as sequências estão alinhadas, o somatório necessita apenas computar a função $\alpha$ para *len* resíduos da sequências, pois estas estarão do mesmo tamanho ao final do alinhamento par-a-par. O valor de $\alpha$ foi definido da seguinte maneira: caso o resíduo na posição $i$ seja igual nas duas sequências, seu valor será 2, caso contrário, será -1.

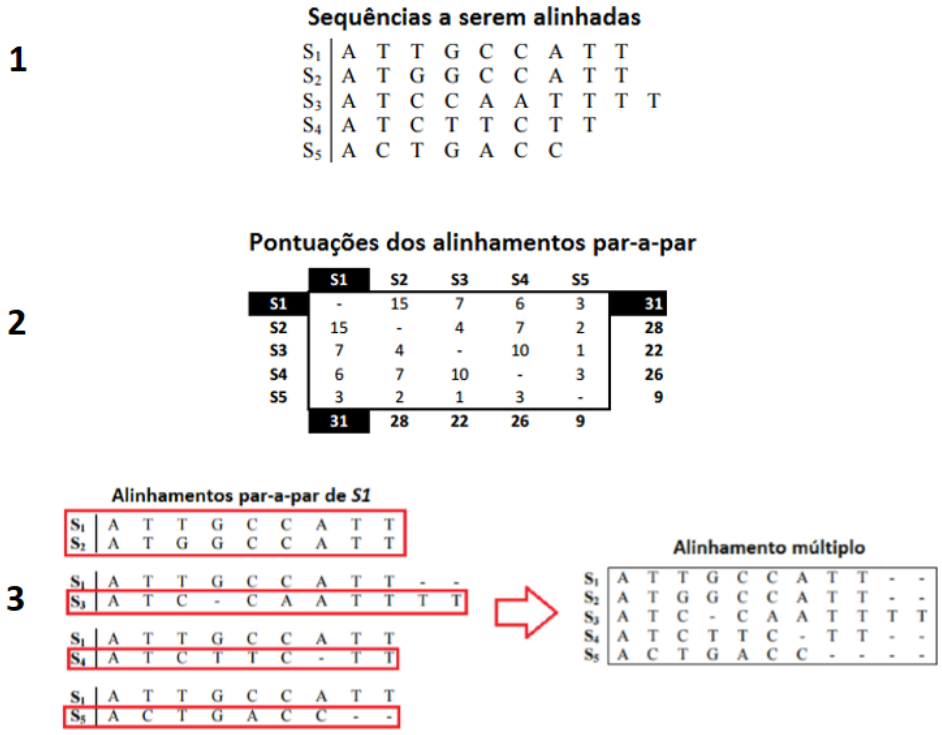
Esse processo de alinhamento par-a-par é repetido para todas as sequências e após a conclusão, uma matriz de 2 dimensões é gerada. Essa matriz possui 2 dimensões para permitir o armazenamento dos alinhamentos de todas contra todas as sequências. Além disso, outro vetor resultante armazena a pontuação acumulada de todos os alinhamentos para cada sequência. Ambos artefatos são utilizados na próxima etapa que é o alinhamento múltiplo de sequências.

## 3.3 Alinhamento múltiplo de sequências

Dentre vários métodos de alinhamento múltiplos, este trabalho utiliza o método *Center Star* (ZOU et al., 2015). Esse método consiste em selecionar a sequência que obteve maior pontuação, iniciar o alinhamento múltiplo por ela e progressivamente inserir as outras sequências, levando em consideração a matriz de alinhamento par-a-par. Na Figura 3.1, observa-se o processo de construção do alinhamento múltiplo por essa técnica, esse processo está dividido em três etapas, com uma enumeração à esquerda indicando cada etapa. Na etapa 1, estão as sequências a serem alinhadas. Na etapa 2, pode ser observada uma representação da pontuação do alinhamento par-a-par, o objetivo é encontrar a sequência que obtiver maior pontuação nessa matriz, para isso

cada sequencia é posicionada na matriz com um score que corresponde ao resultado do alinhamento par-a-par efetuado para cada uma das outras sequencias presentes. E, finalmente, na etapa 3, a sequência $s1$ foi selecionada como *estrela central* e o alinhamento é construído a partir dos alinhamentos par-a-par com essa sequência.

Figura 3.1: Processo de AMS por Center Star




Fonte: Adaptado de (AMORIM, 2017)

Efetuado o alinhamento múltiplo de sequências, o seu *score* é aferido e se for melhor que o anterior (ou caso seja o primeiro), seu valor é armazenado. Como se utiliza um método iterativo, múltiplos alinhamentos serão produzidos e somente o melhor entre todos será utilizado como resultado.

Para calcular o *score* do alinhamento múltiplo de sequências, neste trabalho é utilizada a função *weighted sum-of-pairs* (WSP) (RUBIO-LARGO; VEGA-RODRÍGUEZ; GONZÁLEZ-ÁLVAREZ, 2015). Essa é a função objetivo que é maximizada em na heurística desenvolvida no presente trabalho.

Na equação 3.5, que define a função WSP, $AL$ é o tamanho das sequências alinha-

das e $k$ é o total de sequências. Portanto, $l$ corresponde a cada resíduo ou posição da sequência e $s'_i$ corresponde à sequência resultante do alinhamento múltiplo. A função $\mathrm{SP}(l)$ é definida pela equação 3.6.

$$\mathrm{WSP}(S') = \sum_{l=1}^{\mathrm{AL}} \mathrm{SP}(l) - \sum_{i=1}^{\mathrm{k}} \mathrm{AGP}(s'_i) \tag{3.5}$$

$$SP(l) = \sum_{i=1}^{k-1} \sum_{j=1}^{k} W_{ij} \times \delta(s'_{i,l}, s'_{j,l}) \tag{3.6}$$

Em 3.6, $\delta$ corresponde à alguma matriz de substituição (BLOSUM, PAM). A função dessa matriz na equação é ponderar o peso da substituição de um aminoácido por outro. No presente trabalho, utilizamos a matriz BLOSUM62 (EDDY, 2004) e os valores dessa matriz podem ser observados na Figura 3.2. A função $W_{i,j}(a,b)$ é definida pela equação 3.7, em que $LD(s_i, s_j)$ corresponde à distância de Levenshtein e $s_i$ e $s_j$ correspondem às sequências não alinhadas.

Figura 3.2: Matriz BLOSUM62

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X | * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 | 0 | -4 |
| R | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 | -1 | 0 | -1 | -4 |
| N | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 | 3 | 0 | -1 | -4 |
| D | -2 | -2 | 1 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 | 4 | 1 | -1 | -4 |
| C | 0 | -3 | -3 | -3 | 9 | -3 | -4 | -3 | -3 | -1 | -1 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 | -3 | -3 | -2 | -4 |
| Q | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 | 0 | 3 | -1 | -4 |
| E | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 | -1 | -4 |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 | -1 | -2 | -1 | -4 |
| H | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 | 0 | 0 | -1 | -4 |
| I | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 | -3 | -3 | -1 | -4 |
| L | -1 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 | -4 | -3 | -1 | -4 |
| K | -1 | 2 | 0 | -1 | -3 | 1 | 1 | -2 | -1 | -3 | -2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 0 | 1 | -1 | -4 |
| M | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 | -3 | -1 | -1 | -4 |
| F | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 | -3 | -3 | -1 | -4 |
| P | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 | -2 | -1 | -2 | -4 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 | 0 | 0 | 0 | -4 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 | -1 | -1 | 0 | -4 |
| W | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 | -4 | -3 | -2 | -4 |
| Y | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 | -3 | -2 | -1 | -4 |
| V | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 | -3 | -2 | -1 | -4 |
| B | -2 | -1 | 3 | 4 | -3 | 0 | 1 | -1 | 0 | -3 | -4 | 0 | -3 | -3 | -2 | 0 | -1 | -4 | -3 | -3 | 4 | 1 | -1 | -4 |
| Z | -1 | 0 | 0 | 1 | -3 | 3 | 4 | -2 | 0 | -3 | -3 | 1 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 | -1 | -4 |
| X | 0 | -1 | -1 | -1 | -2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -2 | 0 | 0 | -2 | -1 | -1 | -1 | -1 | -1 | -4 |
| * | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 |

Fonte: Retirada do site da NCBI

$$W_{i,j}(a,b) = 1 - \frac{\mathrm{LD}(s_i, s_j)}{max(len(s_i), len(s_j))} \tag{3.7}$$

Por fim, na equação 3.8, tem-se a função de penalidade pelos *gaps*, em que $g_0$ é o peso para cada abertura de um novo gap e $g_e$ é o peso aplicado para o total de *gaps* consecutivos após cada nova abertura.

$$AGP(s'_i) = (g_0 \times N_{\text{gaps}}) + (g_e \times N_{\text{spaces}}) \tag{3.8}$$

Em algumas situações, pontos de máximo local são encontrados e o alinhamento produzido, mesmo após várias de iterações, não progride como desejado. Para resolver esse problema é efetuado um realinhamento localizado de todas as sequências, aplicando o conceito de *chaotic jump* ao invés de utilizar posições aleatórias das sequências. Esse processo será melhor descrito nas próximas seções.

## 3.4 Realinhamento de subsequências

Quando o algoritmo realiza um determinado número de iterações e a qualidade do alinhamento múltiplo de sequências não obtiver melhora, uma etapa de realinhamento é efetuada em subsequências de todas as sequências envolvidas no alinhamento múltiplo. O valor do número de iterações é o limitante para selecionar a posição na sequência, em que as subsequências serão extraídas, nas quais é aplicada a função de mapa logístico. O resultado dessa função é traduzido para uma posição na sequência e o alinhamento é efetuado por meio da ferramenta *KAlign*. Essa fase de realinhamento pode ser observada por meio do Algoritmo 3:

**Algorithm 3** Funcionamento do realinhamento

1: Calcular limites do realinhamento
2: Extrair fragmentos com mesmo tamanho de todas as sequências
3: Gerar novo arquivo com apenas fragmentos das sequências originais
4: Executar *KAlign*
5: Obter sequências alinhadas pelo KAlign
6: Injetar sequências alinhadas nos intervalos originalmente extraídos

Na primeira parte é sorteado o tamanho do bloco que será extraído das sequências e realinhado. Esse bloco se mantém no intervalo entre $[5, 25]$ por cento do tamanho da menor sequência envolvida no alinhamento, nesse caso, como as sequências são originadas por meio da ClustalW ou *KAlign* elas terão o mesmo tamanho.

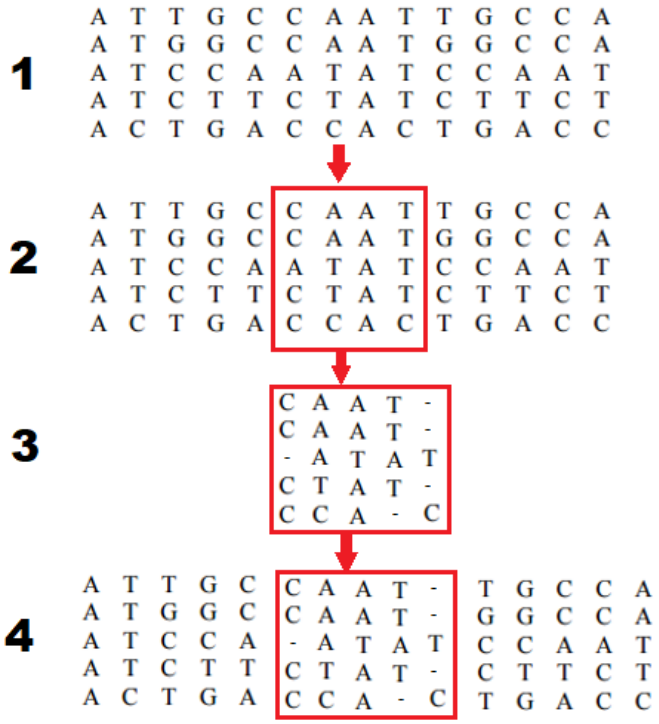
Na segunda parte, a posição inicial do bloco é determinada e em conjunto com o tamanho do bloco selecionado, todas as sequências são percorridas para gerar os fragmentos que serão utilizados para o realinhamento.

A ferramenta *KAlign* é executada por linha de comando, especificando o arquivo com as subsequências e especificando também o nome do arquivo esperado como saída. Ambos os arquivos são esperados no formato *.fasta*.

Finalmente, o resultado da execução do *KAlign* é capturado e as sequências são injetadas novamente nos espaços originais de onde foram extraídas. Uma descrição desse processo pode ser observada na Figura 3.3. Na primeira etapa estão todas as

sequências do AMS, na segunda etapa é feita a seleção do bloco que será realinhado, a terceira etapa elucida um possível realinhamento e na última etapa a inserção do bloco nas posições originais.

Figura 3.3: Processo de Realinhamento




Fonte: Elaborado pelo autor

## 3.5 *Chaotic Jump*

O *Chaotic Jump* é empregado no método proposto na etapa de seleção da posição das sequências para efetuar o realinhamento. Primeiramente, definiu-se a função de mapa logístico, conforme a equação 3.9. Para o presente trabalho, foi utilizado o valor de $\alpha = 4$, pois com esse valor o resultado da função, para qualquer valor de $x$, sempre ocorrerá no intervalo $[0,1]$. Como essa função espera um valor para calcular o próximo, o valor inicial é gerado aleatoriamente no mesmo intervalo.

$$\delta_p = x_{n+1} = \alpha x_n (1 - x_n) \tag{3.9}$$

É possível observar, no Algoritmo 4, como o valor de $\delta_p$ é transformado em uma posição da sequência e como o tamanho do bloco que será realinhado é determinado. Nas duas primeiras linhas, calculam-se respectivamente os limites mínimo e máximo do tamanho do bloco que será realinhado. Na terceira linha, o tamanho de fato do bloco é calculado aleatoriamente no intervalo dos limites definidos anteriormente. Logo, é definido o tamanho limite para o sorteio da posição, para evitar que o número sorteado ultrapasse o comprimento das sequências. Depois disso, as posições iniciais e finais são calculadas e o processo de extração ocorre como demonstrado na seção anterior. Os valores de $P_{\text{min}}$ e $P_{\text{max}}$ são parâmetros do algoritmo e inicializados por padrão em, respectivamente, 5 e 25 do tamanho total das sequências.

**Algorithm 4** Pseudo-código utilizado para determinar as posições e tamanhos do realinhamento

1: $Block_{\text{min}} = P_{\text{min}} * T_{\text{seq}}$
2: $Block_{\text{max}} = P_{\text{max}} * T_{\text{seq}}$
3: $Block_{\text{size}} = Rand(Block_{\text{min}}, Block_{\text{max}})$
4: $T_{\text{limit}} = T_{\text{seq}} - Block_{\text{size}}$
5: $P_{initial} = \delta_p * T_{\text{limit}}$
6: $P_{final} = P_{initial} + Block_{\text{size}}$

# Capítulo 4

# Testes e resultados

Neste capítulo são apresentadas a plataforma efetuada para os testes e os resultados obtidos a partir da implementação do método proposto no presente trabalho.

## 4.1 Plataformas de Testes

Todos os testes deste trabalho foram executados em computador com Sistema Operacional Windows(R) 10 Enterprise. Esse equipamento possui as seguintes configurações:

- Fabricante: Lenovo, Inc.
- Modelo: ThinkPad T490
- Memoria RAM: 16 Gigabytes
- Processador: Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11GHz
- Quantidade de núcleos: 4
- Quantidade de threads: 8
- Disco: SSD 240 Gigabytes

## 4.2 Parâmetros utilizados

Os parâmetros utilizados no método proposto estão na Tabela 4.1:

Tabela 4.1: Parâmetros utilizados na ferramenta ACO

| Nome | Valor | Descrição |
|---|---|---|
| NumberOfAnts | 20 | Quantidade de formigas utilizadas |
| NotImprovedLimit | 10 | Número mínimo de iterações sem melhora para efetuar o realinhamento |
| NumberOfIterations | 300 | Número de iterações por formiga |
| $\alpha$ | 1.0 | Valor de Alpha para o ACO |
| $\beta$ | 5.0 | Valor de Beta para o ACO |
| $Q_0$ | 0.3 | Valor de Q0 para o ACO |
| RO | 0.2 | Valor de RO para o ACO |
| Q | 0.001 | Valor de Q para o ACO |
| $\tau_0$ | 10 | Valor inicial do feromônio para o ACO |
| MaxScore | 200.0 | Valor de MaxScore para o ACO |
| $G_e$ | 0.85 | Valor da penalidade para extensão de *gaps* da função WSP |
| $G_0$ | 6 | Valor da penalidade por abrir *gaps* da função WSP |
| MinFragmentPerc | 0.05 | Percentual mínimo do tamanho bloco de realinhamento em relação ao tamanho sequência |
| MaxFragmentPerc | 0.25 | Percentual máximo do tamanho bloco de realinhamento em relação ao tamanho sequência |

No intuito de comparar a eficiência do método proposto, o mesmo conjunto de sequências foi, paralelamente, alinhado por meio da ferramenta MSA-GA, KAlign e Clustal Ômega. Os parâmetros utilizados pela ferramenta MSA-GA ferramenta estão na Tabela 4.2:

Tabela 4.2: Parâmetros utilizados na ferramenta MSA-GA

| Nome | Valor | Descrição |
|---|---|---|
| PopulationSize | 1000 | Tamanho da População para a MSA-GA |
| Generations | 10000 | Quantidade de Gerações da MSA-GA |
| BlockMutation | 0.01 | Percentual da taxa de mutação da MSA-GA |
| GapExtMutation | 0.05 | Gap Extension Mutation da MSA-GA |
| GapRedMutation | 0.05 | Gap Reduction Mutation da MSA-GA |
| HorCrossover | 0.8 | Percentual da Taxa de Cross-over horizontal da MSA-GA |
| VerCrossover | 0.8 | Percentual da Taxa de Cross-over vertical da MSA-GA |
| HorVerRation | 0.5 | Horizontal/Vertical Ratio da MSA-GA |
| TournamentSize | 2 | Tournament Size da MSA-GA |
| OffSet | 0 | Valor do Offset da MSA-GA |
| MaxSeqSize | 1.2 | Tamanho máximo da sequência alinhada produzida pela MSA-GA |
| Matrix | Blosum62 | Matriz de substituição utilizada (A mesma foi usada pela ACO) |

## 4.3 *Benchmark* BAliBase

O *benchmark* BAliBase 3 é um banco de dados composto por sequências desalinhadas e alinhamentos de referência. Nesse banco de dados, as sequências foram refinadas manualmente e selecionadas especificamente para avaliação e comparação de programas de alinhamento múltiplo de sequências. Em cada agrupamento de referência, esse *benchmark* possui subconjuntos de regiões de sequências, essas regiões foram separadas por diversos atributos, como por exemplo, blocos de conservação, tamanho das sequências, similaridade, presença de inserções. No presente trabalho foi utilizada a versão 3 do *benchmark*, nessa versão as características das regiões foram agrupadas conforme o seguinte modo:

- Referência 1: Sequências equi-distantes com dois níveis diferentes de conservação,
- Referência 2: Famílias alinhadas com sequências "orfãos"altamente divergentes,

- Referência 3: Subgrupos com resíduos de menos de 25% de identidade entre os grupos,
- Referência 4: Sequências extensão terminal N/C,
- Referência 5: Inserções internas.

## 4.4 Resultados

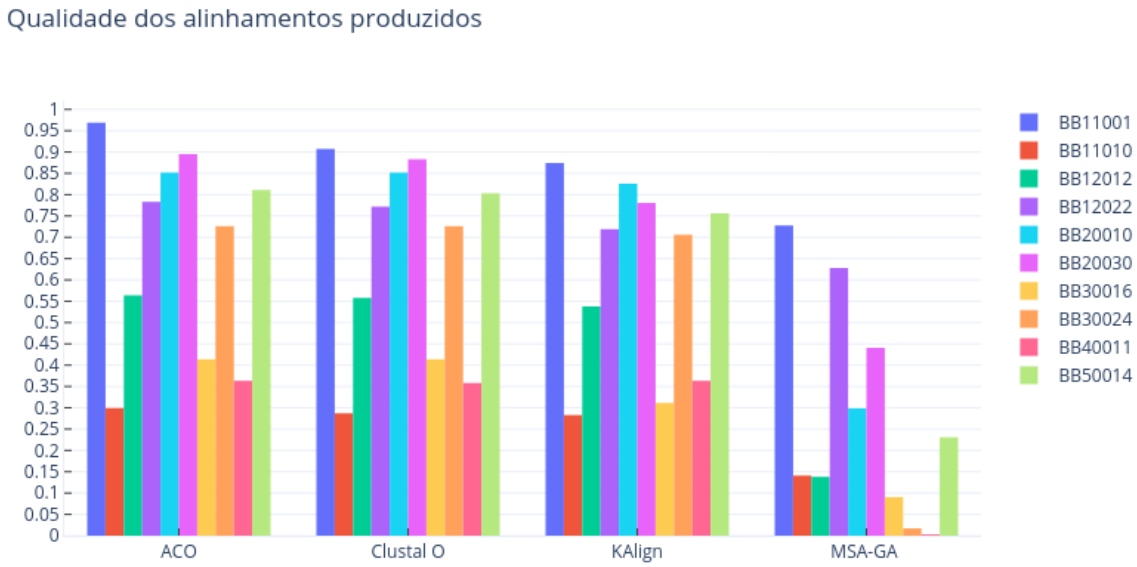
Todos os testes efetuados foram executados sobre o *benchmark* BAliBase 3, foram selecionadas 10 regiões aleatórias de todas as famílias do BAliBase e os testes foram repetidos 6 vezes em cada ferramenta para obter uma média e desvio padrão. Para medir a qualidade do alinhamento, foi utilizada a ferramenta *QScore*, essa ferramenta é uma versão customizada da ferramenta BAliScore, apenas para aceitar o formato de arquivo .fasta.

O comparativo dos alinhamentos efetuados pelas ferramentas são demonstrados na Tabela 4.3 e na Figura 4.1.

Tabela 4.3: Média e desvio padrão dos resultados obtidos pelos testes

| Região | QScore Médio | TC Maior | Método | QScore Desvio |
|---|---|---|---|---|
| **BB11001** | **0.966** | **0.940** | **ACO** | 0.07 |
| BB11001 | 0.907 | 0.830 | Clustal O | 0 |
| BB11001 | 0.874 | 0.727 | KAlign | 0 |
| BB11001 | 0.720 | 0.492 | MSA-GA | 0.04 |
| **BB11010** | **0.289** | **0.177** | **ACO** | 0.02 |
| BB11010 | 0.287 | 0.140 | Clustal O | 0 |
| BB11010 | 0.283 | 0.148 | KAlign | 0 |
| BB11010 | 0.144 | 0.004 | MSA-GA | 0.15 |
| **BB12012** | **0.566** | **0.424** | **ACO** | 0.07 |
| BB12012 | 0.558 | 0.410 | Clustal O | 0 |
| BB12012 | 0.538 | 0.387 | KAlign | 0 |
| BB12012 | 0.133 | 0.022 | MSA-GA | 0.13 |
| **BB12022** | **0.786** | **0.601** | **ACO** | 0.11 |
| BB12022 | 0.772 | 0.532 | Clustal O | 0 |
| BB12022 | 0.719 | 0.468 | KAlign | 0 |
| BB12022 | 0.613 | 0.377 | MSA-GA | 0.05 |
| **BB20010** | **0.852** | **0.535** | **ACO** | 0 |
| **BB20010** | **0.852** | **0.535** | **ClustalO** | 0 |
| BB20010 | 0.826 | 0.440 | KAlign | 0 |
| BB20010 | 0.298 | 0.001 | MSA-GA | 0.08 |
| **BB20030** | **0.897** | **0.096** | **ACO** | 0.07 |
| BB20030 | 0.883 | 0.089 | Clustal O | 0 |
| BB20030 | 0.781 | 0.052 | KAlign | 0 |
| BB20030 | 0.444 | 0.027 | MSA-GA | 0.11 |
| **BB30016** | **0.414** | **0.064** | **ACO** | 0 |
| **BB30016** | **0.414** | **0.064** | **ClustalO** | 0 |
| BB30016 | 0.311 | 0.014 | KAlign | 0 |
| BB30016 | 0.0907 | 0.014 | MSA-GA | 0 |
| **BB30024** | **0.726** | **0.448** | **ACO** | 0 |
| **BB30024** | **0.726** | **0.448** | **ClustalO** | 0 |
| BB30024 | 0.706 | 0.392 | KAlign | 0 |
| BB30024 | 0.017 | 0 | MSA-GA | 0.12 |
| **BB40011** | **0.363** | 0 | **ACO** | 0 |
| BB40011 | 0.358 | 0 | Clustal O | 0 |
| **BB40011** | **0.363** | 0 | **KAlign** | 0 |
| BB40011 | 0.002 | 0 | MSA-GA | 0.09 |
| **BB50014** | **0.811** | **0.398** | **ACO** | 0.02 |
| BB50014 | 0.803 | 0.395 | Clustal O | 0 |
| BB50014 | 0.756 | 0.270 | KAlign | 0 |
| BB50014 | 0.231 | 0.018 | MSA-GA | 0.01 |

Figura 4.1: Resultado geral da qualidade dos alinhamentos produzidos




Fonte: Autor

O valor de QScore é calculado sempre no intervalo $[0,1]$ e quanto mais próximo de 1, melhor o resultado. O Valor de TC representa outro *score* relacionado ao total de colunas conservadas. Enquanto o valor de QScore indica a qualidade do alinhamento, o valor de TC indica a consistência do alinhamento múltiplo efetuado. Assim como no caso de QScore, o valor também é gerado no intervalo $[0,1]$ e quanto mais próximo de 1, melhor. Ambas as métricas são amplamente utilizadas na literatura, como pode ser observado nos trabalhos de (AMORIM, 2017), (RUBIO-LARGO; VEGA-RODRÍGUEZ; GONZÁLEZ-ÁLVAREZ, 2016) e (LEI; SUN; MA, 2009).

# Capítulo 5

# Conclusão

## 5.1 Conclusões gerais

Neste trabalho foi proposto e implementado um método para alinhamento múltiplo de sequências utilizando *Ant Colony Optimization* em conjunto com uma abordagem de estado caótico. Para validar os resultados, o proponente efetuou comparações do novo método com três outros métodos também implementados em ferramentas, denominadas MSA-GA, KAlign e Clustal Ômega.

Por meio da análise dos resultados e da tabela comparativa podemos observar que o método proposto demonstrou melhores resultados, no tocante à qualidade do alinhamento produzido, em mais de cinquenta por cento das execuções, para todas as regiões utilizadas do *benchmark*. Na Tabela 4.3, pode ser observado que o valor do método ACO é no mínimo tão bom quanto o melhor alinhamento entre as ferramentas Clustal Ômega e KAlgin. Esse comportamento é esperado, visto que para efetuar os realinhamentos, essas ferramentas são utilizadas para gerar um alinhamento inicial. E como o método proposto é um refinamento do alinhamento múltiplo, a qualidade será pelo menos a tão boa quanto o alinhamento produzido pelo alinhamento inicial. Quando se observa os valores dos desvios padrões, percebe-se também que o método oferece uma solução robusta, pois os valores de desvios são baixos e similares aos da ferramenta

MSA-GA.

## 5.2 Atividades futuras

Como atividade futura, o proponente sugere uma melhoria do algoritmo, para tentar melhorar o tempo de resposta do programa sem perder a qualidade dos alinhamentos, por meio de abordagens paralelas. Como a implementação do método foi efetuada na linguagem de programação C#, que possui uma excelente biblioteca para processamento com múltiplos núcleos, o esforço para paralelizar o método é relativamente pequeno e poderá reduzir consideravelmente o tempo de execução.

# Referências


ALATAS, B. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications*, Elsevier, v. 37, n. 8, p. 5682–5687, 2010.

ALBERTS, B.; JOHNSON, A.; LEWIS, J.; MORGAN, D.; RAFF, M.; ROBERTS, K.; WALTER, P.; WILSON, J.; HUNT, T. *Biologia molecular da célula*. [S.l.]: Artmed Editora, 2010.

AMORIM, A. R. *Alinhamento múltiplo de sequências utilizando algoritmo genético multifunção e colônia de formigas*. 89 p. Dissertação (Mestrado) — Universidade Estadual Paulista (UNESP), 2017.

AMORIM, A. R.; VISOTAKY, J. M. V.; CONTESSOTO, A. D. G.; NEVES, L. A.; SOUZA, R. C. G. D.; VALÊNCIO, C. R.; ZAFALON, G. F. D. Performance improvement of genetic algorithm for multiple sequence alignment. In: IEEE. *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. [S.l.], 2016. p. 69–72.

AMORIM, A. R.; ZAFALON, G. F. D.; NEVES, L. A.; PINTO, A.; VALÊNCIO, C. R.; MACHADO, J. M. 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.

COHEN, J. Bioinformatics—an introduction for computer scientists. *ACM Computing Surveys (CSUR)*, ACM, v. 36, n. 2, p. 122–158, 2004.

COULL, S.; BRANCH, J.; SZYMANSKI, B.; BREIMER, E. Intrusion detection: A bioinformatics approach. In: IEEE. *19th Annual Computer Security Applications Conference, 2003. Proceedings.* [S.l.], 2003. p. 24–33.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, IEEE, v. 1, n. 1, p. 53–66, 1997.

EDDY, S. R. Where did the blosum62 alignment score matrix come from? *Nature biotechnology*, Nature Publishing Group, v. 22, n. 8, p. 1035–1036, 2004.

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

ISHIKAWA, M.; TOYA, T.; HOSHIDA, M.; NITTA, K.; OGIWARA, A.; KANEHISA, M. Multiple sequence alignment by parallel simulated annealing. *Bioinformatics*, Oxford University Press, v. 9, n. 3, p. 267–273, 1993.

KAISER, C. A.; KRIEGER, M.; LODISH, H.; BERK, A. *Molecular cell biology.* [S.l.]: WH Freeman, 2007.

KATOH, K.; MISAWA, K.; KUMA, K.-i.; MIYATA, T. 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.

KATOH, K.; STANDLEY, D. M. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, Society for Molecular Biology and Evolution, v. 30, n. 4, p. 772–780, 2013.

KATOH, K.; TOH, H. Parallelization of the mafft multiple sequence alignment program. *Bioinformatics*, Oxford University Press, v. 26, n. 15, p. 1899–1900, 2010.

KAUR, G.; ARORA, S. Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering*, Elsevier, v. 5, n. 3, p. 275–284, 2018.

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.

KIRKPATRICK, S.; GELLATT, C.; VECCHI, M. Optimization by simulated annealing, ibm thomas j. *Watson research Center, Yorktown Heights, NY*, 1982.

LEE, Z.-J.; SU, S.-F.; CHUANG, C.-C.; LIU, K.-H. 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.-j.; SUN, J.-j.; MA, Q.-z. Multiple sequence alignment based on chaotic pso. In: SPRINGER. *International Symposium on Intelligence Computation and Applications*. [S.l.], 2009. p. 351–360.

MORGENSTERN, B.; FRECH, K.; DRESS, A.; WERNER, T. Dialign: finding local similarities by multiple sequence alignment. *Bioinformatics (Oxford, England)*, v. 14, n. 3, p. 290–294, 1998.

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

MOUNT, D. W. *Bioinformatics: sequence and genome analysis. 2nd*. [S.l.]: Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press. xii, 2004. v. 692.

NAKAMURA, T.; YAMADA, K. D.; TOMII, K.; KATOH, K. Parallelization of mafft for large-scale multiple sequence alignments. *Bioinformatics*, Oxford University Press, v. 34, n. 14, p. 2490–2492, 2018.

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.

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.; O'BRIEN, E. A.; HIGGINS, D. G. Raga: Rna sequence alignment by genetic algorithm. *Nucleic acids research*, Oxford University Press, v. 25, n. 22, p. 4570–4580, 1997.

PROSDOCIMI, F.; COUTINHO, G.; NINNECW, E.; SILVA, A. F.; REIS, A. N. dos; MARTINS, A. C.; SANTOS, A. C. F. dos; JÚNIOR, A. N.; FILHO, F. C. Bioinformática: manual do usuário. *Biotecnologia Ciência & Desenvolvimento*, v. 29, p. 12–25, 2002.

RIAZ, T.; WANG, Y.; LI, K.-B. Multiple sequence alignment using tabu search. In: AUSTRALIAN COMPUTER SOCIETY, INC. *Proceedings of the second conference on Asia-Pacific bioinformatics-Volume 29*. [S.l.], 2004. p. 223–232.

ROSEN, K. H. *Handbook of discrete and combinatorial mathematics*. [S.l.]: Chapman and Hall/CRC, 2017.

RUBIO-LARGO, Á.; VEGA-RODRÍGUEZ, M. A.; GONZÁLEZ-ÁLVAREZ, D. L. A hybrid multiobjective memetic metaheuristic for multiple sequence alignment. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 20, n. 4, p. 499–514, 2015.

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.

SAITOU, N.; NEI, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, v. 4, n. 4, p. 406–425, 1987.

SAXENA, A.; SHEKHAWAT, S.; SHARMA, A.; SHARMA, H.; KUMAR, R. Chaotic step length artificial bee colony algorithms for protein structure prediction. *Journal of Interdisciplinary Mathematics*, Taylor & Francis, v. 23, n. 2, p. 617–629, 2020.

SCHMOLLINGER, M.; NIESELT, K.; KAUFMANN, M.; MORGENSTERN, B. Dialign p: fast pair-wise and multiple sequence alignment using parallel processors. *BMC bioinformatics*, BioMed Central, v. 5, n. 1, p. 128, 2004.

SCHWARTZ, A. S.; PACHTER, L. Multiple alignment by sequence annealing. *Bioinformatics*, Oxford University Press, v. 23, n. 2, p. e24–e29, 2007.

SCHWEFEL, H.-P.; RUDOLPH, G. Contemporary evolution strategies. In: SPRINGER. *European conference on artificial life*. [S.l.], 1995. p. 891–907.

SIEVERS, F.; HIGGINS, D. G. Clustal omega, accurate alignment of very large numbers of sequences. In: *Multiple sequence alignment methods*. [S.l.]: Springer, 2014. p. 105–116.

SIEVERS, F.; WILM, A.; DINEEN, D.; GIBSON, T. J.; KARPLUS, K.; LI, W.; LOPEZ, R.; MCWILLIAM, H.; REMMERT, M.; SÖDING, J. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, John Wiley & Sons, Ltd, v. 7, n. 1, 2011.

SIMOSSIS, V.; KLEINJUNG, J.; HERINGA, J. An overview of multiple sequence alignment. *Current protocols in bioinformatics*, Wiley Online Library, v. 3, n. 1, p. 3–7, 2003.

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.

SUBRAMANIAN, A. R.; KAUFMANN, M.; MORGENSTERN, B. Dialign-tx: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology*, BioMed Central, v. 3, n. 1, p. 6, 2008.

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.

WILLIAMS, G. *Chaos theory tamed*. [S.l.]: CRC Press, 1997.

XU, C.; DUAN, H.; LIU, F. Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning. *Aerospace Science and Technology*, Elsevier, v. 14, n. 8, p. 535–541, 2010.

YAO, D.; JIANG, M.; YOU, X.; ABULIZI, A.; HOU, R. An algorithm of multiple sequence alignment based on consensus sequence searched by simulated annealing and star alignment. In: IEEE. *2015 International Symposium on Bioelectronics and Bioinformatics (ISBB)*. [S.l.], 2015. p. 3–6.

ZAFALON, G. F. D. *Algoritmos de alinhamento múltiplo e técnicas de otimizaçao para esses algoritmos utilizando Ant Colony*. Tese (Doutorado) — Universidade Estadual Paulista (UNESP), R. Cristóvão Colombo, 2265 - São José do Rio Preto - SP, Brasil, 4 2009.

ZAHA, A.; FERREIRA, H. B.; PASSAGLIA, L. M. *Biologia Molecular Básica-5*. [S.l.]: Artmed Editora, 2014.

ZAWBAA, H. M.; EMARY, E.; GROSAN, C. Feature selection via chaotic antlion optimization. *PloS one*, Public Library of Science, v. 11, n. 3, p. e0150652, 2016.

ZHU, H.; HE, Z.; JIA, Y. A novel approach to multiple sequence alignment using multiobjective evolutionary algorithm based on decomposition. *IEEE journal of biomedical and health informatics*, IEEE, v. 20, n. 2, p. 717–727, 2015.

ZOU, Q.; HU, Q.; GUO, M.; WANG, G. Halign: Fast multiple similar dna/rna sequence alignment based on the centre star strategy. *Bioinformatics*, Oxford University Press, v. 31, n. 15, p. 2475–2481, 2015.