Geraldo Francisco Donegá Zafalon

Algoritmos de alinhamento múltiplo e técnicas de otimização para esses algoritmos utilizando Ant Colony

São José do Rio Preto - SP, Brasil

2009

Geraldo Francisco Donegá Zafalon

Algoritmos de alinhamento múltiplo e técnicas de otimização para esses algoritmos utilizando Ant Colony

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, área de concentração em Computação Científica 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. José Márcio Machado

Programa de Pós-Graduação em Ciência da Computação
Departamento de Ciências de Computação e Estatística
Universidade Estadual Paulista "Júlio de Mesquita Filho"

São José do Rio Preto - SP, Brasil

2009

Geraldo Francisco Donegá Zafalon

Algoritmos de alinhamento múltiplo e técnicas de otimização para esses algoritmos utilizando Ant Colony

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, área de concentração em Computação Científica 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.

BANCA EXAMINADORA

Prof. Dr. José Márcio Machado
UNESP - São José do Rio Preto
Orientador

Profa. Dra. Liria Matsumoto Sato
Universidade de São Paulo

Profa. Dra. Renata Spolon Lobato
UNESP - São José do Rio Preto

São José do Rio Preto, Abril de 2009

Eis como deveis rezar: PAI NOSSO que estais no céu, santificado seja o vosso nome; venha a nós o vosso Reino; seja feita a vossa vontade, assim na terra como no céu. O pão nosso de cada dia nos dai hoje; perdoai-nos as nossas ofensas, assim como nós perdoamos aos que nos ofenderam; e não nos deixeis cair em tentação, mas livrai-nos do mal.

Mt 6, 9-13

Dedico este trabalho ao Senhor Jesus Cristo, a minha Noiva Érica, a minha Mãe Maria Aparecida e ao meu Pai Geraldo.

Agradecimentos

Agradeço primeiramente à Deus e ao meu amado Jesus Cristo por terem me permitido chegar até aqui e guiado-me em todos os momentos da minha vida. Agradeço pela intercessão contínua, junto a Jesus, a Nossa Senhora Aparecida, Mãe Rainha e Santo Expedito.

Agradeço a minha Noiva Érica (Neguinha) por todos os momentos juntos, pelo amor, pelo carinho, pela compreensão, por todos os momentos de superação das dificuldades, até quando eu sou meio chatinho. Aos meu pais, Maria Aparecida e Geraldo, por todo o amor, carinho, presença e suporte durante todo os dias, desde que eu nasci e, principalmente, nos meus estudos até agora.

Aos meus familiares que também estiveram presentes nessa caminhada, em especial, minha tia Madalena, meu tio Adalberto, minha prima Vivian e meu primo Rafael.

Aos meus amigos, que me acompanharam durante todo o tempo e proporcionaram e proporcionam muitos bons momentos: Raphael Fagliari, Marcos Dutra, Raul Strombeck, Daniel Franco Pedro, Álvaro Guilhermith, Marcelo Della Torre, Fábio Vitoriano Fernandes, Rafael Fabri e José Leôncio. Aos meus amigos da Unesp: André Ferrizzi, Antônio Carlos Fernandes, Carlos Eduardo Bonalumi, Jorge Luiz Corrêa, Leandro Rincon, Toni Jardini, José Nelson Falavinha, Otávio Kozyrski, Willian Lima, e especial, ao amigo Evandro Augusto Marucci, por todos os anos de convívio, trabalho, parceria e amizade na Universidade.

Ao Prof. Dr. José Márcio Machado, por todas as orientações em momentos de dúvidas, todas as boas conversas nos almoços e todos os encaminhamentos científicos e pessoais, e se Deus quiser que essa parceria continue em meu doutorado. Também agradeço

à FRUBESPESQ (Fundação Rubão de Estímulo à Pesquisa) por todos os auxílios sempre.

Ao Prof. Dr. Aleardo Manacero Júnior, pela amizade, orientações, pelas corridas e pela contribuição ao meu crescimento pessoal e acadêmico. À prof. Dra. Renata Spolon Lobato, também pela amizade, orientações e participação constante em meu crescimento, também pessoal e acadêmico.

Ao Prof. Dr. Angelo Passaro e a todo o pessoal do IEAv/CTA (Instituto de Estudos Avançados / Comando-Geral de Tecnologia Aeroespacial) em São José dos Campos, pelo auxílio e parceria durante o projeto, cedendo o uso do *cluster* para os testes e pela grande amizade adquirida.

À Prof. Dra. Liria Matsumoto Sato pela enorme contribuição técnica em minha defesa e pelos preciosos conselhos passados a mim.

Ao pessoal do Laboratório de Estudos Genômicos (LEGO), Prof. Dra. Paula Rahal, às Meninas Super-Poderosas da Paula Rahal (Ana Carolina Jardim, Lilian Yamasaki, Cintia Bittar), ao Prof. Dr. José Geraldo Nery (Gerrard) por todas as dicas e amizade, e em especial, ao Prof. Dr. Paulo Peitl Júnior e ao Dr. Sérgio Morais Aoki, pelas conversas, amizade, dicas, que ficarão para sempre.

Às secretárias do Departamento de Ciências de Computação e Estatística (DCCE), a Olga Maria, a Carina e a Esther, por todas as ajudas sempre que precisei e pela amizade.

À Unesp/Ibilce pelos incentivos financeiros e físicos durante todo o período de trabalho e aos seus funcionários pela prestatividade sempre.

Resumo

A biologia, como uma ciência bastante desenvolvida, foi dividida em diversas áreas, dentre elas, a genética. Esta área passou a crescer em importância nos últimos cinquenta anos devido aos inúmeros benefícios que ela pode trazer, principalmente, aos seres humanos. Como a genética passou a apresentar problemas com grande complexidade de resolução, estratégias computacionais foram agregadas a ela, surgindo assim a bioinformática.

A bioinformática desenvolveu-se de forma bastante significativa nos últimos anos e esse desenvolvimento vem se acentuando a cada dia, devido ao aumento da complexidade dos problemas genômicos propostos pelos biólogos. Assim, os cientistas da computação têm se empenhado no desenvolvimento de novas técnicas computacionais para os biólogos, principalmente no que diz respeito às estratégias para alinhamentos múltiplos de sequências.

Quando as sequências estão alinhadas, os biólogos podem realizar mais inferências sobre elas, principalmente no reconhecimento de padrões que é uma outra área interessante da bioinformática. Através do reconhecimento de padrões, os biólogos podem identificar pontos de alta significância (*hot spots*) entre as sequências e, consequentemente, pesquisar curas para doenças, melhoramentos genéticos na agricultura, entre outras possibilidades.

Este trabalho traz o desenvolvimento e a comparação entre duas técnicas computacionais para o alinhamento múltiplo de sequências. Uma é baseada na técnica de alinhamento múltiplo de sequências progressivas pura e a outra, é uma técnica de alinhamento múltiplo de sequências otimizada a partir da heurística de colônia de formigas. Ambas as técnicas adotam em algumas de suas fases estratégias de paralelismo, focando na redução do tempo de execução dos algoritmos.

Os testes de desempenho e qualidade dos alinhamentos que foram conduzidos com as duas estratégias mostraram que a abordagem otimizada apresenta melhores resultados quando comparada com a abordagem puramente progressiva.

Abstract

Biology as an enough developed science was divided in some areas, and genetics is one of them. This area has improved its relevance in last fifty years due to the several benefits that it can mainly bring to the humans. As genetics starts to show problems with hard resolution complexity, computational strategies were aggregated to it, leading to the start of the bioinformatics.

The bioinformatics has been developed in a significant way in the last years and this development is accentuating everyday due to the increase of the complexity of the genomic problems proposed by biologists. Thus, the computer scientists have committed in the development of new computational techniques to the biologists, mainly related to the strategies to multiple sequence alignments.

When the sequences are aligned, the biologists can do more inferences about them mainly in the pattern recognition that is another interesting area of the bioinformatics. Through the pattern recognition, the biologists can find hot spots among the sequences and consequently contribute for the cure of diseases, genetics improvements in the agriculture and many other possibilities.

This work brings the development and the comparison between two computational techniques for the multiple sequence alignments. One is based on the pure progressive multiple sequence alignment technique and the other one is an optimized multiple sequence alignment technique based on the ant colony heuristics. Both techniques take on some of its stages of parallel strategies, focusing on reducing the execution time of algorithms.

Performance and quality tests of the alignments were conducted with both strategies and showed that the optimized approach presents better results when it is compared with the pure progressive approach.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Considerações Iniciais

A biologia, por ser uma ciência bastante antiga, teve os seus inúmeros problemas, nas suas mais diversas sub-áreas, resolvidos ao longo dos anos. Vários desses problemas estavam contidos no campo da genética. Há muitos anos, desde o descobrimento da estrutura do DNA (*deoxyribonucleic acid* ou ácido desoxirribonucleico, em português), que os estudos relativos à genética têm despertado muito interesse por parte dos biólogos.

O crescimento desse interesse fez com que pesquisadores de outras áreas, que pudessem se associar à biologia, também passassem a pesquisar sobre os temas inerentes à genética. Dessa forma, uma das áreas que se associou à biologia foi a ciência da computação. Essa junção proporcionou o desenvolvimento da bioinformática.

A bioinformática é uma área relativamente nova, que passou a crescer bastante nos últimos 15 anos. Desde então, ela encontra-se em constante desenvolvimento, trazendo inovações a cada dia e propondo novos desafios aos pesquisadores [4].

Com o seu surgimento, a bioinformática tem despertado profundo interesse de muitos cientistas da computação e de pesquisadores de áreas afins à ciência da computação [5]. Esse interesse deve-se, principalmente, ao fato dos biólogos necessitarem do poder de processamento dos computadores para a análise de dados obtidos através das pesquisas experimentais, procurando alinhar e encontrar padrões entre diferentes sequências de nucleotídeos ou amino-ácidos. Os padrões podem conter informações muito importantes para os biólogos, por exemplo, podem ser utilizados para identificar a cura para uma

determinada doença, através da identificação de chamados *hot spots* ("pontos quentes") [6], pontos que contém informações relevantes para um determinado genótipo.

A análise das biossequências torna-se extremamente custosa e proibitiva computacionalmente quando se analisa mais de três sequências ao mesmo tempo [7]. Os algoritmos de programação dinâmica executam de forma eficiente para duas sequências [8]. Assim, os algoritmos de alinhamento múltiplo passaram a se destacar como a alternativa viável para esse problema [7].

No entanto, com a evolução dos problemas encontrados pelos pesquisadores, a complexidade desses problemas também aumentou. Com isso, o número de sequências a serem analisadas também cresceu, passando de algumas dezenas, para milhares de sequências. Desse forma, necessitou-se de mais poder computacional para fazer essas análises, visto que o computador sequencial até então utilizado não supria mais as necessidades.

Com a demanda por mais poder computacional, necessitou-se do uso de computação paralela e distribuída para melhorar o desempenho da execução das tarefas de bioinformática. Começa-se a usar, dessa forma, computação de alto desempenho para resolver problemas de bioinformática. A junção entre computação de alto desempenho e bioinformática trouxe inúmeros avanços e benefícios para ambas as áreas.

Mesmo com o aumento da complexidade dos problemas biológicos, a computação de alto desempenho não consegue resolver todos os problemas de bioinformática sozinha. Existe a necessidade da busca de algoritmos mais elaborados e, muitas vezes, desenvolvidos para um problema biológico em particular. Agregadas a esses algoritmos refinados, aplicam-se algumas técnicas necessárias para realizar uma otimização em sua execução. Essas técnicas são chamadas de métodos heurísticos, ou simplesmente, heurísticas.

Como os algoritmos determinísticos não eram mais suficientes, as heurísticas trouxeram uma abordagem estocástica aos problemas de bioinfomática, gerando soluções com um determinado percentual de precisão ou refinamento. Esses dois fatores, ou proximidade do resultado ótimo, podem ser ajustados conforme a técnica de otimização utilizada.

Assim, este trabalho mostra a implementação de um algoritmo de alinhamento múltiplo progressivo, bem como uma versão otimizada desse algoritmo, utilizando estimativas de pontuação e colônias de formigas.

1.2 Organização dos Capítulos

Esse trabalho organiza-se da seguinte maneira: no capítulo 1 tem-se uma introdução sobre o estudo realizado no presente trabalho, no capítulo 2 tem-se toda a fundamentação teórica necessária para o entendimento do trabalho, no capítulo 3 apresenta-se toda a fase de desenvolvimento do projeto e de seus algoritmos. No capítulo 4 são apresentados os resultados obtidos com as execuções dos algoritmos e, finalmente, o capítulo 5 traz as conclusões sobre o presente trabalho.

2 Fundamentação Teórica

2.1 Contexto Biológico

2.1.1 Organização Estrutural da Célula

O estudo do mundo vivo mostra que a evolução produziu uma imensa variedade de formas. Por volta de quatro milhões de espécies foram identificadas entre bactérias, protozoários, vegetais e animais, que diferem em sua morfologia, funcão e comportamento [1]. Para um conhecimento mais aprofundado dos organismos vivos, realiza-se um estudo celular e molecular nesses organismos. Esse estudo permite criar um plano organizado para entender como é o funcionamento desses organismos. Assim, conseguiu-se a divisão em dois grupos, os procariontes e os eucariontes.

Os procariontes são organismos cujas células não possuem envoltório nuclear. Os eucariontes possuem esse envoltório, apresentando um núcleo celular bem definido. A figura 1 mostra a complexidade de uma célula animal.

É dentro da célula, em especial em sua região nuclear (região mais centralizada da célula), que se encontra o material genético, motivo de estudo e de muita pesquisa por parte dos biólogos.

Ainda dentro da célula são realizados todos os mecanismos importantes para síntese do RNA e proteínas, que também são motivos de muitos estudos, juntamente com o DNA, por parte dos biólogos e, com advento da bioinfomática, também pelos bioinformatas.

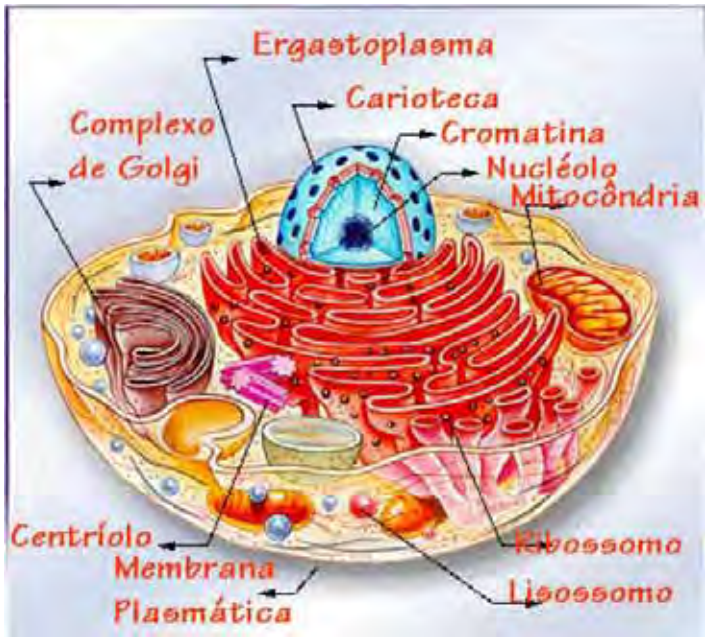

Figura 1: Célula Animal [1].

2.1.2 Macromoléculas Biológicas

Existem três macromoléculas biológicas que são primordiais aos seres humanos, o DNA, o RNA e as proteínas. Elas são cadeias de moléculas orgânicas e podem possuir tamanhos bastante variáveis. Uma macromolécula pode ser codificada por uma cadeia em um alfabeto de comprimento quatro para o DNA e o RNA (compostos por quatro bases), ou comprimento vinte para as proteínas (compostos por combinações de vinte aminoácidos) [9].

As cadeias para as moléculas de DNA/RNA são chamadas de sequências de nucleotídeos e cada elemento nessa sequência é chamado de base. Analogamente, as cadeias que representam as proteínas são chamadas de sequências de proteínas, e cada elemento dessa sequência é chamado de aminoácido. Tanto as sequências de nucleotídeos, como as sequências de proteínas são chamadas de biossequências ou, simplesmente, de sequências[10].

Todas as informações dos seres humanos são armazenadas em seu DNA, em português ADN (Ácido Desoxirribonucléico), tornando-se as características genotípicas de cada ser humano [11]. Estas informações são *transcritas* para as moléculas de RNA, cujas sequências de núcleotídeos contêm o código para a ordenação específica de aminoácidos.

As proteínas são então sintetizadas num processo que envolve a *tradução* do RNA, como pode ser visto na figura 2.

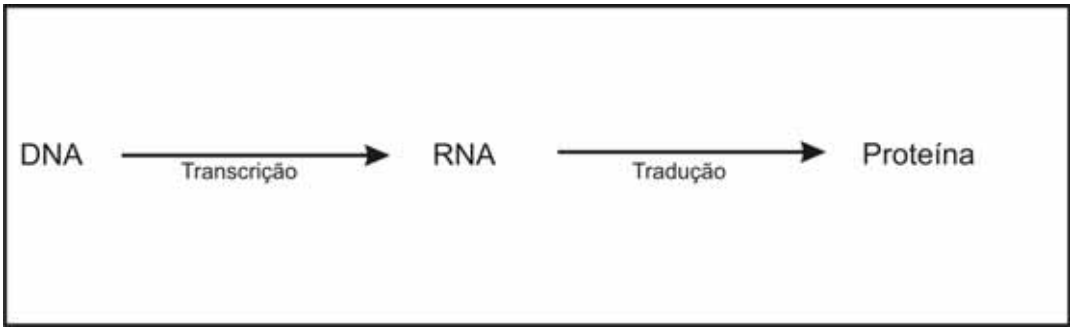

Figura 2: Transcrição e Tradução [1].

A tabela 1 mostra todos os aminácidos que são codificados para a formação das proteínas.

Tabela 1: Os vinte aminoácidos que são codificados [1].

Nome	**Símbolo**	**Abreviação**
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

Existem mais três conjuntos de aminoácidos que também são utilizados pelos pesquisadores, porém eles não são classificados no conjunto dos vinte principais, pois são combinações de outros aminoácidos. Esse três conjuntos podem ser vistos na tabela 2.

Tabela 2: Conjuntos extras de aminoácidos [1].

Nome	**Símbolo**	**Abreviação**
Asparagina/Ácido Aspartâmico	Glx	Z
Glutamina/Ácido Glutâmico	Asx	B
Qualquer aminoácido		X

A tradução que é realizada a partir do RNA para a formação dos aminoácidos ocorre com a junção de uma trinca de núcleotídeos do RNA. Essa trinca passa a se chamar *códon* e cada um desses *códons* da origem a um determinado aminoácido. Um mesmo aminoácido pode ser codificado por *códons* distintos, mas são apenas vinte os aminoácidos que compõe as proteínas. A partir disso, as proteínas que são conjuntos de aminoácidos, em sua essência, na verdade, são conjuntos de *códons*.

O conjunto composto pelas fases mostradas na figura 2 é chamado de dogma central da biologia molecular.

O DNA é composto por quatro bases, chamadas de bases nitrogenadas, devido à presença do nitrogênio em sua composição. As quatro bases são: Adenina (A), Timina (T), Citosina (C) e Guanina (G). Tem a forma de uma estrutural helicoidal, com duas fitas, também chamadas de hélices, interligadas pelas bases nitrogenadas. A relação entre essas quatro bases para sua ligação é mostrada na figura 3.

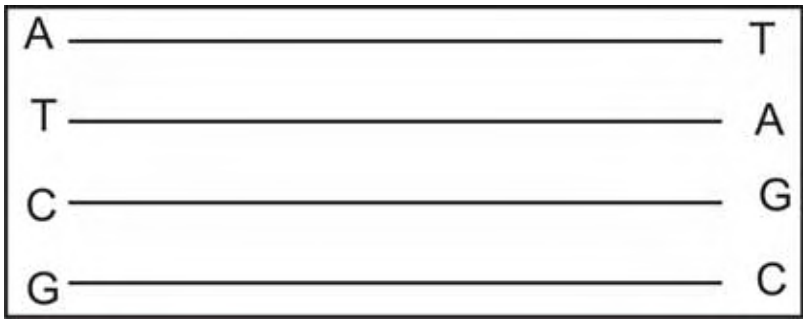

Figura 3: Bases que compõe o DNA.

A figura 4 mostra a forma estrutural do DNA descoberta por Watson e Crick em meados dos anos 50.

O RNA trata-se de uma fita simples, formada também por quatro bases. A diferença

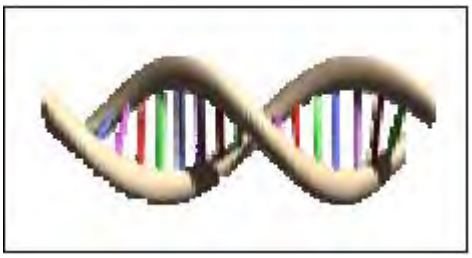

Figura 4: Estrutura Dupla Hélice do DNA [1].

em relação ao DNA, é que quando o RNA é transcrito, nas posições em que estão as Timinas (T), entra a Uracila (U).

2.1.3 Filogenia

Dentro do estudo da evolução das espécies de seres vivos, torna-se necessário procurar por homologias entres as epécies envolvidas [12]. Homologias tratam-se de similaridades resultantes da herança de um ancestral comum.

A filogenia é o estudo da evolução dos seres vivos. O seu objetivo é encontrar uma árvore evolucionária dos seres vivos, partindo do pressuposto que as diferentes espécies derivam de um ancestral comum.

Os primeiros estudos envolvendo filogenia baseavam-se na observação do fenótipo das espécies. Porém, esse método pode levar a erros, visto que duas espécies sem um ancestral comum podem possuir características em comum [12]. Assim, essa observação classificaria, provavelmente, as espécies de forma incorreta.

Biologicamente, o estudo baseado no uso de genótipos faz muito mais sentido, assim a análise gênica dos seres vivos é muito mais relevante. Com o atual desenvolvimento do sequenciamento de genomas inteiros, o DNA completo de espécies está ficando disponível em bases de dados. Devido a essa grande quantidade de informação gerada é possível abandonar o estudo da filogenia por fenótipo e começar a usar o genótipo dos seres.

No entanto, isso traz consigo um fator de complicação, pois a quantidade de informação a ser analisada quando se trata do estudo dos genes dos seres vivos é imensamente

maior se comparada a quantidade de informações fenotípicas. Com isso, o uso da computação para a realização do alinhamento das biossequências passou a ter um papel muito mais importante e tornou-se, na verdade, um artefato fundamental na busca pela identificação de espécies e de suas caracaterístcas gênicas.

2.1.4 Padrões em Biossequências

Os padrões geralmente correspondem a elementos importantes para a estrutura e função das sequências de DNA e proteínas. Existem suposições de que estas regiões são melhor conservadas durante a evolução porque são importantes para a estrutura ou função da molécula [9].

A descoberta de tais padrões pode ajudar no entendimento dos relacionamentos entre as biossequências, na estrutura e função das proteínas e além disso, na interpretação da atividade dos organismos vivos. Assim, um dos objetivos de descoberta de padrões em biologia é o de classificação, por exemplo nas famílias de proteínas. Os padrões são considerados padrões classificadores, se e somente se, dada uma proteína desconhecida, ela irá ser classificada como membro de uma família, se tiver os padrões daquela família [13].

Uma outra vertente, além da classificação, é a da descoberta de padrões significativos. Novamente, no contexto das proteínas, é interessante, por exemplo, encontrar elementos estruturais ou funcionais importantes, independente deles terem especificidade suficiente para distinguir uma família da outra.

Geralmente, define-se uma classe de padrões que se deseja procurar e descobre-se o padrão com maior pontuação que tem aprovação suficiente. A aprovação baseia-se no número de vezes que o padrão ocorre. No entanto, algumas vezes os padrões longos com poucas ocorrências são mais importantes que os padrões pequenos com muitas ocorrências.

Abordando a descoberta de padrões em biossequências através da aplicação de técnicas algorítmicas, essas técnicas podem encontrar o melhor padrão (maior pontuação),

encontrar os melhores padrões (todos os que tiveram pontuações altas) ou encontrar todos os padrões com algum nível pré-definido de aprovação e pontuação. Alguns algoritmos podem até usar informações adicionais para auxiliar na identificação dos padrões, como por exemplo, a estrutura secundária ou terciária das sequências.

2.2 Alinhamentos de Biossequências

Não se trata de uma tarefa simples quando os biólogos necessitam realizar avaliações sobre os genes e sobre características das proteínas de uma determinada espécie.

O uso do alinhamento para poder tirar conclusões sobre as sequências foi de grande valia para os biólogos, devido à necessidade de avaliar grandes conjuntos de DNA/RNA ou proteínas, ambos de grande diversidade.

O alinhamento dessas biossequências tornaria-se inviável se fosse feito manualmente devido à grande quantidade de genes e de suas combinações possíveis. O surgimento da bioinfomática, trazendo consigo a habilidade dos cientistas da computação em desenvolver algoritmos, facilitou, de forma considerável, o trabalho dos biólogos com relação à realização desses alinhamentos.

Um alinhamento de sequências de DNA/RNA ou proteínas de diferentes espécies de seres vivos é uma hipótese de homologia entre as bases que constituem os genes (ou proteínas) sendo comparados naqueles seres vivos. Alinhamentos podem ser tratados como modelos que podem ser usados para testar hipóteses evolucionárias e, portanto, são importantes para estudos de filogenia [12].

Alinhamento de sequências, simplificadamente, é o posicionamento e a comparação entre cadeias de caracteres, elaborando inferências que podem ser feitas a partir das partes similares que elas apresentam.

Conseguir um bom alinhamento entre as sequências de várias espécies é geralmente uma tarefa difícil e envolve vários problemas. Por exemplo, a diferença de tamanho entre

os diversos trechos que apresentam similariadade, os tamanhos variados das sequências, entre outros problemas que podem ser encontrados [12].

2.2.1 Algoritmo de Programação Dinâmica

A tarefa de alinhar duas sequências de DNA geralmente é baseada na técnica do algoritmo de programação dinâmica. Esta técnica busca o melhor alinhamento entre duas cadeias de caracteres [14].

Basicamente, o que a técnica faz é a construção de uma matriz de comparação de prefixos das duas sequências a serem alinhadas. No momento do alinhamento sucessivo dos prefixos, o algoritmo atribui um *score* (valor de pontuação) para cada um dos prefixos. Esse valor de pontuação é computado com o intuito de penalizar as diferenças entre os prefixos e privilegiar as similaridades.

Diversos casos contam com sequências de comprimentos distintos. Dessa forma, o algoritmo de programação dinâmica insere *gaps* (espaços) nas sequências para fazer com que as duas fiquem com o mesmo comprimento após o alinhamento.

Para ilustrar o funcionamento do algoritmo de programação dinâmica, considere as duas sequências de DNA seguintes: $GACGCATTAG$ e $GATCGGAATAG$. Um possível alinhamento para essas sequências é o mostrado na figura 5.

Figura 5: Sequências de DNA alinhadas.

Note que as sequências são bem similares. As únicas diferenças são: um nucleotídeo (T) a mais na segunda sequência. Assim, na primeira sequência houve a necessidade da inserção de um *gap* para deixá-la com o mesmo comprimento da segunda. A outra diferença é uma troca de A por T na quarta posição da direita para a esquerda.

Quando existe a coincidência, ou similaridade, entre as bases nas mesmas posições nas

diferentes sequências, ocorre um *match*. Contrariamente, quando na mesma posição, nas diferentes sequências não ocorre a similaridade entre as bases, ou nas sequências existe a inserção de *gap* naquela mesma posição, tem-se um *mismatch*.

Para o cálculo da pontuação do alinhamento, *matches* recebem pontuação positiva, *mismatches* e *gaps* recebem pontuação negativa. Como exemplo, a seguir encontra-se o cálculo da pontuação para o exemplo da figura 5. Os *matches* recebem o valor +2, os *gaps* e os *mismatches* recebem o valores -2.

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

Neste caso, tem-se 9 *matches*, 1 *gap* e 1 *mismatch*. A pontuação calculada para esse alinhamento obteve o valor final de 14.

Vários alinhamentos são possíveis com as sequências mostradas na figura 5. O alinhamento ótimo é aquele que atinge o maior valor no cálculo da pontução, podendo existir mais de um alinhamento ótimo, dependendo de como são rearrajandos os nucleotídeos.

Várias extensões do algoritmo de programação dinâmica podem ser criadas, visto que estas introduzem conceitos mais próximos da realidade biológica. Por exemplo, quando necessita-se inserir vários *gaps*, esses representando as mutações, a chance de mutações ocorrerem sequencialmente no DNA são bem maiores do que elas acontecerem espalhadas por várias partes da fita. Assim, o algoritmo prefere, na tentativa de atingir uma maior pontuação possível, inserir os *gaps* em sequência e não separados por nucleotídeos.

A busca por similaridades nas sequências pode ser tratada de forma global ou local, dependendo da necessidade e da abordagem de cada análise. Assim, a figura 6 mostra um exemplo de um alinhamento global e um local para duas sequências de DNA, utilizando um algoritmo de programação dinâmica para realizá-lo.

Na figura 6, encontra-se à esquerda um alinhamento global para esse par de sequências. À direita estão demonstrados alguns alinhamentos locais para essas sequências.

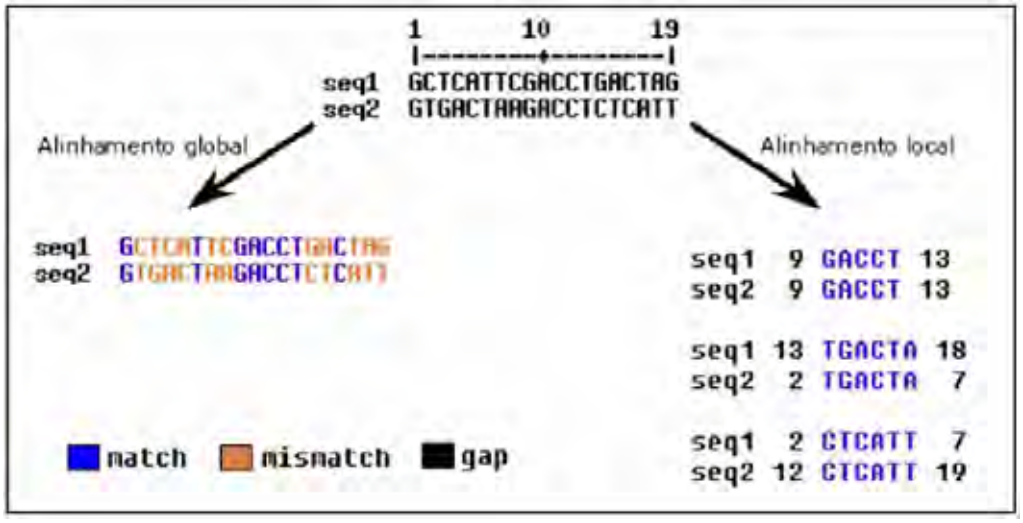

Figura 6: Alinhamento Global e Local.

Os alinhamentos globais focam em uma análise das sequências como um todo, independente se essas sequências sejam de nucleotídeos ou de aminoácidos, procurando por um máximo de similaridade entre elas. Com isso, o alinhamento global tenta identificar sequências muito parecidas, não se preocupando com pontos em particular dessas sequências.

O alinhamento local, contrariamente ao alinhamento global, preocupa-se em encontrar pontos específicos de similaridade dentro das sequências. Esses pontos podem representar características que serão úteis para as análises feitas pelos biólogos. Por exemplo, em um genoma de um vírus, se os pesquisadores tiverem conhecimento de uma região em particular que correponda a sua capacidade de mutação, eles podem tentar desenvolver um determinado inibidor para esse vírus. A partir do momento em que essas regiões possuem algum sentido biológico de fato, elas passam a ser classificadas como *hot spots* (pontos quentes).

2.2.2 Alinhamentos de Sequências de Aminoácidos

Conforme mostrado anteriormente o algoritmo de programação dinâmica consegue realizar os alinhamentos entre duas sequências de nucleotídeos de forma bastante eficiente com as ponderações que ele estabelece entre os *matches*, *mismatches* e *gaps*. No entanto,

quando se realiza um alinhamento entre sequências de aminoácidos, uma análise mais cuidadosa deve ser feita, levando-se em consideração características da evolução entre esses aminoácidos [15].

Focados nessa necessidade de considerar características evolucionárias dos aminoácidos, os biólogos desenvolveram tipos de matrizes triagulares de tamanho 20x20, referentes à quantidade de aminoácidos, que provêm os pesos necessários para se comparar aminoácidos idênticos e diferentes, bem como atribuirem pesos aos *gaps* [15].

Os dois tipos tipos mais utilizados de matrizes são as matrizes PAM (*Percent Accepted Mutation*) e BLOSUM (*Blocks Substitution Matrix*), porém existem outras, como por exemplo a GONNET. Essas matrizes são chamadas de matrizes de substituição, pois elas refletem os pesos obtidos pela comparação das substituições dos aminoácidos durante a evolução [15].

Essas matrizes ainda podem ser classificadas por um número, como BLOSUM30, BLOSUM62, PAM250, PAM70, entre outros. Esses números identificam diferentes suavidades na distribuição dos pesos pelas matrizes [14]. As diferentes suavidades exploram características interessantes nas diferentes análises que podem ser feitas.

2.2.3 Matriz BLOSUM

A matriz BLOSUM é baseada em alinhamentos locais. O foco principal dessa matriz é varrer os blocos para cada região conservada das famílias de proteínas e, dessa maneira, contar a frequência relativa dos aminoácidos e as suas probabilidades de substituição.

Matrizes BLOSUM com números altos são projetadas para comparar sequências muito relacionadas (ser humano e o macaco), enquanto que matrizes BLOSUM com números baixos são projetadas para comparar sequências bastante distantes (ser humano e as baratas). Por exemplo, uma BLOSUM80 é utilizada para alinhamentos menos divergentes e uma BLOSUM45 é utilizada para alinhamentos bem mais divergentes.

O cálculo da matriz BLOSUM é feito através da equação:

$$S_{ij} = \left(\frac{1}{\lambda}\right) \log\left(\frac{p_{ij}}{q_i * q_j}\right)$$

Na equação, p_{ij} é a probabilidade de dois aminoácidos i e j substituirem um ao outro em sequências homólogas, e q_i e q_j são as probabilidades de fundo de encontrar os aminoácidos i e j em qualquer sequência de proteínas aleatórias. O fator λ é um fator de escala para facilitar a computação de valores inteiros [16]. A figura 7 mostra um exemplo de uma matriz BLOSUM62.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figura 7: Exemplo de uma matriz BLOSUM62.

2.2.4 Matriz PAM

Utilizada para pontuar alinhamentos de biossequências, a matriz PAM foi introduzida por Margaret Dayhoff em 1978 baseada em 1572 observações de mutações em 71 famílias relacionadas de proteínas.

As matrizes PAM utilizam um modelo de cadeia de Markov de mutações em proteínas. Essas matrizes são normalizadas, resultando que a PAM1 possui 1 ponto de mutação em cada 100 aminoácidos e é apropriada para pontuação de sequências que são muito similares. Matrizes PAM para a comparação de sequências de baixa similaridade são

calculadas a partir de repetidas multiplicações da matriz PAM1 por ela mesma [17]. Por exemplo, PAM250 é equivalente a 250 substituições em cada 100 aminoácidos. A figura 8 mostra um exemplo de uma matriz PAM70.

```
A   5
R  -4   8
N  -2  -3   6
D  -1  -6   3   6
C  -4  -5  -7  -9   9
Q  -2   0  -1   0  -9   7
E  -1  -5   0   3  -9   2   6
G   0  -6  -1  -1  -6  -4  -2   6
H  -4   0   1  -1  -5   2  -2  -6   8
I  -2  -3  -3  -5  -4  -5  -4  -6  -6   7
L  -4  -6  -5  -8 -10  -3  -6  -7  -4   1   6
K  -4   2   0  -2  -9  -1  -2  -5  -3  -4  -5   6
M  -3  -2  -5  -7  -9  -2  -4  -6  -6   1   2   0  10
F  -6  -7  -6 -10  -8  -9  -9  -7  -4   0  -1  -9  -2   8
P   0  -2  -3  -4  -5  -1  -3  -3  -2  -5  -5  -4  -5  -7   7
S   1  -1   1  -1  -1  -3  -2   0  -3  -4  -6  -2  -3  -4   0   5
T   1  -4   0  -2  -5  -3  -3  -3  -4  -1  -4  -1  -2  -6  -2   2   6
W  -9   0  -6 -10 -11  -8 -11 -10  -5  -9  -4  -7  -8  -2  -9  -3  -8  13
Y  -5  -7  -3  -7  -2  -8  -6  -9  -1  -4  -4  -7  -7   4  -9  -5  -4  -3   9
V  -1  -5  -5  -5  -4  -4  -4  -3  -4   3   0  -6   0  -5  -3  -3  -1 -10  -5   6
B  -1  -4   5   5  -8  -1   2  -1   0  -4  -6  -1  -6  -7  -4   0  -1  -7  -4  -5   5
Z  -1  -2  -1   2  -9   5   5  -3   1  -4  -4  -2  -3  -9  -2  -2  -3 -10  -7  -4   1   5
X  -2  -3  -2  -3  -6  -2  -3  -3  -3  -3  -4  -3  -3  -5  -3  -1  -2  -7  -5  -2  -2  -3  -3
* -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11   1
```

Figura 8: Exemplo de uma matriz PAM70.

2.2.5 Algoritmos para Análises de Sequências em Bases de Dados

Com a evolução do uso dos sistemas de comunicação nas décadas de 80 e 90 e do armazenamento distribuído, passou-se a difundir ainda mais o uso de bases de dados e, com isso, a melhoria dos sistemas para armazenamento das biossequências [18]. Dessa forma, as bases de dados de biossequências passaram a crescer de forma muito rápida e o seu tamanho começava a dificultar a análise dos dados nelas contidos, devido ao grande tempo necessário para as comparações.

Verificou-se que o uso do algoritmo de programação dinâmica para análise dessas biossequências armazenadas nas bases de dados era totalmente inviável devido à quantidade de sequências, o que torna essa análise um problema NP-Completo. Para isso algumas heurísticas foram desenvolvidas para auxiliar nessa busca e dentre as mais conhecidas estão as das famílias FAST [19, 20] e BLAST (*Basic Local Alignment Search Tool*) [21].

A família FAST começou a ser descrita em 1983 com um algoritmo que permitiu uma

busca rápida em bancos de dados de proteínas e nucleotídeos, focando-se somente em um grupo de identidades entre as duas sequências. Depois do FASTP que foi o primeiro a ser criado e utilizado para a comparação entre proteínas, várias modificações foram realizadas, surgindo, por exemplo, o FASTN para comparação entre nucleotídeos e o FASTA que utiliza as melhores características do FASTP e do FASTN, permitindo uma maior sensibilidade nas comparações.

Existem ainda mais algumas variantes como o LFASTA e o PFASTA. Enquanto que o FASTA retorna um único alinhamento local, o LFASTA mostra todas as regiões locais de similaridade compartilhadas entre as sequências que estão sendo comparadas. O PFASTA é idêntico ao LFASTA, só que apresenta os alinhamentos em uma forma matricial particular.

O algoritmo BLAST trata-se de uma técnica mais eficiente do que os da família FAST [22]. Basicamente, o método detecta similaridades fracas de sequências, mas que possuem significado biológico. Os programas da famílias BLAST foram feitos para se comparar sequências de consultas de DNA ou proteína com bancos de dados de DNA ou proteína em qualquer combinação.

Assim como os algoritmos da famílias FAST, os algoritmos da família BLAST também possuem variantes, como o BLASTP, que é utilizado para comparações entre proteínas e o BLASTN para comparações entre ácidos nucléicos.

2.2.6 Modelos de Markov Ocultos (*Hidden Markov Models*)

A variação nas famílias de sequências pode ser descrita estatisticamente e, realmente, é a base para a maioria dos métodos utilizados em análises de biossequências. Assim, tem-se os Modelos de Markov Ocultos (HMM), que são modelos estatísticos bastante aceitos para várias tarefas da biologia molecular [23].

O uso mais comum do HMM em biologia molecular se relaciona a criação de perfis probabilísticos das famílias de proteínas, o que comumente se chama de perfil HMM. Esse

perfil pode ser traçado através da varredura dos membros de uma família de proteínas [23]. Assim, o HMM é muito utilizado para a predição de genes e da busca por homologias [24].

Um HMM pode ser visto como uma variante de transdutores de estados finitos (FST) de probabilidades ou estocásticos. Em um FST, o autômato muda de estado de acordo com as entradas que são examinadas. Considerando um FST probabilístico, as transições são especificadas pelas probabilidades [15]. Assim, trata-se de um sistema não determinístico.

Os Modelos de Markov são motivos de muita pesquisa na área de bioinformática. Um ponto que passou a limitar o uso desses modelos, está ligado a sua capacidade de agregação com modelos de análises de sequências mais novos e poderosos, pois, geralmente, encontra-se dificuldade em se realizar essa agregação, devido à diversas características particulares dos HMMs [24].

2.2.7 Alinhamentos Múltiplos

As estratégias de alinhamentos múltiplos objetivam, da mesma maneira que o alinhamento simples, para duas sequências, a máxima pontuação, ou seja, o maior número de coincidências entre as sequências. Ele se difere do alinhamento simples, pois trata de um conjunto maior do que duas sequências simultaneamente.

Os alinhamentos utilizando algoritmos de programação dinâmica funcionam de forma relativamente eficiente para duas sequências. Sua complexidade para a comparação entre duas sequências é $O(n^2)$. Pode-se ainda generalizar o algoritmo para que ele resolva o alinhamento para um número n fixo de sequências, contudo, a complexidade exponencial de memória e processamento torna a aplicação do algoritmo impraticável [12]. Assim, ele funciona bem para um conjunto de duas sequências.

O problema de alinhar um número n qualquer de sequências através do algoritmo de programação dinâmica é um problema NP-Completo. Os problemas NP-Completos são problemas que não possuem solução conhecida que apresente uma resposta em um tempo

polinomial ou menor com relação ao tamanho da entrada de dados. Significa dizer que o tempo de processamento seria extremamente alto e impraticável.

Com o intuito de minimizar esse problema da complexidade, algumas heurísticas foram desenvolvidas para diminuir essa complexidade de tempo de alinhamentos com mais de duas sequências. Mais adiante serão vistas outras heurísticas que podem ser agregadas ao algoritmo de alinhamento múltiplo, permitindo uma maior otimização em sua execução.

Os algoritmos de alinhamento múltiplo têm a capacidade de trabalhar com mais de duas sequências e alinhá-las de forma eficiente em um tempo hábil. Existem algumas variantes de algoritmos de alinhameto múltiplo. No entanto, a mais utilizada é a de alinhamentos múltiplos progressivos.

Os algoritmos de alinhamento múltiplo nessa seção são tratados de forma sequencial. Na seção 2.3 será mostrado que se pode paralelizar esses algoritmos permitindo um ganho significativo de desempenho em sua execução, resultando em um tempo menor para a obtenção de resultados finais com relação às sequências analisadas.

2.2.8 Alinhamentos Múltiplos Progressivos

O alinhamento múltiplo progressivo é uma abordagem para o alinhamento de várias sequências que tenta inferir uma evolução entre as espécies sendo comparadas [25].

Quando se fala em alinhamento múltiplo progressivo, pode-se dividí-lo em três fases: o alinhamento par-a-par, a construção da árvore guia e o alinhamento múltiplo propriamente dito.

No alinhamento par-a-par, constrói-se uma matriz de distâncias, também chamada de matriz de pontuação, dos alinhamentos de todos os possíveis pares de conjuntos de sequências de entrada. A distância do alinhamento de cada par é obtida através do algoritmo de programação dinâmica. Considerando um conjunto de n sequências de entrada, o algoritmo de programação dinâmica será invocado $n(n-1)/2$ vezes. Assim, essa fase apresenta-se como a mais crítica com relação ao tempo de processamento.

Depois de obtida a matriz de distâncias, gera-se uma árvore guia, ou árvore filogenética, que conduzirá posteriormente o algoritmo múltiplo propriamente dito. Diversos métodos podem ser empregados para a construção dessa árvore. Um dos algoritmos mais utilizados é o *neighbor-joining*, pois geralmente apresenta os resultados biológicos mais coerentes. O *neighbor-joining* junta sucessivamente em uma árvore as sequências mais próximas em ramos mais próximos, iterativamente, até que a árvore guia contenha todas as sequências [26].

Com a árvore guia construída, por exemplo como a árvore mostrada na figura 9, o alinhamento é feito progressivamente, de modo que as sequências em ramos mais próximos sejam alinhadas primeiramente, por exemplo na figura 9, o humano e o macaco.

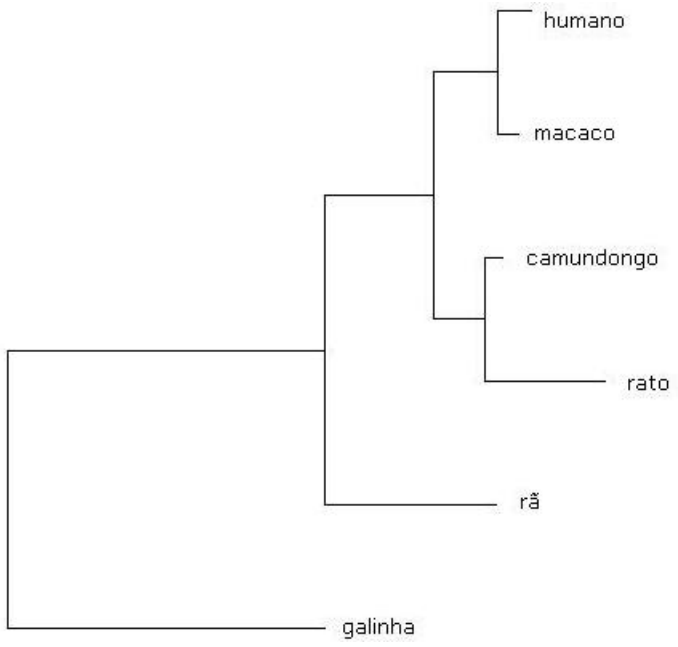

Figura 9: Ilustração de Árvore Guia ou Filogenética Construída.

No momento de alinhar uma sequência com um grupo de sequências já alinhadas, os *gaps* inseridos no grupo alinhado são preservados. Dessa mesma forma, o alinhamento é feito entre a sequência ainda não alinhada e a sequência do grupo alinhado que possua a menor distância em relação a ela [7], [12]. Todas as demais sequências do grupo alinhado são apenas ajustadas para acompanhar o alinhamento. Para um número n de sequências de entrada, são necessários $n - 1$ alinhamentos nessa fase. Dessa forma, o algoritmo de

programação dinâmica é invocado $n - 1$ vezes.

O alinhamento múltiplo progressivo sempre retorna uma resposta para um determinado alinhamento, independente se essa resposta terá sentido biológico ou não. Assim, em duas sequências com nenhuma relação biológica entre elas, o resultado também não terá nenhum sentido biológico.

O alinhamento progressivo não faz uma varredura exaustiva em todas as possibilidades de alinhamento. Assim, um alinhamento obtido pode possuir um mínimo local bem distante do mínimo global, ou seja, pode haver divergências muito grandes quando se busca a mesma característica em proporções distintas, como a local e a global. Mesmo com esse problema, as respostas obtidas pelo método progressivo são bastante boas e próximas da solução final, necessitando de pequenos ajustes [12].

2.3 Computação de Alto Desempenho

Os algoritmos de alinhamento múltiplo atuam paliativamente sobre o problema que afeta os algoritmos de programação dinâmica, que é o de alinhar mais de duas sequências em tempo hábil, pois trata-se de um problema NP-completo.

No entanto, mesmo os algoritmos de alinhamento múltiplo, pelo fato de sua execução ser de forma sequencial, quando o número de biossequências aumenta significativamente, eles também passam a ser um problema NP-Completo. Com isso, o uso de computação de alto desempenho para a execução dos algoritmos passou a ser visto como um fator de suma importância, considerando que o volume de dados a serem avaliados aumenta diariamente [27].

Utilizar a computação de alto desempenho para a execução das tarefas relativas ao problema de alinhamentos múltiplos para grande quantidades de sequências trouxe um ganho extremamente alto no tempo de obtenção dos resultados e permitiu que problemas ainda mais complexos pudessem ser solucionados.

A computação de alto desempenho está diretamente ligada ao uso de programação paralela e distribuída para a execução dos algoritmos. Dessa maneira, pode-se dividir a execução das tarefas em várias unidades de processamento, diminuindo significativamente o tempo total de processamento [28]. Assim, a utilização de sistemas computacionais multiprocessados passou a ser o ponto chave na resolução de problemas de bioinformática de alto grau de complexidade computacional.

Para se fazer uso da computação de alto desempenho, necessita-se de todo um aparato de *software* e *hardware* para suportar as aplicações paralelas. Necessita-se da interligação dos nós participantes (unidades de processamento) e da comunicação entre eles através de um ambiente de passagem de mensagem.

Não necessariamente os nós participantes possuem uma única unidade de processamento. Isso vai depender do tipo de arquitetura e do meio de comunicação adotados e também de detalhes de *software*, como por exemplo, a granularidade do algoritmo paralelo. A decisão por um determinado tipo de máquina para realizar as tarefas vai depender da necessidade, mas, principalmente, de questões financeiras.

Por exemplo, considerando o algoritmo de alinhamento múltiplo progressivo, sem nenhuma dúvida, a fase mais crítica da execução do algoritmo é o alinhamento par-a-par, pois demanda a maior carga de processamento. Assim, o algoritmo paralelo distribui vários alinhamentos par-a-par entre os vários processadores da máquina paralela. Na figura 10, ilustra-se como é a execução do alinhamento par-a-par.

Essa divisão de tarefas acontece em máquinas paralelas, podendo ser elas supercomputadores ou não, interligados por canais de comunicação.

Atualmente, devido ao alto custo dos supercomputadores, uma grande parte dos pesquisadores procura realizar computação de alto de desempenho com máquinas de arquiteturas convencionais. São computadores com arquitetura do tipo x86 (PCs), interligados por um elemento comutador (*switch*), comunicando-se através de um ambiente de passagem de mensagem e utilizando sistemas operacionais de código aberto (*open source*)

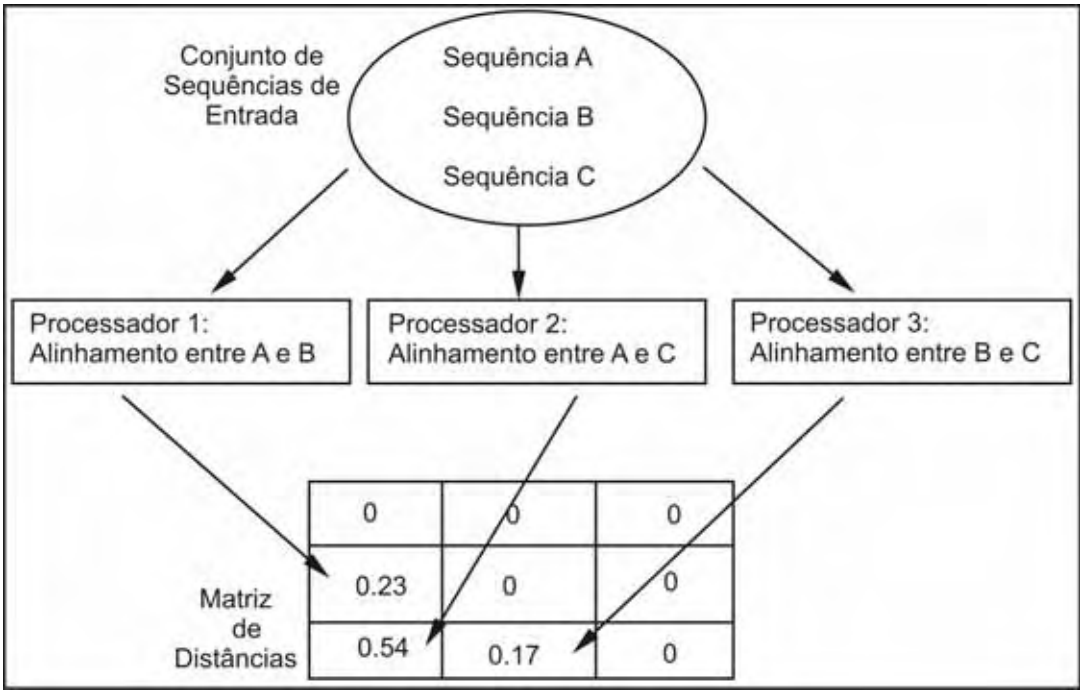

Figura 10: Esquema de alinhamento par-a-par processado em paralelo.

como, por exemplo, Linux, BSD ou Solaris. A esse tipo de máquina de alto desempenho deu-se o nome de *cluster Beowulf* [29].

2.3.1 *Clusters Beowulf*

Muitas pesquisas na área de bioinfomática, simulação de sistemas, física de partículas e várias outras áreas que possuem grande volume de dados a serem processados e exigem uma capacidade de processamento muito grande estão adotando a ideia do uso de *clusters Beowulf*.

A criação desse tipo de *cluster* foi feita por pesquisadores da NASA que possuiam poucos recursos financeiros e necessitavam realizar a análise de grandes quantidades de dados de exploração espacial. O nome *Beowulf* faz alusão a um herói da mitologia nórdica que era conhecido por ser um excelente guerreiro que acabava com todos os problemas. Nesse sentido, o cluster *Beowulf* foi criado para ser um solucionador de problemas também, só que agora matemáticos.

Basicamente, ele se organiza conforme mostrado na figura 11. Os computadores são

interconectados por um elemento comutador central (*switch*) e se comunicam através de ambientes de passagem de mensagem.

O nó denominado *Front-End* atua como um centralizador das tarefas. É a partir desse nó que partem os comandos para os demais nós participantes do processamento. O *Front-End* indica qual nó vai tratar qual processo e fica esperando a resposta desse nó. Em alguns casos, o próprio *Front-End* pode participar também do processamento, e não ficar apenas com a terefa de gerenciar.

Para elucidar um pouco mais como o algoritmo de alinhamento múltiplo progressivo pode ser executado em um *cluster*, tomando novamente por base a figura 10, a elipse onde estão as sequências A, B e C pode ser encarada como o *Front-End* que guarda as sequências e depois as envia para os nós por passagem de mensagem. Cada um dos retângulos representa um nó do *cluster*. Depois que cada nó executou a sua tarefa, a resposta é enviada ao *Front-End* que a armazena na matriz de distâncias.

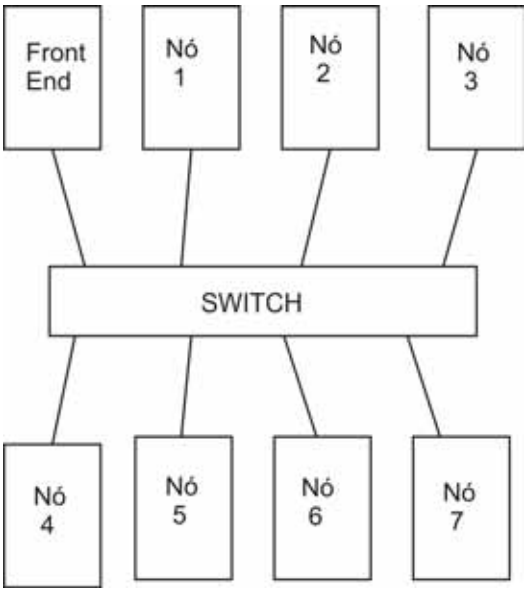

Figura 11: Esquematização de um *cluster Beowulf*.

Os *clusters Beowulf*, diferentemente dos computadores paralelos comerciais, são marcados por não possuirem um balanço equivalente entre a velocidade de comunicação e a velocidade de processamento, visto que esta é muito maior. Assim, os canais de comunicação podem gerar gargalos na execução da tarefa no *cluster*. Uma forma de melhorar o

desempenho na comunicação é com o uso de redes de comunicação *Gigabit - 1000 Mbits*, que transmitem a uma velocidade dez vezes maior do que a *Fast Ethernet - 100 Mbits*.

Outra estratégia que pode ser adotada para reduzir o impacto do gargalo da comunicação é explorar a granularidade do programa parelelo, escolhendo entre paralelismo em grão fino (*fine-grained*) ou grão grosso (*coarse-grained*) durante os processamentos.

2.3.2 Granularidade de Programas Paralelos

A granularidade dos programas paralelos é um grande fator a ser explorado durante a execução em um sistema multiprocessado como no caso dos *clusters Beowulf*.

Certamente, a granularidade não só pode ser explorada se em sistemas multiprocessados, mas também em máquinas vetoriais. Os computadores vetoriais vem perdendo um pouco de espaço no mercado, dando lugar aos computadores multiprocessados, visto que estes se tornaram muito mais poderosos computacionalmente, permitindo o uso de múltiplas linhas de execução simultâneas [30].

O tipo de granularidade mais indicada para um programa paralelo executado em computadores vetoriais é a granularidade fina, que nada mais é do que tarefas individuais que são relativamente pequenas em relação ao tamanho do código e da execução. Os dados são transferidos entre os processadores, frenquentemente, em montantes que variam de uma a poucas palavras de memória.

Os sistemas multiprocessados não só conseguiram vantagens sobre os computadores vetorias em termos de poder de processamento, mas, em algumas vezes, com relação ao custo. Isso mostra-se diretamente ligado ao surgimento dos *clusters Beowulf*, computação de alto desempenho a um baixo custo.

Os conjuntos de sistemas multiprocessados exploram as características de granularidade do paralelismo em grão grosso, visto que como existem vários elementos para realizar computações, os dados só são comunicados entre os processadores após grandes quantidades de processamento [31].

Quanto menor a granularidade, maior é o potencial para o paralelismo e o aumento da quantidade de operações, mas maiores serão as sobrecargas de sincronização e de comunicação.

2.3.3 Ambientes de Passagem de Mensagem

Os ambientes de passagem de mensagem são rotinas que permitem a comunição entre os elementos participantes do *cluster*, consequentemente, a comunicação entre os processos. Sem esses ambientes de passagem de mensagem, não seria possível fazer o gerenciamento dos processos participantes. Basicamente, existem dois ambientes de passagem de mensagem bastante utilizados que são o PVM (*Parallel Virtual Machine*) e o MPI (*Message Passing Interface*).

2.3.4 *Parallel Virtual Machine - PVM*

Este é um pacote de software que permite que uma rede heterogênea de computadores de todos os tipos seja programada como se fosse apenas uma única máquina paralela virtual. O PVM é composto, basicamente, de três partes: uma biblioteca de funções que implementam para o usuário as diretivas de programação da Máquina Virtual, um processo daemon que rodará em cada nó participante e um console de onde podem ser executadas algumas funções básicas de controle [32].

A implementação do PVM é baseada em processos do Unix. Na verdade, cada tarefa PVM é um processo Unix. Isto explica parcialmente a alta portabilidade do sistema para computadores de arquiteturas tão diferentes. Tarefas são as unidades básicas de execução do PVM

Os programas em PVM podem ser executados espalhados por uma rede de natureza heterogênea. Mais particularmente, é possível disparar tarefas em computadores em qualquer parte da Internet, desde que o usuário tenha acesso à máquina. Esta independência em relação à rede de comunicações que liga os nós participantes é garantida pelo uso do

protocolo TCP/IP para comunicação entre as tarefas através da rede.

A abordagem utilizando PVM não será feita no presente trabalho devido a decisões de projeto.

2.3.5 *Message Passing Interface - MPI*

Um dos ambientes mais utilizados de passagem de messagem é o MPI (*Message Passing Interface*). O MPI é uma biblioteca de passagem de messagem e de domínio livre em relação a *software* [28].

O MPI é uma interface para a troca de mensagens entre processos. Essa interface dá suporte para a comunicação entre processos trabalhando com o conceito de instruções simples de *send* e *receive*. Existem derivações dessas instruções um pouco mais complexas para a realização de tarefas coletivas entre os processos [33].

No que diz respeito à bioinformática, devido ao *overhead* de comunicação entre os processos, gerado pelo MPI, evita-se paralelizar as fases de construção da árvore guia e do alinhamento múltiplo propriamente dito, pois o ganho de desempenho não seria significante.

Na fase de alinhamento, também não geraria um ganho de desempenho significativo, pois como pode ser visto na figura 9, os alinhamentos entre as sequências humano e macaco, e as sequências camundongo e rato poderiam ser processadas paralelamente. Contudo, os demais alinhamentos teriam que ser conduzidos de forma sequencial, visto que há dependência de dados.

Devidos às abordagens do projeto, e o maior conhecimento na manipulação de suas funções, o MPI foi escolhido como ambiente de passagem de mensagem.

2.4 Otimização de Algoritmos

Na área de computação, otimização é o processo em que se modifica um sistema para fazer com que alguns aspectos dele trabalhem de forma mais eficiente, promovendo um melhor uso dos recursos disponíveis. Assim, um programa de computador pode ser otimizado, de tal maneira, para que ele execute uma determinada tarefa mais rapidamente, operando com menos memória para armazenamento e com uma necessidade de processamento menor do que aconteceria convencionalmente. O sistema citado anteriormente pode ser um simples programa de computador, uma coleção de computadores, ou mesmo uma rede inteira como a Internet.

Assim, analisando-se a evolução do uso dos algoritmos para bioinfomática, passando da programação dinâmica, para os algoritmos de alinhamento múltiplo de sequências e, finalmente, os alinhamentos múltiplos paralelos, nota-se que ainda existem melhorias que podem ser feitas para conseguir bons resultados e com um bom desempenho, e aumentar o número de sequências a serem analisadas. Essas melhorias podem partir de otimizações feitas nesses algoritmos, através de diversas heurísticas existentes, como por exemplo *simulated annealing* [34], buscas tabu [35], colônias de formigas [2], entre outras.

Mostra-se cada vez mais interessante a busca por novas heurísticas que possam otimizar ainda mais o processo de alinhamento múltiplo, permitindo manipular uma quantidade ainda maior de dados[36], visto que os problemas de bioinformática, em um determinado ponto, são problemas de busca. Pode-se ainda trabalhar com heurísticas diferentes em um mesmo problema, como pode ser visto em [37], mesclando-se as estratégias para a solução do problema da melhor maneira possível.

2.4.1 Heurísticas

Uma heurística pode ser definida como um método para auxiliar na solução de um problema, geralmente utilizada para levar de forma mais rápida a uma resposta mais próxima da melhor solução. Tecnicamente, na área de Ciência da Computação, trata-se

de um algoritmo que tenta encontrar ou a solução ótima no menor tempo possível, ou encontrar o menor custo possível de um dertminado caminho.

O grande problema das heurísticas é que mesmo que em uma determinada execução consiga-se encontrar a solução ótima, ou o melhor custo possível, nem sempre esses casos desejados serão atingindos, ou seja, não se consegue garantir que sempre as melhores condições serão conseguidas.

Em problemas de otimização a verdadeira otimalidade de um determinado resultado é medida pelo valor da função objetivo ou funçao do custo [38]. A principal ideia de uma técnica de otimização é encontrar o valor ideal da função objetivo para cada caso em que ela é aplicada.

Existem casos em que se busca maximizar o valor da função objetivo em busca dos chamados pontos de máximo. Para alguns problemas, os valores máximos são os mais interessantes. Contrariamente, para determinados casos, os pontos de mínimo são buscados. Assim, procura-se minimizar o valor da função objetivo para se alcançar a melhor solução [39].

Em bioinfomática, para buscar o melhor alinhamento em um tempo razoável, procura-se maximizar o valor da função objetivo com relação a pontuação do alinhamento. Sempre que a execução de um alinhamento é iniciada, almeja-se sempre atingir a pontuação máxima (maior número de coincidências) para aquele alinhamento. Nem sempre isso é possível e nem sempre que se consegue uma pontuação muito elevada, as coincidências trazem consigo algum sentido biológico.

Nas seções 2.4.2, 2.4.3 e 2.4.4 foram selecionadas algumas heurísticas para estudo. Nelas são mostradas como essas heurísticas trabalham, bem como suas aplicações em bioinformática.

2.4.2 Problema de Busca Tabu

A abundância de problemas de otimização encontrados na prática, como por exemplo em telecomunicações, logística, planejamento financeiro, transporte, produção, entre outros, motivaram o desenvolvimento de poderosas técnicas de otimização [40]. Estas técnicas são, geralmente, resultado de adaptações de ideias de várias áreas de pesquisa. A necessidade é desenvolver procedimentos que sejam eficientes e capazes de manipular a complexidade dos problemas de otimização da atualidade.

Como alguns exemplos que podem ser citados, tem-se programação de fluxo de redes que herdou e explorou ideias da eletricidade e modelos hidráulicos; *simulated annealing* que é baseado em processos físicos da metalurgia; algoritmos genéticos que procuram imitar os fenômenos biológicos da reprodução evolucionária; sistemas de formigas, que simulam uma colônia de formigas que cooperam numa atividade de solução de um problema comum.

Seguindo por essas vertentes, surgiu a busca tabu que nada mais é que derivar e explorar uma coleção de princípios de solução de problema inteligente. Nesse sentido, pode-se dizer que busca tabu é baseada em conceitos selecionados que unem os campos da inteligência artificial e otimização [40].

O método de busca tabu é baseado em procedimentos projetados para transpor as barreiras da factibilidade ou da otimilidade local que são tratadas como barreiras. Esse método concentra-se em restrições impostas ao problema para guiar um processo de busca e negociar regiões difíceis [40].

A busca tabu pode ser aplicada em bioinformática para encontrar um alinhamento global para problemas de alinhamento múltiplo. Basicamente, a busca tabu começa de uma solução inicial e iterativamente explora as vizinhanças da solução inicial para gerar os próximos movimentos para as soluções, como pode ser visto em [35].

2.4.3 *Simulated Annealing*

O *Simulated Annealing* consiste em um bom método heurístico para resolver problemas de otimização combinatória [41]. Trata-se de uma abordagem probabilística que pode ser utilizada para encontrar mínimos globais de funções em problemas de otimização.

Basicamente, para a aplicação dessa técnica em problemas de otimização, deve-se definir um espaço de estados $X = \{x_1, ..., x_n\}$ e uma função custo $C : X \rightarrow R$, onde R é um conjunto de números reais a ser definido. Um valor real $C(X)$ deve ser atribuído a cada estado x. O objetivo dos problemas de otimização é encontrar o estado ótimo x_{opt}. O *Simulated Annealing* gera continuamente um novo estado x_{new} a partir de um estado corrente $x_{current}$ através da aplicação de regras de transição e regras de aceitação. Os critérios para as regras de aceitação são:

1. Se $\Delta E \leq 0$, aceita um novo estado x_{new} com ΔE sendo a variação de energia.
2. Se $\Delta E > 0$, aceita um novo estado x_{new} com probabilidade $P(\Delta E) = e^{\Delta E/T}$, onde T é a temperatura e $\Delta E = C(x_{new}) - C(x_{current})$ é a diferença de custo.

A probabilidade $P(\Delta E)$ impede a fixação de um mínimo local. O estado $x_{current}$ é chamado de mínimo local se não existir nenhum estado x_{new} em X que é gerado a partir do estado $x_{current}$ pela aplicação de uma regra de transição simples e que tenha um custo menor do que o do $x_{current}$.

A temperatura T controla a probabilidade de aceitar um novo estado x_{new}. Inicialmente, T começa com uma alta temperatura e depois de cada iteração, T diminui até zero pela aplicação do escalonamento *annealing*. Se um escalonamento *annealing* e uma cuidadosa seleção do número de iterações forem feitas, o *Simulated Annealing* converge para um estado de mínimo global x_{opt}.

Os problemas de alinhamentos múltiplos em bioinformática encaixam-se como problemas grandes e complexos de análise combinatória que objetivam encontrar um mínimo

global entre as sequências que são comparadas [34], com isso, gerando um custo menor em sua execução. Dessa maneira, encontra-se perfeitamente viável o uso do *Simulated Annealing* para problemas de alinhamentos múltiplos.

O uso do *Simulated Annealing* em problemas de bioinfomática esbarra, inicialmente, na principal desvantagem do método que se relaciona com o grande custo computacional para se processá-lo. Pelo fato do *Simulated Annealing* ser baseado no método de Monte Carlo [42], que permite que um novo estado (um alinhamento) tenha um custo maior do que o estado atual, torna-se inviável o seu uso direto como heurística para a resolução dos problemas. Assim, deve-se agregar uma outra estratégia que aumente a vazão do *Simulated Annealing* quando utilizado nessas aplicações.

As estratégias de aumento da vazão do *Simulated Annealing* baseiam-se em trabalhar principalmente com as suas duas fases principais, que são as fases de alta temperatura e a fase de baixa temperatura. A fase de alta temperatura tenta identificar todos os estados com um custo mais elevado do que o estado atual e a fase de baixa temperatura, tenta identificar os estados de temperatura mais baixa que o atual. Isso permite que os estados fiquem mais próximos do estado atual. Em bioinformática o que se realiza é a exclusão de uma das fases através da utilização de outra técnica, reduzindo assim o tempo computacional para o método e permitindo a obtenção de bons alinhamentos de saída [34].

Além dessa estratégia citada anteriormente, pode-se também fazer uso de outro conceito utilizado em *Simulated Annealing* que é o da energia livre de Gibbs, que nada mais é do que a energia disponível para realizar o trabalho útil. Esse conceito, relacionado à genômica, diz respeito a análise de conformações de proteínas, quando se estuda as suas relações de homologia com outras proteínas [43] e também pode ser utilizado em algoritmos evolutivos como sua métrica para a predição de estruturas secundárias de RNA [44].

2.4.4 Colônias de Formigas

Colônia de formigas é uma metaheurística inspirada na natureza para resolver difíceis problemas de otimização combinatorial [45]. O uso de colônias de formigas como uma heurística para otimização de problemas pode ser aplicado em uma série de casos, definindo a sua classificação como uma metaheurística.

Um metaheurística é uma heurística para se resolver de forma genérica problemas de otimização [46]. Essas metaheurísticas quando aplicadas à problemas computacionais, tentam reduzir a complexidade computacional desses problemas e proporcionar um melhor desempenho para determinados algoritmos. Consequentemente, permitem um ganho, na maioria das vezes bastante significativo, em relação ao tempo de obtenção da solução, permitindo um melhor aproveitamento dos tempos de computação das unidades de processamento.

A metáfora de colônias de formigas é inspirada pelos pesquisadores na tentativa de entender como quase todas as formigas cegas estabelecem o menor caminho de suas colônias às fontes de comida e vice-versa [47]. Essencialmente, percebe-se que as formigas utilizam o feromônio com o intuito de demarcação, que acaba se transformando em um mecanismo de comunicação entre elas. Na medida que uma formiga se movimenta, ela deixa quantidades variadas de feromônio que são detectadas pelas outras formigas no decorrer do caminho. Dessa forma, o caminho é demarcado por uma trilha dessas substâncias [48]. Quanto mais as formigas passam, mais feromônio é depositado no caminho. Como as formigas seguem o feromônio, quanto mais forte for a trilha de feromônio no caminho, mais preferido será esse caminho pelas formigas. Consequentemente, as formigas podem estabelecer o menor caminho entre a sua colônia e as fontes de comida e vice-versa.

A figura 12 traz uma descrição de como ocorre o comportamento das formigas. Inicialmente, três formigas deixam o ninho em direções aleatórias para procurar comida. Enquanto elas andam, elas depositam feromônio nas trilhas. Agora, assuma que a formiga 1 encontra uma fonte de comida. Ela coletará alguma comida e voltará ao ninho

pela sua trilha de feromônio, deixando mais feromônio na trilha, enquanto as formigas 2 e 3 ainda procuram um caminho aleatório. Quando o próximo grupo deixar o ninho para procurar comida, eles detectarão duas vezes mais feromônio no caminho 1, do que no caminho 2 ou 3. Considerando ainda que a probabilidade para cada caminho é ponderada pelo valor do feromônio, mais formigas seguirão o caminho 1 na sua busca por comida. Desta forma, as formigas estabelecem o menor caminho entre as suas colônias e as fontes de comida.

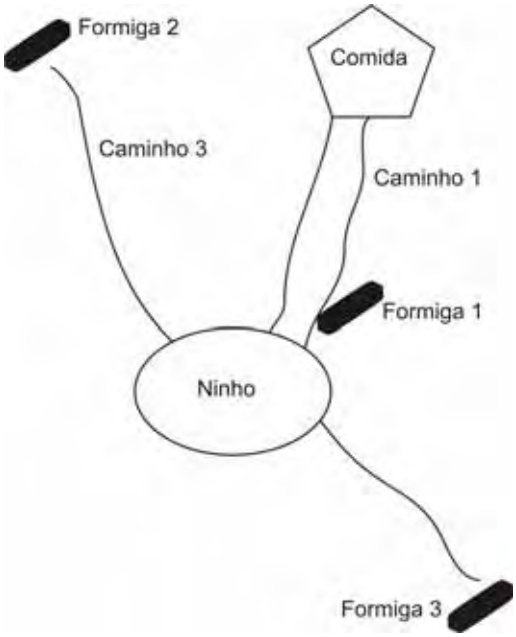

Figura 12: Movimento das formigas entre a colônia e a fonte de comida [2].

No que se relaciona a parte matemática do método de colônias de formigas, o componente central do algoritmo é um modelo probabilístico parametrizado que é chamado de modelo de feromônio. O modelo de feromônio consiste de um vetor T de parâmetros modelo chamado de parâmetros das trilhas do feromônio. Os parâmetros das trilhas do feromônio $\mathsf{T_i} \in \mathsf{T}$ que são associados aos componentes das soluções têm valores τ_i que são chamados de valores dos feromônios. O modelo de feromônios é utilizado para gerar probabilisticamente soluções para o problema em questão construindo-as através de um conjunto finito de componentes da solução. No momento da execução, o algoritmo de colônia de formigas atualiza os valores dos feromônios usando as soluções geradas anteriormente [45, 49].

A atualização dos valores concentra-se em encontrar regiões do espaço de busca que contenham soluções de alta qualidade, pois essa alta qualidade é um item muito importante dos algoritmos de colônias de formigas e que permite que a técnica receba muitos refinamentos, para cada vez mais possuir solução muito próximas da ideal. As soluções qualificadas encontradas pelo algoritmo de otimização advêm de bons componentes da solução.

A abordagem de colônias de formigas fundamenta-se basicamente na repetição de dois passos para resolver os problemas de otimização [45]. Esses passos são descritos a seguir:

1. Soluções candidatas são construídas usando o modelo de feromônios, que é uma distribuição de probabilidade parametrizada sobre um espaço de soluções.

2. As soluções candidatas são utilizadas para modificar os valores do feromônio, de tal forma que são consideradas como um item de entrada de uma futura amostra, na busca das soluções com uma ótima qualidade.

Nas diversas aplicações em que os algoritmos de colônias de formigas são utilizados esses passos podem variar, principalmente na forma de como atualizar o feromônio, e geralmente variam, pois a eles são acrescentadas algumas melhorias à técnica que acaba tornando-a ainda mais poderosa e útil. Na seção 2.4.5 são mostrados alguns problemas onde a técnica pode ser aplicada e na seção 2.4.6 mostra-se a viabilidade do uso de colônias de formigas em bioinformática para problemas de alinhamentos múltiplos de biossequências.

2.4.5 Alguns Problemas Solucionáveis Através de Colônias de Formigas

Existem alguns problemas clássicos que podem utilizar a metaheurística de colônias de formigas para resolvê-los. Alguns como o Problema do Caixeiro Viajante [48], o Problema de Satisfatibilidade Booleana [50],o Problema da Mochila [51], o Problema de Roteamento

de Veículos [3] e finalmente, no foco deste trabalho, o problema de alinhamentos múltiplos de biossequências, que podem fazer uso dessa técnica para otimizá-los.

Problema do Caixeiro Viajante: Trata-se de um problema típico de otimização, bastante utilizado para demonstrar problemas de difícil solução em computação. Trata-se de um problema da classe dos NP-Completos [48].

Basicamente o problema resume-se em um caixeiro viajante que deseja visitar N cidades de uma certa localização e que, entre algumas cidades existem rotas através das quais ele pode viajar a partir de uma cidade para outra. Cada rota tem um número associado que pode representar a distância ou o custo necessário para percorrê-la. Dessa forma, o caixeiro viajante almeja encontrar um caminho que passe por cada uma das N cidades apenas uma vez, e que esse caminho seja o que desempenhe o menor custo possível para o caixeiro.

A figura 13 mostra o caixeiro viajante no momento da escolha do melhor caminho. Basicamente é a análise de um grafo não dirigido, em que as arestas (rodovias) interligam os vértices (cidades) A, B, C e D. Às possíveis arestas para a figura 13 são: $\overline{AB} = \overline{BA}$, $\overline{AC} = \overline{CA}$, $\overline{AD} = \overline{DA}$, $\overline{BC} = \overline{CB}$, $\overline{BD} = \overline{DB}$, $\overline{CD} = \overline{DC}$.

Problema de Satisfatibilidade Booleana: Primeiro problema identificado como um problema NP-Completo. Determina se existe uma valoração para as variáveis de uma determinada fórmula booleana tal que essa valoração satisfaça esta fórmula em questão [52].

Considerando x_1, x_2, x_3 como variáveis booleanas e a expressão $(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4})$, caso exista uma atribuição de valores verdade para as variáveis da fórmula que torne a fórmula avaliada verdadeira, esta fórmula é considera satisfatível. Em contrapartida, se nenhuma atribuição levou a uma avaliação da fórmula como verdadeira, ela é considerada insatisfatível.

Problema da Mochila: Trata-se também de um problema de otimização e enquadra-se também na classe dos problemas NP-completos.

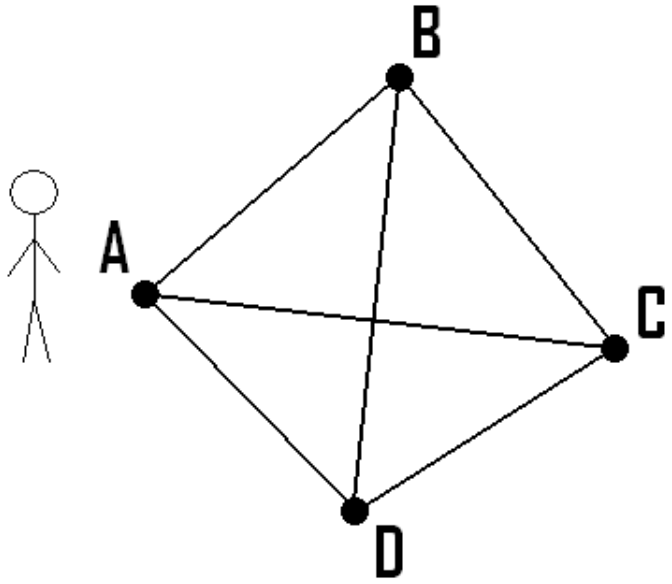

Figura 13: Caixeiro viajante escolhendo o melhor caminho.

O seu nome se deve ao modelo de uma situação em que é necessário preencher uma mochila com objetos de diferentes pesos e valores, objetivando-se preencher a mochila com o maior valor possível, sem ultrapassar o peso máximo [51].

Uma utilidade que pode ser encontrada para esse problema, por exemplo, é no momento de carregar-se um contêiner. Existem vários produtos de tamanhos distintos e necessita-se carregar de tal forma que o contêiner tenha o maior valor, ou a maior quantidade de produtos colocados dentro dele.

Problema de Roteamento de Veículos (PRV): Trata-se de um dos problemas mais estudados em otimização combinatória. Consiste basicamente no atendimento de um conjunto de consumidores, por intermédio de uma frota de veículos, que partem de um ou mais pontos denominados depósitos. A restrição presente no PRV é que cada veículo ν possui uma capacidade C_ν e o somatório de todas as demandas dos consumidores atendidos por um veículo ν não pode ultrapassar $C\nu$ [3].

O PRV, apesar do seu enunciado relativamente simples, apresenta elevada complexidade computacional, por isso se torna interessante a aplicação da metaheurística de colônia de formigas para resolvê-lo.

A função objetivo para esse problema pode cumprir diversas funcionalidades, claro que de forma independente, que são: minimizar o custo total da operação, minimizar o tempo total de transporte, minimizar a distância total percorrida, minimizar o tempo de espera, maximizar o benefício, maximizar o serviço ao cliente, minimizar a utilização de veículos, equilibrar a utilização dos recursos.

Na figura 14, mostra-se o funcionamento da técnica de roteamento de veículos com um único depósito central necessitando distribuir para as rotas 1, 2, 3 e 4.

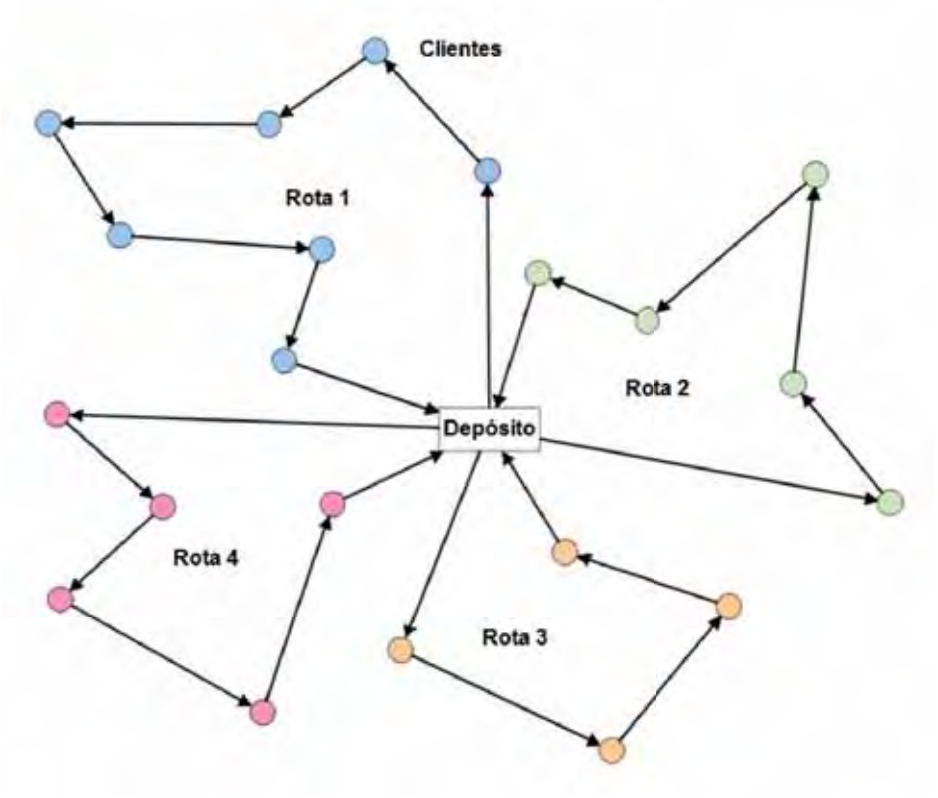

Figura 14: Esquema de roteamento de veículos [3].

2.4.6 Colônias de Formigas aplicadas à Bioinformática

Conforme visto anteriormente, as colônias de formigas possuem uma grande variedade de aplicações em diversos problemas de otimização e um dos problemas que necessitam do uso dessas otimizações são os alinhamentos múltiplos de biossequências. Como mostrado em seções anteriores, esses problemas de alinhamento apresentam um alto grau de complexidade computacional e necessitam, além do paralelismo aplicado aos algoritmos,

de técnicas de otimização apuradas que possam contribuir para a redução do tempo de computação.

Alguns trabalhos relacionados à área de bioinformática mostram que é possível o uso de colônias de formigas em problemas de alinhamento adotando diferentes abordagens.

Por exemplo, em [2], pode ser vista a aplicação de colônias de formigas no momento da realização do alinhamento múltiplo. A ideia principal, tratando-se de forma ilustrativa, é que as formigas peguem as subsequências e movimentem em um intervalo associado com cada sequência, fortalecendo a trilha de feromônio (aumento da probabilidade) quando uma coincidência é encontrada para aquela sequência quando comparada com as demais sequências.

A abordagem apresentada em [53] realiza a mesclagem do uso da técnica de colônia de formigas agregada à técnica de algortimos genéticos para a realização do alinhamento múltiplo de biossequências. As colônias de formigas são utilizadas para melhorar o desempenho da busca local do algoritmo genético e provocam um aumento significativo do desempenho do algoritmo genético em relação aos tempos de busca nos alinhamentos. O trabalho mostra o ganho de desempenho em relação a alguns outros algoritmos. O algoritmo executa iterando os processos de inicialização, os operadores de reprodução sexual, os operadores de mutação, os operadores de seleção e a técnica de colônia de formigas para alinhamentos múltiplos até que um determinado critério pré-estabelecido seja satisfeito.

A técnica de colônias de formigas pode ser utilizada também para construção das árvores filogenéticas durante o processo de alinhamento múltiplo de biossequências com o objetivo de reduzir os pesos dos ramos da árvore [54]. Em [54] a construção das árvores filogenéticas é feita através do problema das árvores de *Steiner* [55] e para se resolver esse problema, aplica-se colônias de formigas. Essa abordagem para construção da árvore filogenética ou árvore guia, pode ser uma alternativa em relação ao uso do algoritmo *Neighbor-Joining* [56].

Com uma vertente um pouco mais voltada a biologia molecular, especialmente voltada

a exploração de dados proteômicos, a metaheurística da colônias de formigas pode ser aplicada para a análise de dados de proteínas obtidos a partir de espectrometria de massa como pode ser visto em [57]. A técnica de colônias de formigas é aplicada para realizar o processo de busca em grafos na pesquisa nas bases de biossequências. Para um dado espectro obtido, o algoritmo de colônia da formigas é aplicado para cada peptídeo da base de dados de sequências considerada, levando em conta uma classificação hierárquica desses peptídeos. A busca nessa bases torna-se mais complexa quando a procura por mutações também é considerada [57].

3 Desenvolvimento do Trabalho

3.1 O Algoritmo Proposto

As seções a seguir trazem todas as descrições do algoritmo de alinhamento múltiplo proposto e implementado nesse trabalho, bem como as técnicas de otimizações empregadas nele através das Colônias de Formigas. Alguns fluxogramas foram construídos para auxiliar nas descrições e torná-las mais claras no que se relaciona à execução do algoritmo.

3.1.1 Descrição Geral

O algoritmo proposto se divide em várias subpartes no que diz respeito a sua implementação e a sua organização. No entanto, existem três grandes partes que englobam as demais, as quais foram citadas em seções anteriores e são elas, na seguinte ordem: Cálculo da Matriz de Distâncias, Geração da Árvore Guia e a execução do Alinhamento Múltiplo propriamente dito.

Na figura 15, apresenta-se o fluxograma geral do algoritmo de alinhamento múltiplo. Acrescenta-se a esse fluxograma o estágio de entrada de dados que é feita através da leitura de um arquivo com as biossequências.

A leitura dos arquivos de biossequências é realizada através da manipulação de cadeias de caracteres (*strings*) e essas cadeias são inseridas em duas matrizes, uma que recebe os nomes das biossequências, denominada *names* e uma outra que recebe as biossequências propriamente ditas, denominada *sequences*. Essas duas matrizes são declaradas dentro de uma estrutura (*struct*) criada para manipular apenas as entradas de dados e essa estrutura

Figura 15: Fluxograma genérico do funcionamento do algoritmo.

denomina-se *SequenceSet*.

Todos os arquivos lidos devem estar em formato FASTA que é o formato adotado dentro da bioinformática para normatizar a leitura dos arquivos de biossequências nos programas de alinhamentos. Esses arquivos possuem algumas peculiaridades como a quebra de linha separando uma sequência da outra e o símbolo > indicando o início da nova sequência. Na mesma linha do símbolo >, encontra-se o nome da sequência que pode ser utilizado para organizar os resultados no momento da apresentação das saídas após o processamento, ou mesmo para a identificação das sequências no próprio arquivo FASTA.

Na figura 16, pode-se verificar um exemplo de um arquivo de entrada de nucleotídeos em formato FASTA com o nome `HomoSapiens_Teste.fas`. Todos arquivos FASTA devem ter a extensão `.fas`.

Na figura 17, pode-se verificar um exemplo de um aquivo de entrada de aminoácidos em formato FASTA com o nome `alinhamento2.fas`.

Os arquivos submetidos para as análises, em geral, possuem milhares, algumas vezes milhões de sequências para serem alinhadas e comparadas. Evidencia-se que, pensando no mesmo gene a ser analisado, os arquivos de aminoácidos para esse gene são menores que os arquivos de nucleotídeos em tamanho, visto que para formar um aminoácido precisa-se de três nucleotídeos (*códon*).

3.1.2 Cálculo da Matriz de Distâncias

O cálculo da matriz de distâncias, conforme citado anteriormente, mostra-se como a fase mais crítica do algoritmo de alinhamento múltiplo, visto que para o cálculo dessa matriz utiliza-se o alinhamento par-a-par, o que consome bastante capacidade de processamento.

Assim, como o algoritmo de alinhamento múltiplo proposto foi construído para operação em paralelo, a fase que mais necessita desse paralelismo é a fase do cálculo da matriz de distâncias. Dessa forma, explora-se o paralelismo no momento do alinhamento par-a-par,

Figura 16: Trecho de um arquivo de nucleotídeos em formato FASTA.

Figura 17: Trecho de um arquivo de aminoácidos em formato FASTA.

distruibuindo-se os alinhamentos, que são realizados por um algoritmo de programação dinâmica, entre os nós da máquina paralela.

Antes dessa distruibuição existe um cálculo das cargas (conjuntos de sequências a serem alinhadas) que serão designadas para cada processador que gerenciará os seus processos localmente.

A figura 18 traz um fluxograma da rotina que realiza o cálculo da carga de cada processador. Inicialmente, aloca-se o tamanho da matriz de distâncias para a matriz que receberá essas distâncias após os cálculos, inicializa-se cada elemento dessa matriz com o valor zero e, na sequência, calcula-se a carga para cada uma das unidades de processamento. Esse cálculo de carga é realizado localmente nas unidades de processamento.

No que diz respeito ao alinhamento par-a-par, ele utiliza programação dinâmica e posiciona os caracteres entre as duas sequências participantes, procurando obter o maior número de coincidências entre elas, como descrito em seções anteriores.

Após a realização desse alinhamento, os nós devolvem para o *front-end* os seus resultados, e o *front-end* os organiza, designando novas tarefas até que todos os alinhamentos par-a-par sejam realizados. Por isso, dependendo do número de biossequências a serem processadas, o tempo para esse estágio pode aumentar significativamente.

Para a realização desses alinhamentos par-a-par são invocadas basicamente três funções: `bestPairAlignScore`, `bestPairAlignment` e `distance`. A função `bestPairAlignScore` realiza um cálculo de alinhamentos semi-globais, que são utilizados para contribuir na tarefa de realização do alinhamento global das biossequências, realizando um pré-alinhamento, o que otimiza o alinhamento global, tornando-o um pouco mais rápido. O alinhamento global é realizado pela função `bestPairAlignment` que realmente termina o alinhamento par-a-par. O valor final da distância, ou pontuação, entre cada par das biossequências é calculado pela função `distance`, que retorna para uma matriz `myDistances` o valor calculado e essa matriz é retornada para o *front-end* e agrupada juntamente com as matrizes calculadas nos demais nós, formando uma única matriz com todos os

Figura 18: Fluxograma relativo à rotina de cálculo da carga para cada processador.

valores de pontuação de todas as combinações de pares de biossequências.

3.1.3 Construção da Árvore Guia

Após a construção da matriz de distância, cujo processo foi descrito na seção 3.1.2, inicia-se o passo de construção da árvore guia, ou árvore filogenética, que é apresentado nesta seção.

O processo de construção da árvore guia está diretamente relacionado a sua fase anterior que é o do cálculo da matriz de distâncias, pois a partir das distâncias calculadas, as sequências com valores de pontuação mais próximos são alocados em ramos mais próximos da árvore. A alocação possui a ideia de identificar sequências que estão biologicamente mais próximas.

Essa reorganização que é feita com as biossequências é motivada pelo uso na fase seguinte que é a do alinhamento múltiplo, reduzindo assim, a carga de trabalho dessa última fase. Assim, os blocos de sequências mais próximas podem ser alinhados, culminando em um valor maior da função objetivo e, muito provavelmente, uma significância biológica.

A técnica algorítmica adotada para a construção da árvore guia é a de *Neighbor-Joining* [26], que simplificadamente, atua através da união dos vizinhos mais próximos. Nesse caso, realiza-se o agrupamento com as sequências vizinhas mais próximas, ou seja, com valores de pontuação mais próximos. Pensando em uma estrutura de árvore binária e considerando que a raiz receba o menor valor, o algoritmo vai alocando os valores de pontuação, contidos na matriz de pontuação, nas folhas da árvore, em ordem crescente.

A figura 19 apresenta um fluxograma da rotina de construção da árvore. Ela basicamente cria a árvore para o tamanho da matriz de distâncias, com todos os nós nulos. Após esse passo, os nós são alocados, colocando-os de forma ordenada para que os elementos filogeneticamente mais semelhantes estejam mais próximos. Esse processo ocorre até que todos os valores da matriz de pontuação tenham sido alocados na árvore filogenética.

Na construção da árvore guia, existe uma função mestre que lastreia todo o seu funci-

Figura 19: Fluxograma relativo à rotina de alocação dos elementos na árvore guia.

onamento e a sua construção, que é a função `createNJTree`. Ela é responsável por alocar todos os nós da árvore nas respectivas posições e gerenciar esses nós.

Toda a organização da árvore guia é escrita em arquivo por duas outras funções que são `printDND` e a `printDNDRecursive`. Esse arquivo é utilizado pela fase do alinhamento múltiplo. Na verdade, a função `printDND` identifica o nó raiz da árvore e a partir dai, faz uma chamada para a função recursiva `printDNDRecursive` para que ela posssa então percorrer os nós da árvore.

3.1.4 Alinhamento Múltiplo

O alinhamento múltiplo propriamente dito é a última fase realizada no alinhamento progressivo. Basicamente, a ideia dessa fase é que o alinhamento seja feito de forma progressiva, permitindo que sequências de ramos mais próximos sejam alinhadas primeiramente.

O algoritmo preserva todos os *gaps* inseridos no grupo de sequências alinhadas na iteração anterior no momento de alinhar uma nova sequência com esse grupo. Sucessivamente, o algoritmo alinha a sequência ainda não alinhada com uma sequência com a pontuação mais próxima a ela, prosseguindo até o término das sequências. Vale ressaltar que o algoritmo de alinhamento múltiplo progressivo sempre retorna algum tipo de resposta, independente se esta possua ou não sentido biológico.

As figuras 20, 21 e 22 retratam o funcionamento do processo do alinhamento múltiplo, a partir das sequências que compõem a árvore guia. Na figura 20, o algoritmo alinha as sequências Ns00 e Ns01 e as agrupa. Na figura 21 o grupo já alinhado das sequências Ns00 e Ns01 é alinhado em conjunto com a sequência Ns02. Na figura 22, o grupo formado por Ns00, Ns01 e Ns02 é alinhado em conjunto com Ns03, e assim caminha-se, sucessivamente, até que todas as sequências sejam agrupadas em um determinado passo N.

Nesta etapa do algoritmo, basicamente são três funções que realizam todas as ações, que são `progressiveAlign`, `progressiveAlignRecursive` e a `alignProfiles`. A função

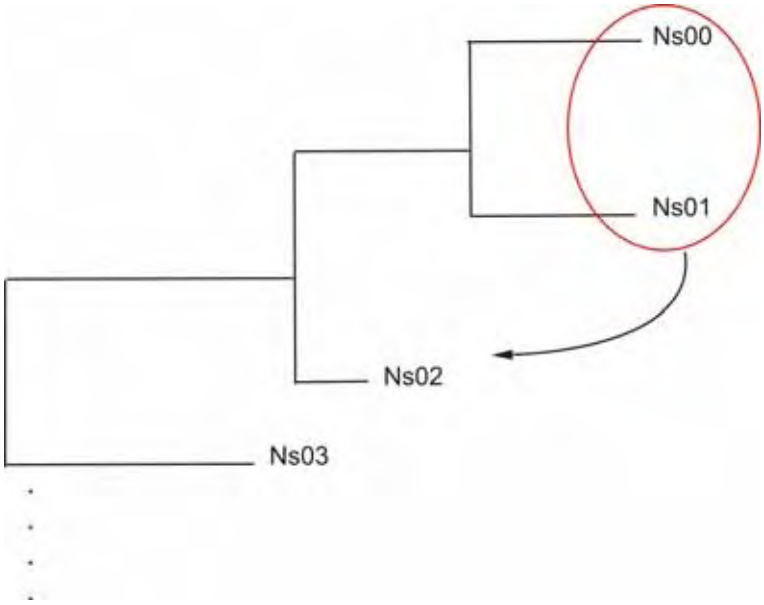

Figura 20: Exemplificação do uso da árvore guia pelo alinhamento múltiplo - Passo 1.

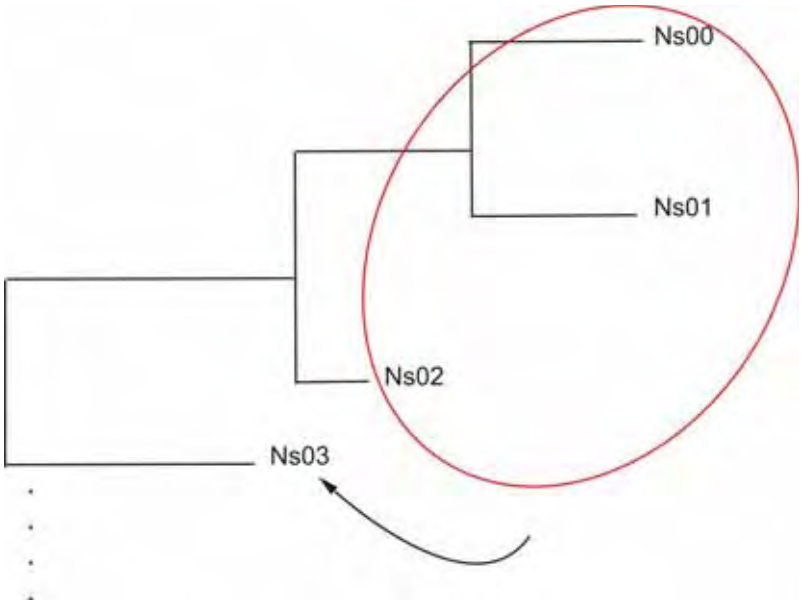

Figura 21: Exemplificação do uso da árvore guia pelo alinhamento múltiplo - Passo 2.

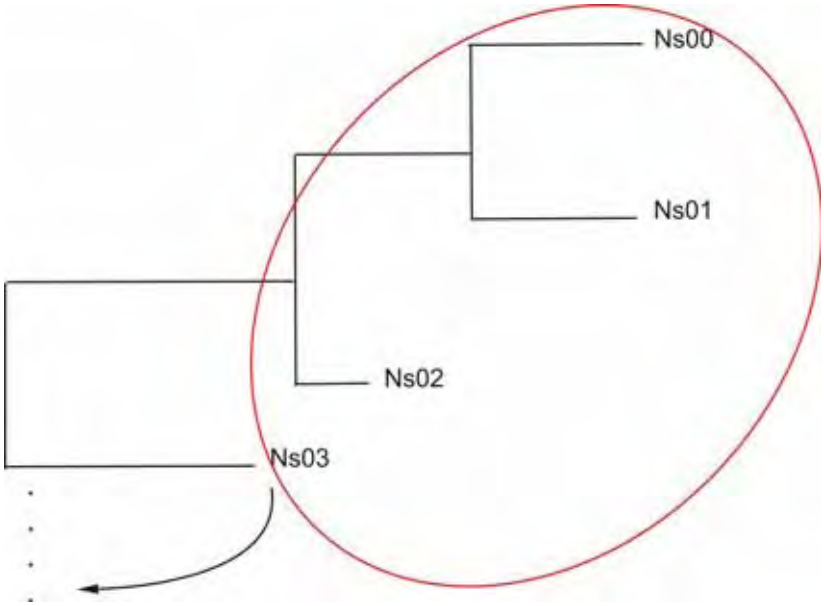

Figura 22: Exemplificação do uso da árvore guia pelo alinhamento múltiplo - Passo 3.

`progressiveAlign` funciona apenas como uma interface para receber a árvore e a partir dai ela passa o nó raiz para a função `progressiveAlignRecursive`. A função `progressiveAlignRecursive` vai posicionando as sequências para o alinhamento múltiplo e então realiza as chamadas para a função `alignProfiles` que vai adequando os perfis das biossequências para conseguir o melhor alinhamento entre todas elas para o resultado final. A figura 23 traz o fluxograma que mostra exatamente o que foi descrito neste parágrafo.

Assim, termina-se a descrição do algoritmo de alinhamento múltiplo progressivo implementado, com alguns de seus detalhes. Nas seções seguintes, serão mostrados o uso das técnicas de Colônias de formigas para otimizar algumas partes do algoritmo e alguns testes feitos com o algoritmo antes e depois das otimizações.

3.2 Otimização com Colônias de Formigas

Esta seção mostra a maneira como foi implementada a otimização para o algoritmo de alinhamento múltiplo utilizando colônias de formigas e outras estratégias.

Figura 23: Fluxograma do algoritmo de alinhamento múltiplo.

3.2.1 Estimativa de Pontuação

Como visto em seções anteriormente apresentadas, a heurística de colônias de formigas, quando adequada para determinados problemas, pode tornar-se uma técnica bastante eficiente e muito poderosa.

Com o intuito de aproveitar o que a técnica de colônia de formigas tem para oferecer, o presente trabalho, baseando-se no algoritmo de alinhamento múltiplo progressivo, otimizou alguns dos pontos do algoritmo durante a sua execução, modificando a sua estrutura de funcionamento, no que diz respeito à adoção de diferentes estratégias para a fase do cálculo da matriz de distâncias e para a fase de alinhamento múltiplo.

Inicialmente, houve uma reestruturação na forma como são calculados os valores de pontuação entre os pares de sequências, para a construção da matriz de pontuações e também o tipo de algoritmo utilizado nesse estágio. O cálculo da pontuação passa a ocorrer de forma particionada, e aplica-se uma técnica baseada em algoritmos de estimativas de pontuação [58].

Esse algoritmo de estimativa de pontuação auxilia na redução da complexidade antes gerada pelo algoritmo de programação dinâmica que realizava o alinhamento par-a-par. A complexidade do alinhamento par-a-par, considerando-se duas sequências X e Y de tamanhos m e n, respectivamente, é da ordem de $O(mn)$. Adotando a nova estratégia para o cálculo das pontuações entre os pares de sequências, a complexidade cai para a ordem de $O(m+n)$, o que traz um ganho bastante significativo.

O algoritmo de estimativas atua em quatro estágios em sua execução. Esses estágios fazem varreduras em diferentes direções (direita/esquerda) nos pares de sequências, calculando uma pontuação para aquela varredura. Os estágios são: Superior-Esquerda, Superior-Direita, Inferior-Esquerda, Inferior-Direita. Esses nomes, superior e inferior, advêm do posicionamento das sequências no momento do cálculo visto que, ilustrativamente, uma fica em cima (superior) e a outra em baixo (inferior). A questão da direção, direita ou esquerda, vai depender do sentido da varredura que será escolhido para percorrer as

sequências. A máxima pontuação obtida entre os quatro estágios é considerada a pontuação ideal e, então, é inserida na matriz de pontuação, como a pontuação para aquele par de sequências.

A figura 24 traz uma ilustração do funcionamento do algoritmo de estimativa de pontuação. Para ilustrar o funcionamento do algoritmo, adota-se, como exemplo, a execução Superior-Direita do primeiro quadrante da figura 24. Marca-se o último caracter mais à direita da sequência superior, nesse caso é o A. A partir dele, começa-se a realizar comparações com os caracteres da sequência inferior, começando da direita para a esquerda, a partir do primeiro caracter também. Se nenhum caracter semelhante for encontrado na sequência inferior, passa-se para o próximo caracter à esquerda da sequência supeior. Senão, um ponto é marcado, e começa-se novamente a varredura, tomando agora como base o próximo caracter à esquerda, do caracter que coincidiu (terceiro da direita para a esquerda na sequência inferior), na sequência inferior. Tomando por base então o T (o quarto da direita para a esquerda da sequência inferior), começam-se as comparações com a sequência superior a partir do próximo caracter à esquerda (nesse caso o segundo da direita para a esquerda na sequência superior) daquele que coincidiu na iteração anterior na sequência superior, que no caso era o caracter base, e assim sucessivamente até o final das sequências. Os demais passos, Superior-Esquerda, Inferior-Esquerda e Inferior-Direita funcionam com o mesmo procedimento, apenas alterando o sentido da varredura direita/esquerda nas sequências e o caracter de inicialização.

Com o objetivo de melhorar ainda mais o desempenho dessa fase, realiza-se em paralelo, ou seja, particionando-se as excuções desses quatro estágios. Com isso reduz-se ainda mais os tempos de computação para o cálculo dessas pontuações. A idéia é que cada estágio seja executado em um nó de processamento e devolva o seu resultado ao nó mestre.

A questão do particionamento, como pode ser visto na figura 25, é resolvida atribuindo-se um quadrante (um passo) para um nó de processamento, nesse caso representado apenas

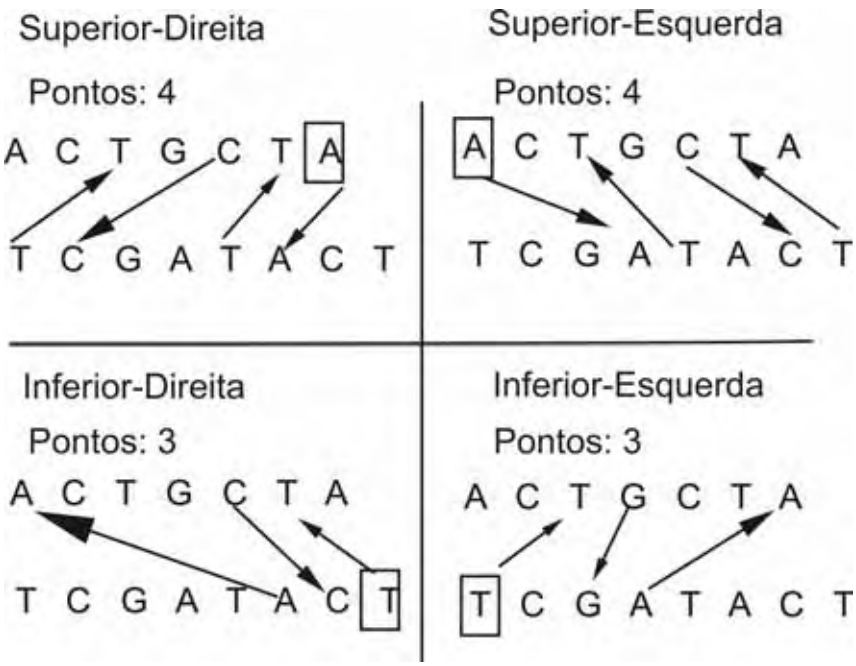

Figura 24: Ilustração do funcionamento algoritmo de estimativas de pontuação.

pela distruição de um par de sequências. Porém, a distribuição é feita conforme a quantidade nós disponíveis, chamando outros pares de sequências assim que o nó é desocupado, visto que não existe dependência para a sua execução.

3.2.2 Alinhamento Múltiplo

Agregada à técnica de estimativas de pontuação apresentada na seção 3.2.1 e à fase de construção da árvore filogenética, que é a fase intermediária e funciona da mesma maneira que a descrita na seção 3.1.3, precisa-se da fase final, que é a de alinhamento múltiplo.

Na última fase, que é descrita nessa seção, o algoritmo de colônias de formigas é aplicado de forma efetiva, mesmo essa fase não se comportando como a mais crítica em alinhamentos múltiplos de biossequências. Porém, qualquer otimização obtida traz ganhos ao tempo de execução global.

O seu funcionamento ocorre de forma sequencial, como também ocorre no alinhamento múltiplo progressivo convencional. A fase se constitui de forma descritiva e ilustrativa, por uma colônia de N formigas, cada uma representa uma solução para o alinhamento. Cada uma dessas formigas se movimenta nas sequências para encontrar caracteres coincidentes.

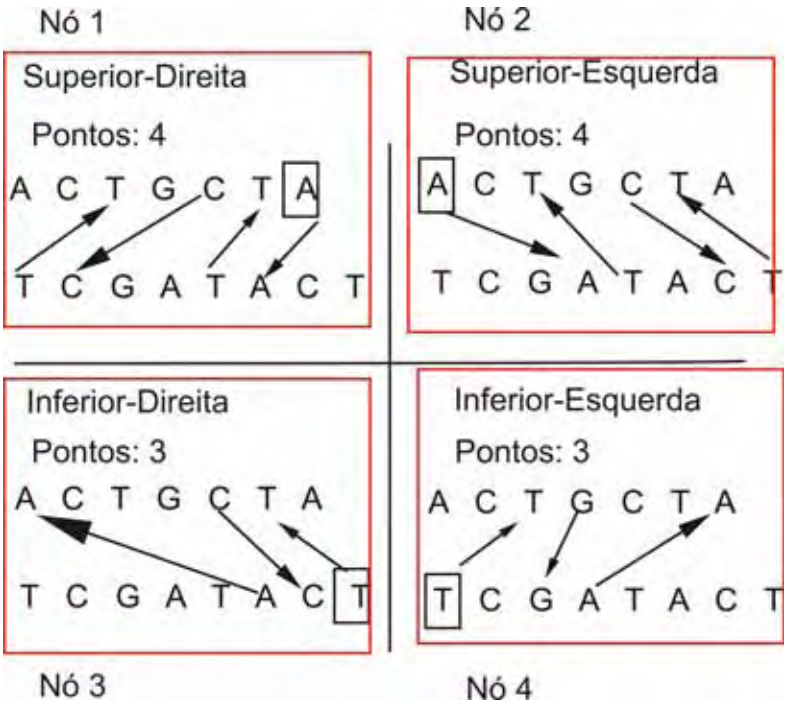

Figura 25: Ilustração do particionamento da estimativa de pontos.

Pensando em N sequências, com A_0,...., A_{N-1} sendo as sequências, uma formiga inicia a sua caminhada a partir de $A_0[0]$, que é o primeiro caracter de A_0 e seleciona um caracter de cada uma das sequências A_1,....,A_{N-1}, confrontando com $A_0[0]$. A partir da sequencia A_i, sendo *i=1,2,...,n-1*, a formiga seleciona um caracter $A_i[j]$ e através da combinação de probabilidades determinadas pela pontuação de comparações feitas com a posição $A_0[0]$, dos desvios calculados entre a sua localização com relação a posição $A_0[0]$ e das funções de trilhas de feromônios e evaporação entre $A_i[j]$ e $A_0[0]$. Além disso, uma formiga pode inserir, caso necessário, um espço vazio (*gap*) para reajustar as sequências para uma condição melhor.

No passo seguinte, a formiga começa do $A_0[1]$, seleciona os caracteres de A_1,...., A_{N-1}, confrontando agora com $A_0[1]$ para esse passo. Sucessivamente, elas vão realizando o mesmo procedimento para $A_0[2]$,..., $A_0[|A_0| - 1]$ e podendo formar diversos outros caminhos, considerando cada um desses caminhos $|A_0|$ como uma solução de alinhamento.

Na figura 26, pode ser vista a ilustração para a execução do algoritmo conforme descrito nos dois parágrafos anteriores. Nessa representação é mostrado o alinhamento

múltiplo, em seu início, com duas sequências, A_0 e A_1. Cada retângulo que envolve as bases é considerado uma formiga realizando a sua tarefa de comparar os caracteres e escolher o melhor reposicionamento entre eles.

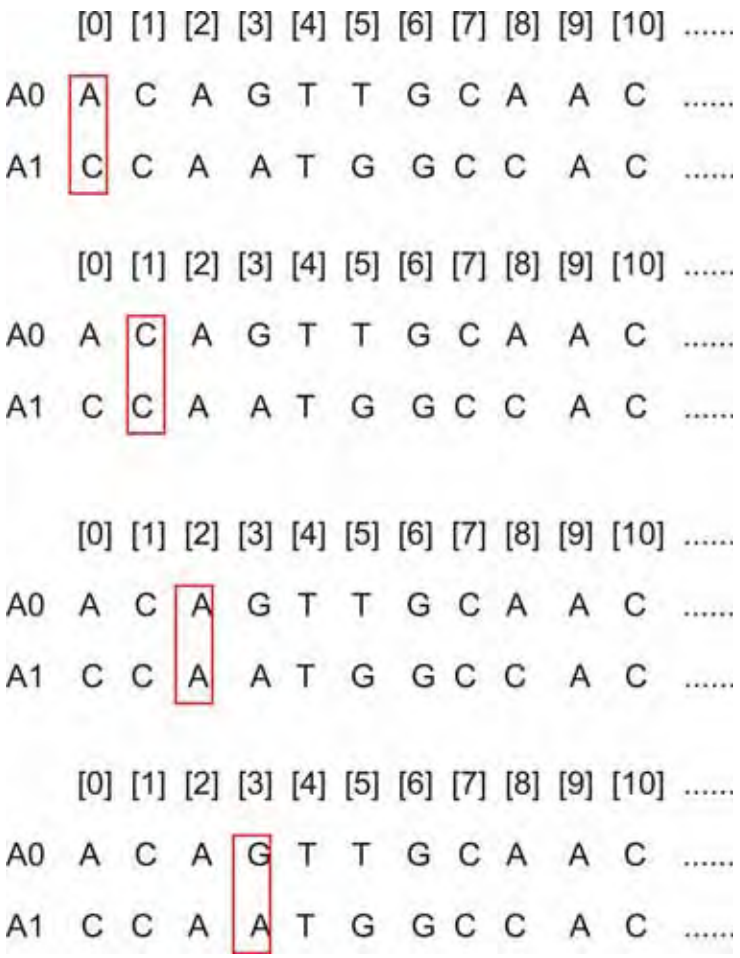

Figura 26: Ilustração da execução do alinhamento múltiplo com *Ant Colony* no primeiro passo.

Na figura 27, pode ser verificado o segundo passo do alinhamento múltiplo, em que mais uma sequência é colocada para ser alinhada, agora possuindo A_0, A_1 e A_2. A cada passo, mais uma sequência é inserida para ser comparada com o conjunto alinhado existente, baseando-se na proximidade de sua localização na árvore guia, ou seja, quanto mais próxima das sequências anteriormente alinhadas, mais breve essa sequência será alinhada.

Depois que cada formiga completa uma solução, ou seja, a cada nova inserção de uma sequência no alinhamento múltiplo, as trilhas de feromônios são atualizadas de acordo

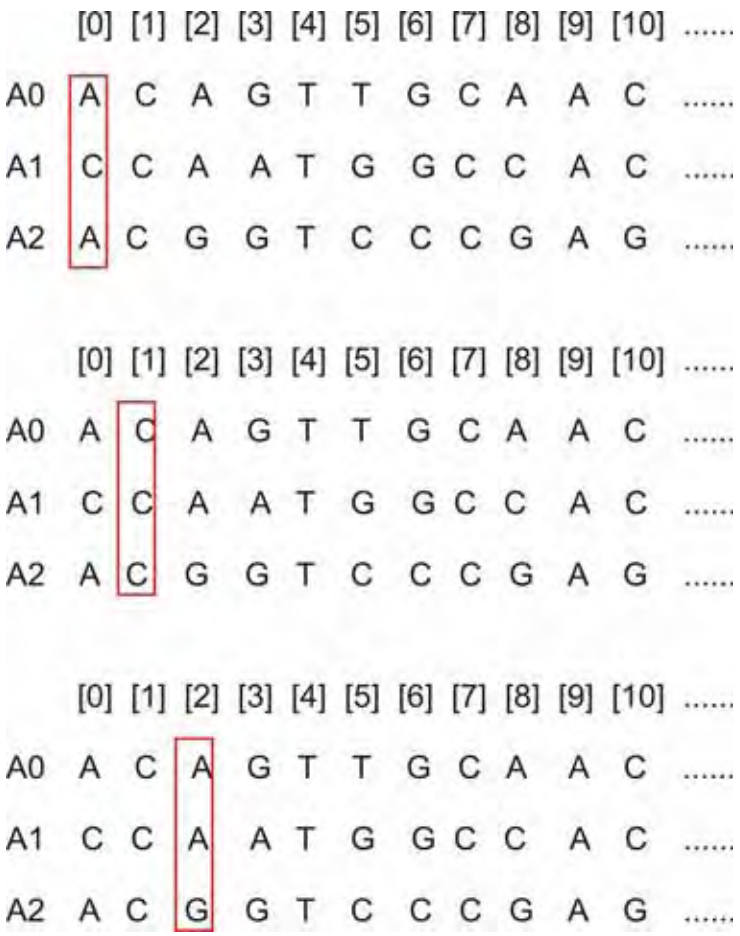

Figura 27: Ilustração da execução do alinhamento múltiplo com *Ant Colony* no segundo passo.

com um valor de pontuação daquela solução, sempre objetivando a máxima pontuação possível. A nova trilha é calculada pela função de feromônio [58]:

$$fero(k, l, m, n) = fero(k, l, m, n)\ X\ (1 - evap) + (score - media)\ X\ evap$$

Na função $fero(k, l, m, n)$ mostrada anteriormente, k é a quantidade de sequências envolvidas, l o comprimento da maior sequência na iteração, m o comprimento da maior sequência entre todas e n o comprimento da menor sequência entre todas. O valor $evap$ é o coeficiente de evaporação, que varia entre 0 e 1, o valor $score$ é computado a cada novo alinhamento, devido à mudança do valor de pontuação do alinhamento, e o valor $media$ é o valor médio dos valores $score$ dos alinhamentos anteriores.

Na figura 28 pode ser visto o fluxograma do funcionamento das colônias de formigas para a fase de alinhamento múltiplo.

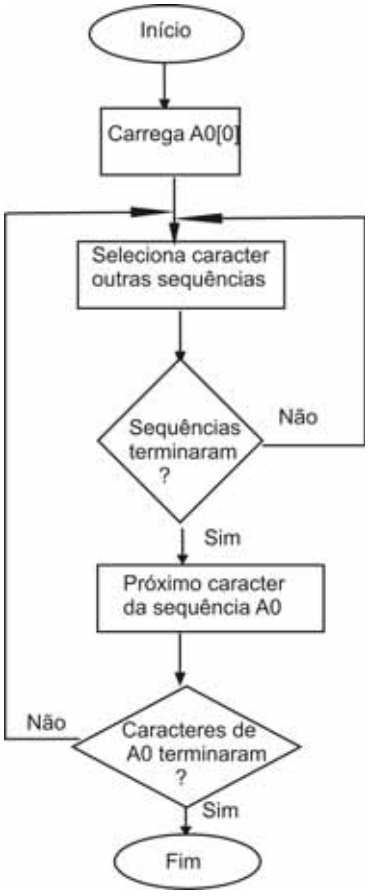

Figura 28: Fluxograma para o alinhamento múltiplo utilizando Colônias de Formigas.

4 Resultados

4.1 Algoritmo de Alinhamento Progressivo

As seções a seguir trazem diversos testes que foram realizados com o algoritmo de alinhamento múltiplo progressivo para diferentes configurações. Alguns testes foram feitos antes de ser inserida a técnica de colônia de formigas e, posteriormente, com o uso da técnica e mais algumas variações.

Para demonstrar a eficiência do algoritmo de alinhamento múltiplo progressivo proposto e a sua qualidade em relação aos alinhamentos finais, alguns testes foram feitos para comprovação.

4.1.1 Desempenho do Algoritmo

Os testes de desempenho do algoritmo objetivam mostrar que a forma como o algoritmo foi construído e o paralelismo utilizado nessa abordagem proposta do alinhamento múltiplo progressivo, promovem um ganho de desempenho, reduzindo o tempo de processamento, conforme o número de processadores vai sendo ampliado.

Os testes foram conduzidos em dois *clusters Beowulf* ambos com 9 nós operacionais, um deles localizado dentro do Laboratório de Bioinformática da Unesp/São José do Rio Preto (*cluster Rubão Dog*) e o outro, localizado no Instituto de Estudos Avançados (IEAv) do Comando-Geral de Tecnologia Aeroespacial - CTA/São José dos Campos (*cluster Believe*).

Os valores de tempo de execução podem variar, quando se leva em consideração o tipo

de equipamento utilzado para as execuções. Máquinas mais poderosas certamente levarão a uma redução no tempo de execução quando comparados os resultados com máquinas menos poderosas.

Na figura 29, mostra-se o gráfico comparativo da execução do alinhamento múltiplo para diferentes tamanhos de conjuntos de sequências, com a mesma quantidade de resíduos. Os conjuntos escolhidos tinham, respectivamente 71 sequências, 44 sequências, 27 sequências. Cada uma dessas sequências possui 550 nucleotídeos (resíduos). A escolha por essas sequências deu-se através de análises prévias das necessidades dos biólogos do Laboratório de Estudos Genômicos (LEGO/IBILCE/UNESP), assim como das demais sequências utilizadas em todos os testes. Nesse gráfico, as sequências que foram comparadas são sequências de nucleotídeos.

Percebe-se, acompanhando o gráfico da figura 29, que existe um queda acentuada do tempo de execução quando o número de processadores é aumentado. Quando se passa da execução sequencial, com um único processador, para a execução em paralelo com dois processadores, é onde existe a maior acentuação para todas as sequências analisadas. Isso enquadra-se dentro do esperado, visto que o paralelismo oferecido pelo algoritmo visa reduzir o tempo de processamento. Logo, esse ponto, com dois processadores, é onde se começa utilizar o paralelismo. Verifica-se também, que com o aumento do número de sequências, mesmo com o algoritmo paralelo, o tempo de processamento também cresce.

A figura 30 traz um gráfico semelhante ao da figura 29. No entanto, eles diferem primeiramente no tempo de processamento e segundo, e o mais importante, no tipo de sequência que é analisada. Na gráfico da figura 30 são analisadas sequências de aminoácidos. A redução no tempo de processamento dos aminoácidos, em relação as núcleotídeos, para as mesmas quantidades de sequências, de um mesmo organismo, deve-se pois, a quantidade de comparações nos aminoácidos ser menor. Verifica-se também na curva apresentada na figura 30, a mesma característica da curva apresentada na figura 29, que a redução do tempo de execução conforme o número de processadores é aumentado.

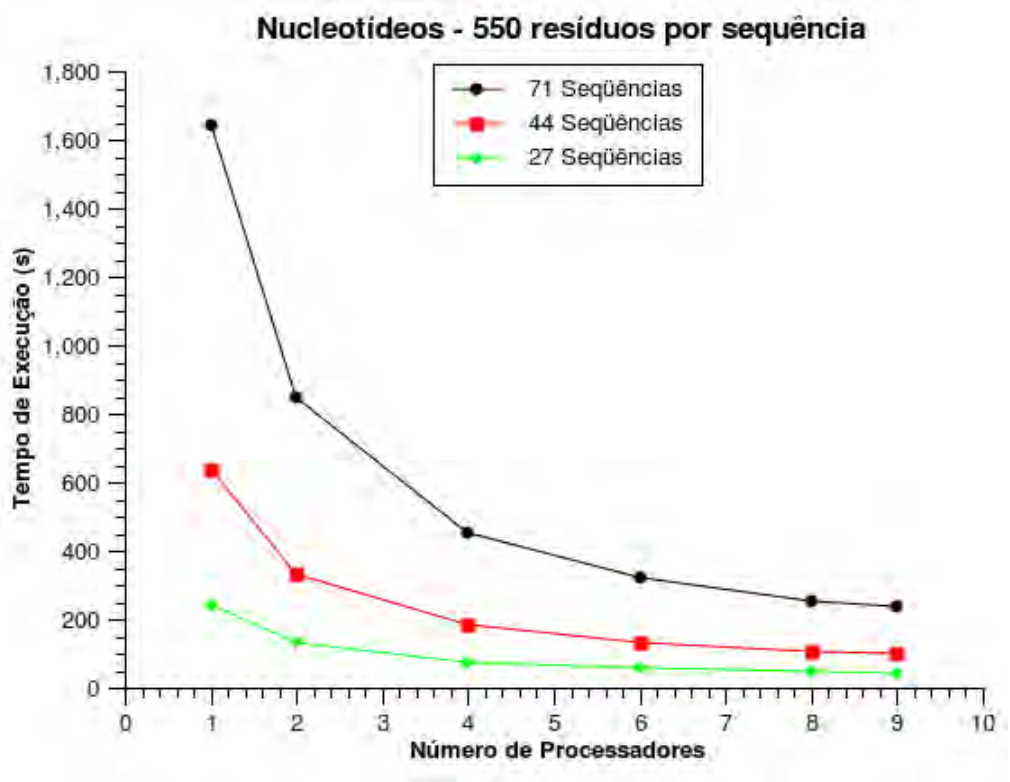

Figura 29: Gráfico que compara número de processadores com tempo de execução para nucleotídeos.

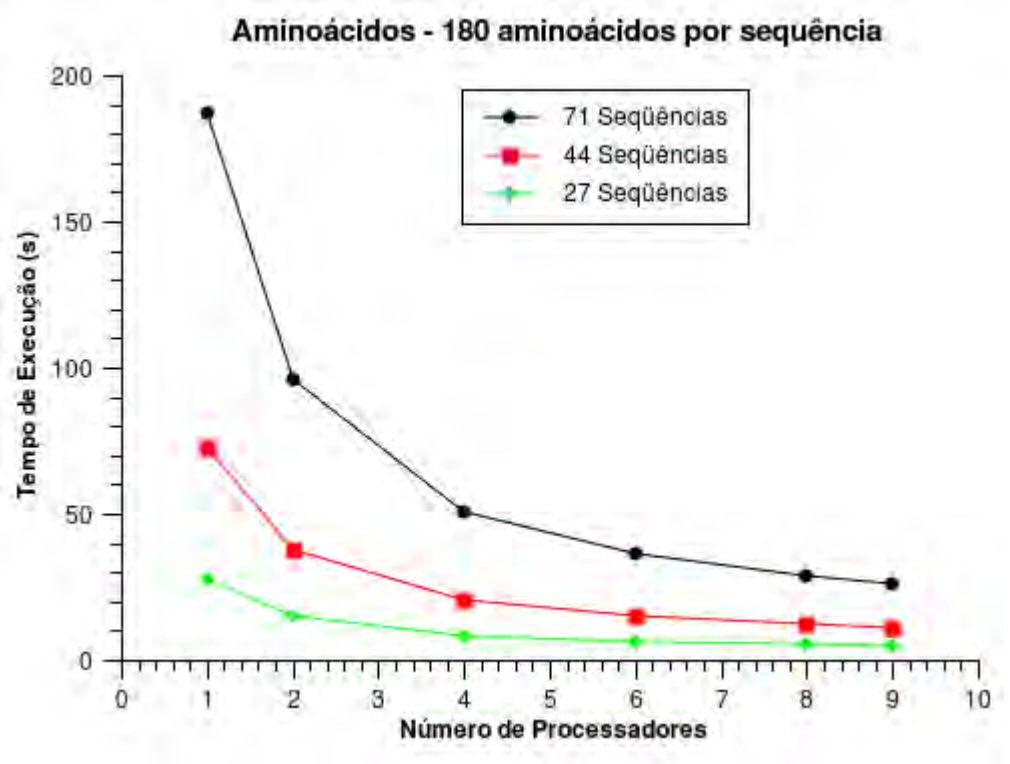

Figura 30: Gráfico que compara número de processadores com tempo de execução para aminoácidos.

Tanto as curvas mostradas na figura 29, como na figura 30 servem para corroborar que o algortimo de alinhamento múltiplo proposto trabalha conforme o esperado, ocorrendo a redução do tempo de execução, com o aumento do número de processadores.

A figura 31 traz uma comparação entre o tempo consumido por um dos estágios do algoritmo em relação ao tempo total. Esse estágio é o do alinhamento par-a-par. As barras vermelhas representam o alinhamento par-a-par e as barras pretas representam o alinhamento total. Percebe-se pelo gráfico da figura 31 que essa fase consome mais de 90% do tempo de execução do algoritmo completo. Assim, qualquer ganho de tempo de execução que se consiga na fase do alinhamento par-a-par, mostra-se muito interessante para o tempo total do alinhamento.

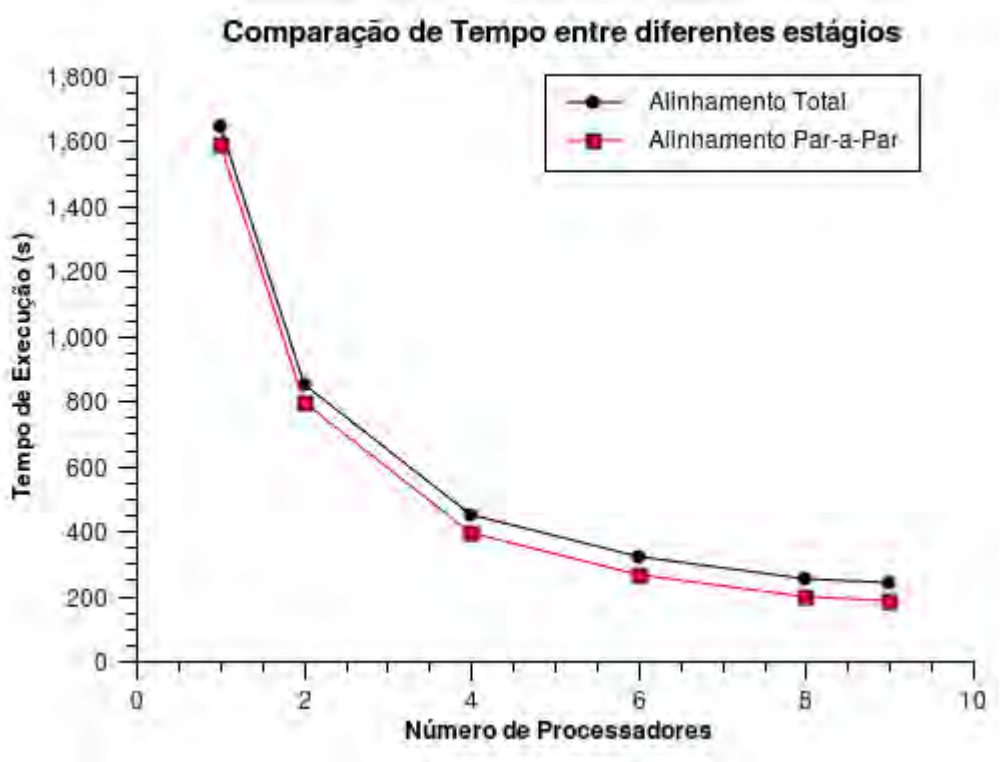

Figura 31: Gráfico que compara número de processadores com tempo de execução em relação aos tempos de alinhamentos par-a-par e o tempo total do alinhamento.

Para se realizar uma comparação entre o tempo consumido pela fase de alinhamento par-a-par, com um outra fase do algoritmo, gerou-se o gráfico da figura 32, que a compara com a fase de alinhamento múltiplo. Percebe-se que o tempo de execução consumido pela fase de alinhamento múltiplo é bem menor quando comparado com o do alinhamento

par-a-par. Essa diferença entre as duas fases reduz a sua acentuação quando o número de processadores é aumentado. Essa redução, pois a fase de alinhamento múltiplo se mantém constante em relação ao tempo de execução com a variação do número de processadores, e a fase de alinhamento par-a-par sofre uma redução de tempo de execução com o acréscimo do número de processadores.

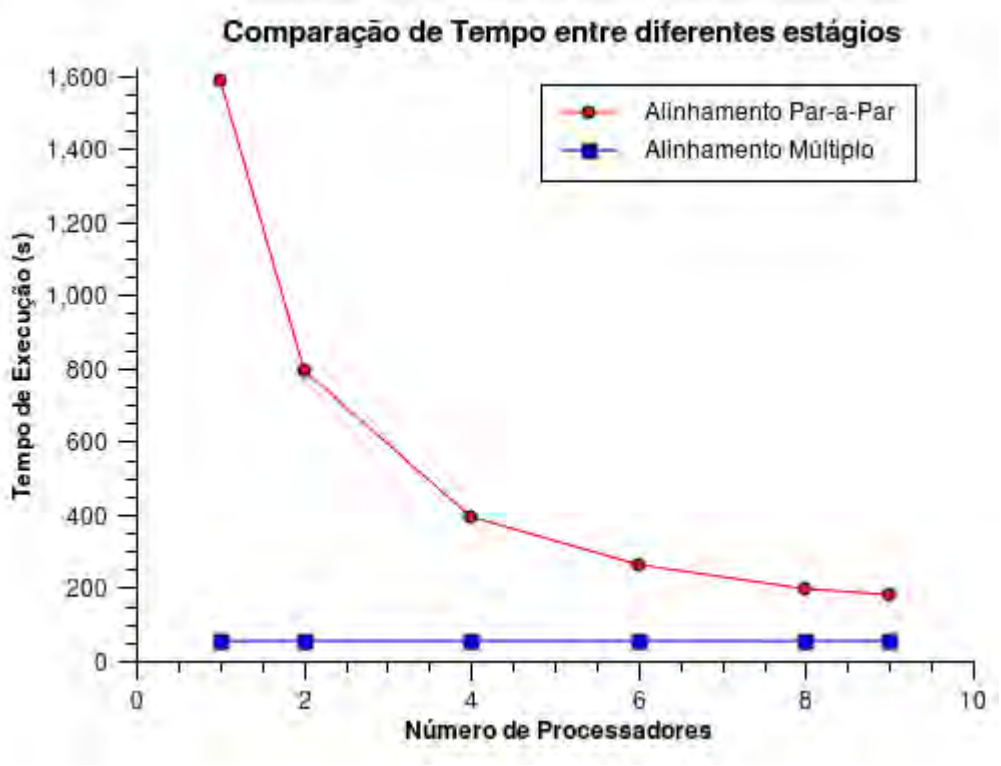

Figura 32: Gráfico que compara número de processadores com tempo de execução em relação aos tempos de alinhamentos par-a-par e a fase de alinhamento múltiplo.

O gráfico mostrado na figura 33 vem com o intuito de demostrar o que foi citado no parágrafo anterior, com relação à característica do tempo de execução da fase de alinhamento múltiplo. Esta se mantém constante independente do número de processadores envolvidos, visto que não é uma fase paralelizada. Ela varia conforme o número de sequências também varia, como pode ser observado no gráfico da figura 33, aumentando o seu tempo de execução à medida que o número de sequências aumenta.

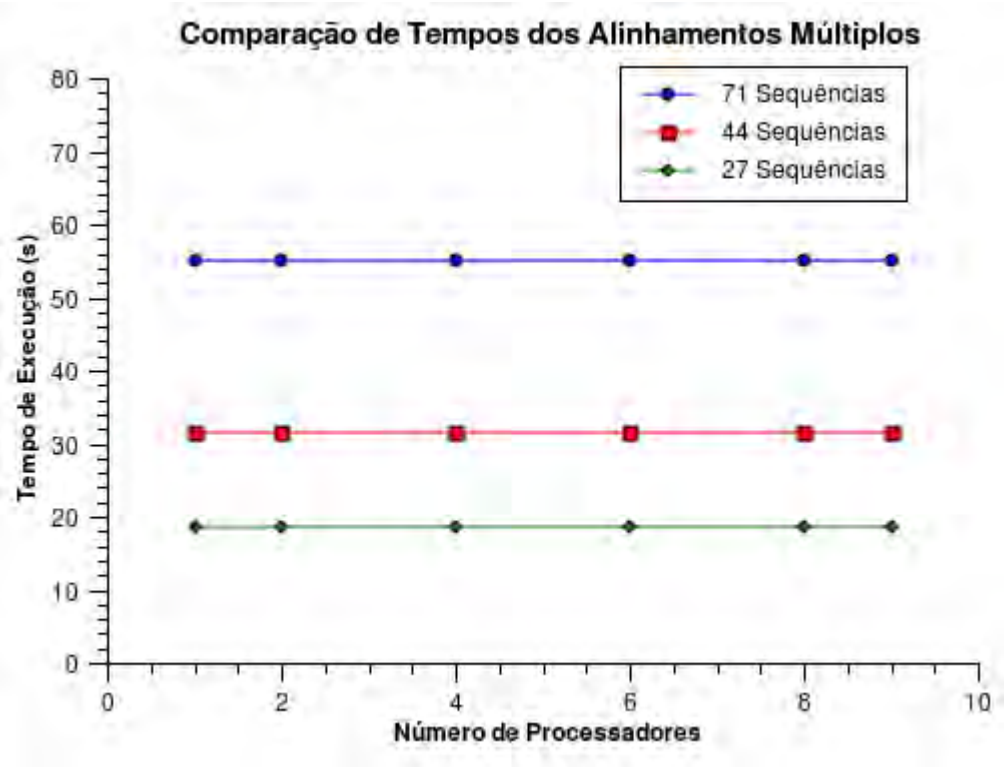

Figura 33: Gráfico que compara número de processadores com tempo de execução em relação aos tempos dos alinhamentos múltiplos para quantidades diferentes de sequências.

4.1.2 Qualidade dos Alinhamentos

Esta seção objetiva mostrar que a proposta de algoritmo de alinhamento múltiplo sugerida e que foi posteriormente otimizada, gera resultados biológicos confiáveis.

A escolha das sequências de aminoácidos para serem analisadas obedeceu um caráter aleatório e foi realizado no site *BaliBase* [59] [1]. O *BaliBase* trata-se de um banco de dados de proteínas alinhadas, em que cada alinhamento dos conjuntos de sequências de aminoácidos é feito manualmente, através do estudo estrutural da proteína. Isso garante que os alinhamentos do *BaliBase* sejam alinhamentos 100% confiáveis.

Para a comparação, criou-se a tabela 3 com resultados de análises feitas com o algoritmo proposto e com programas de alinhamentos múltiplos bastante conhecidos, como o Dialign [60], o ClustalX [61] e o Saga [62]. Cada um desses programas têm suas características particulares, mas todos têm o mesmo foco, que são os alinhamentos múltiplos.

[1] http://www-bio3d-igbmc.u-strasbg.fr/balibase/

Tabela 3: Tabela comparativa com resultados de alguns programas de alinhamento múltiplo.

Genes	**Num. Seq.**	**Proposta**	**Dialign**	**ClustalX**	**Saga**
1ad3_ref1	4	0,899	0,998	1,000	0,877
1cpt_ref1	9	0,845	0,234	0,000	0,520
1fmb_ref4	9	0,852	1,000	0,866	0,851
1ubi_ref3	48	0,786	0,000	0,650	0,000
1thm_ref5	49	0,775	0,543	0,234	0,634

Os valores mostrados na tabela são resultados da análise comparativa feita com o programa *bali_score* [2] com relação a cada um dos alinhamentos produzidos pelos programas. O *bali_score* é um programa criado pelos desenvolvedores do *BaliBase* que compara o grau de qualidade dos alinhamentos produzidos pelos programas existentes, em relação ao mesmo alinhamento feito manualmente no *BaliBase*. Quanto mais próximo de 100% os valores da comparação alcançarem, mais bem qualificados serão os seus alinhamentos. Na tabela 3 são colocados os valores relativos, bastando multiplicá-los por 100 (cem) para obter os valores em percentual.

Cada um dos programas utilizados para comparação objetiva a pontuação máxima do alinhamento entre as biossequências. No entanto, verifica-se uma diferença entre a abordagem prosposta e os demais programas na tabela 3. Percebe-se que a abordagem proposta sofre menos variações na pontuação final encontrada, do que a análise feita pelos demais programas. Isso não significa que os outros programas são ruins ou inconstantes e isso pode ser explicado, considerando que a abordagem proposta objetiva analisar de forma mais global as biossequências, através de uma heurística mais generalizada, não buscando por uma determinada característica. Os demais programas apresentados, em geral, comportam-se muito bem para determinados casos, não mantendo uma uniformidade nos valores, pois prezam por procurar algumas características em específico em sua análise.

[2] http://www-bio3d-igbmc.u-strasbg.fr/balibase/BalibaseDownload/

4.2 Algoritmo com a Otimização

As seções seguintes mostram os testes realizados para checar desempenho e qualidade dos alinhamentos com o algoritmo utilizando uma estratégia de otimização com estimativas de pontuações e colônias de formigas.

4.2.1 Desempenho do Algoritmo

Novamente, os testes de desempenho do algoritmo foram conduzidos nos mesmos equipamentos anteriormente citados na seção 4.1.1 e também com as mesmas sequências utilizadas.

Na figura 34, pode-se observar como foi o comportamento em termos do tempo de execução da proposta otimizada do algoritmo de alinhamento múltiplo com relação à variação do número de processadores. Percebe-se claramente uma redução no tempo de execução total do algoritmo, se comparado com o gráfico da figura 29 que mostra o alinhamento múltiplo sem as otimizações.

A proporcionalidade de queda no tempo de execução conforme o número de processadores é aumentado, também existe aqui, pois, assim como o alinhamento par-a-par, o cálculo das estimativas de pontução também é paralelizado.

Na figura 35 pode-se verificar também o desempenho com relação ao tempo de execução para sequências de aminoácidos. Novamente, mantem-se a tendência também apresentada no gráfico da figura 30, em que os tempos de execução para os alinhamentos de sequências de aminoácidos são menores, quando comparados com os tempos dos nucleotídeos. Da mesma maneira, verifca-se uma redução de tempo de tempo de execução significativa no gráfico da figura 35 em relação ao gráfico da figura 30.

O gráfico de barras da figura 36 mostra a comparação entre o tempo de execução consumido pela fase de estimativa de pontuação em relação ao tempo total do alinhamento. Pode ser feita uma análise desse gráfico com o da figura 31 que compara a fase

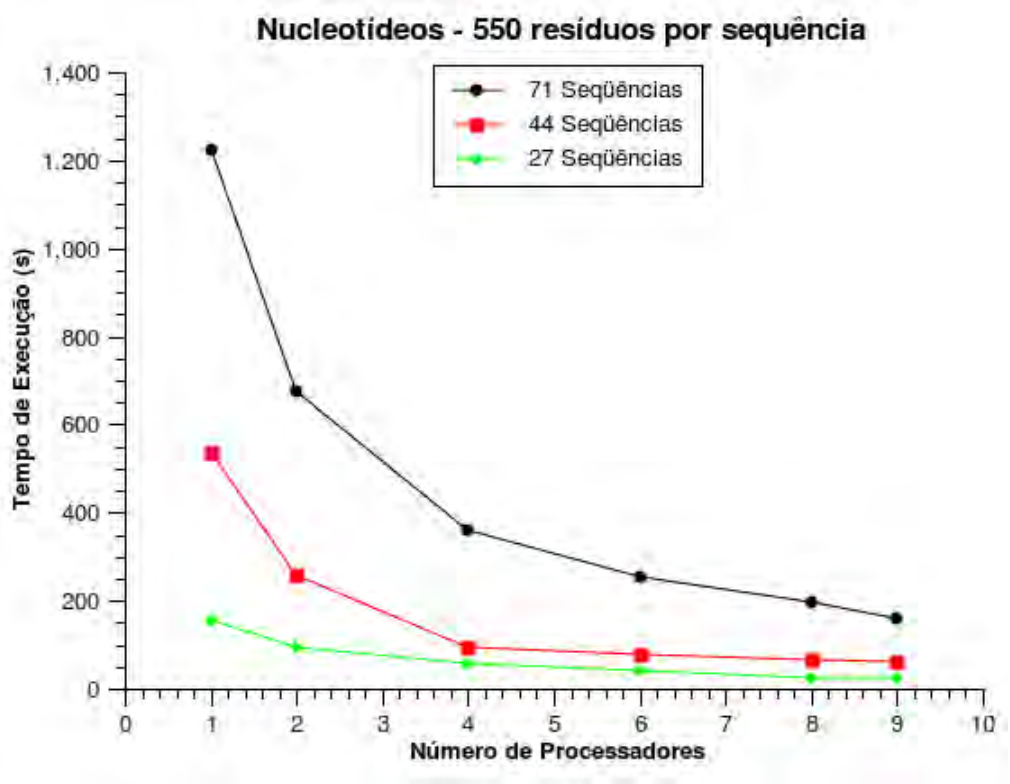

Figura 34: Gráfico que compara número de processadores com tempo de execução para nucleotídeos com as otimizações implementadas.

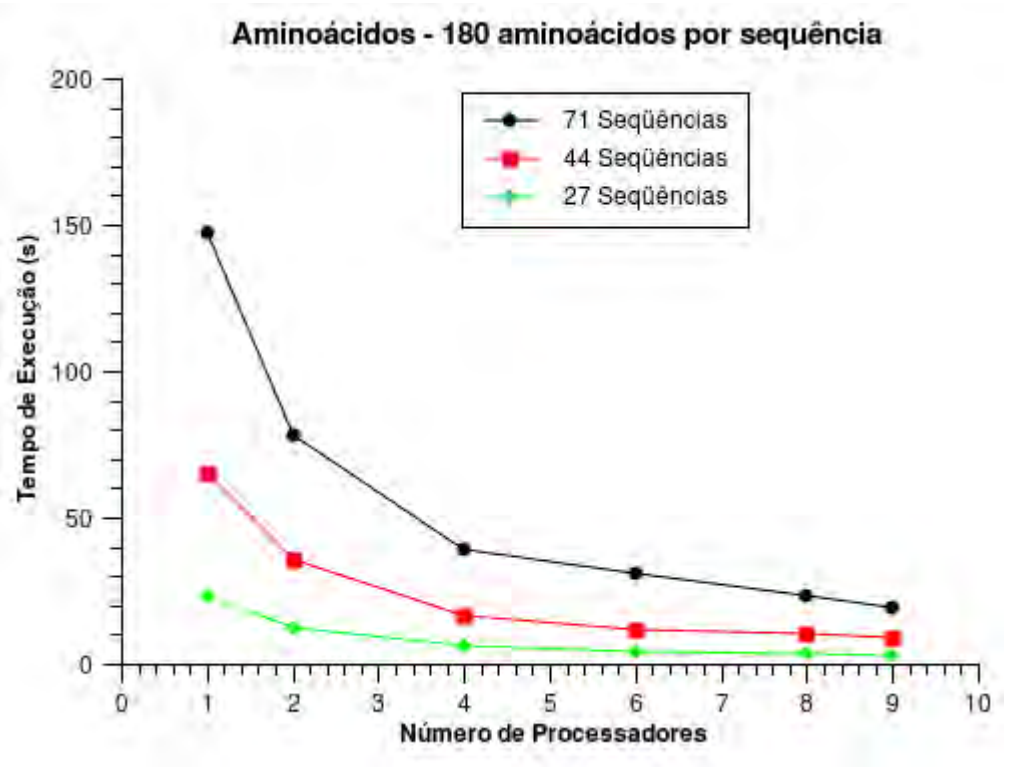

Figura 35: Gráfico que compara número de processadores com tempo de execução para aminoácidos com as otimizações implementadas.

do alinhamento par-a-par, com o tempo total do alinhamento. Considerando-se as fases de estimativas de pontuação e a de alinhamento par-a-par como equivalentes, verifica-se uma redução bastante significativa no tempo de execução do gráfico da figura 31 para o gráfico da figura 36.

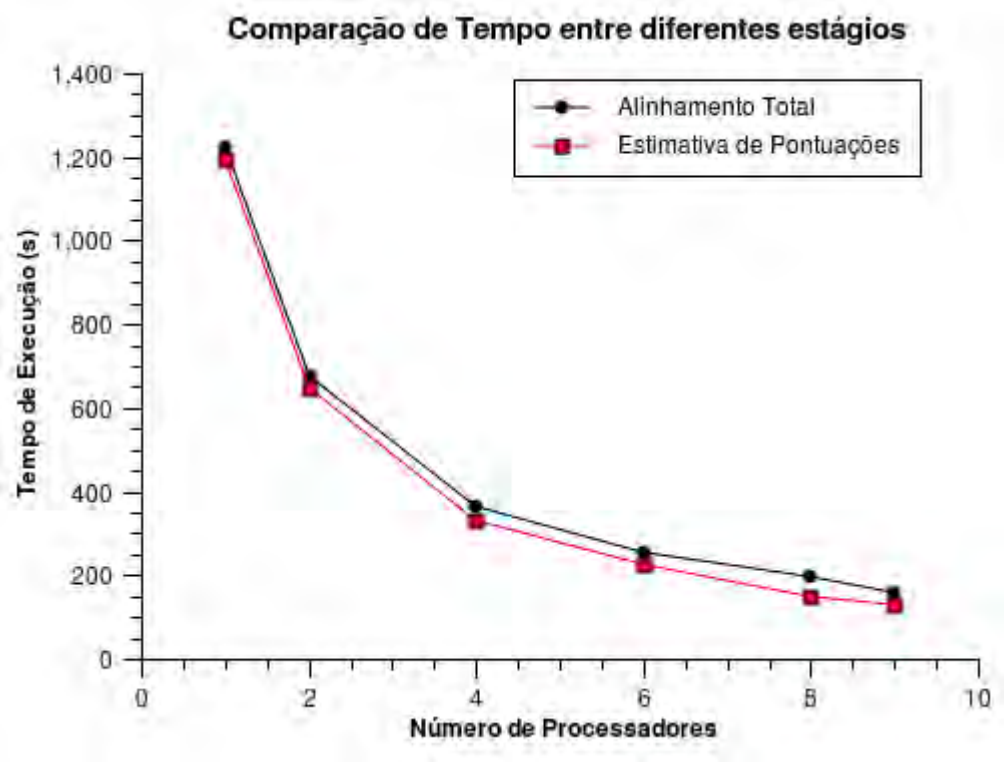

Figura 36: Gráfico que compara número de processadores com tempo de execução em relação aos tempos do cálculo da estimativa de pontuação e o tempo total do alinhamento.

Na figura 37, apresenta-se um gráfico com as comparações de tempo de execução pelo número de processadores para as duas fases da proposta otimizada do algoritmo. Nesse caso, percebe-se que a fase de estimativas de pontuações é muito mais longa do que a fase de alinhamento múltiplo. No entanto, percebe-se uma diminuição da proporção de tempo consumido pelo alinhamento múltiplo nessa nova abordagem com colônias de formigas, quando comparado com a aborgagem puramente progressiva, apresentada no gráfico da figura 32.

No gráfico apresentado pela figura 38, são apresentados os tempos de execução consumidos pelo alinhamento múltiplo utilizando colônias de formigas com relação ao número de processadores. Verifica-se, da mesma maneira que no gráfico da figura 33, uma cons-

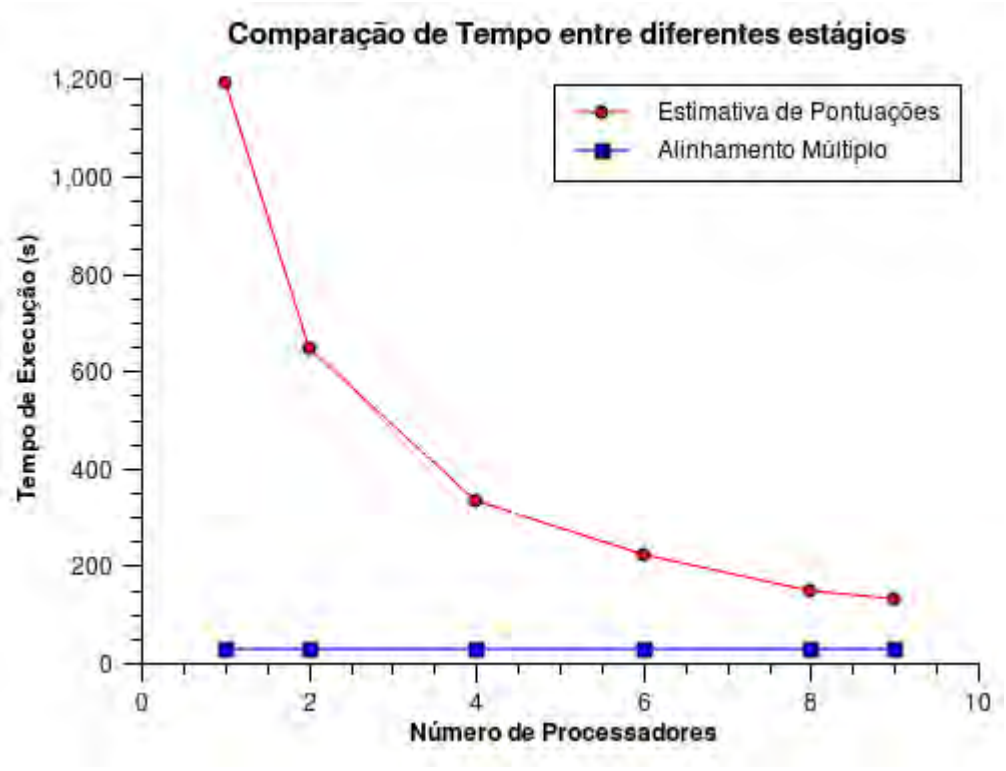

Figura 37: Gráfico que compara número de processadores com tempo de execução em relação aos tempos do cálculo da estimativa de pontuação e o alinhamento múltiplo.

tância nos valores de tempo da parte de alinhamento múltiplo. O que se percebe no gráfico da figura 38 é uma redução de tempo com relação ao gráfico da figura 33 quando confrontados para as mesmas sequências.

Para ilustrar um pouco mais essa idéia do ganho de desempenho do algoritmo otimizado em relação à aborgadagem progressiva pura, alguns gráficos em barra foram gerados, comparando a mesma execução para os mesmos conjuntos de sequências de nucleotídeos e aminoácidos com as diferentes estratégias de algoritmos.

O gráfico da figura 39 mostra a comparação do tempo total para a execução dos alinhamentos de sequências de nucleotídeos com 71 sequências, contendo 550 resíduos por sequência, para as duas abordagens algorítmicas. Percebe-se que as barras em cor laranja, representando o algoritmo otimizado, conseguem um melhor desempenho em termos de tempo, do que a abordagem progressiva.

No gráfico da figura 40, percebem-se características muito semelhantes às identificadas

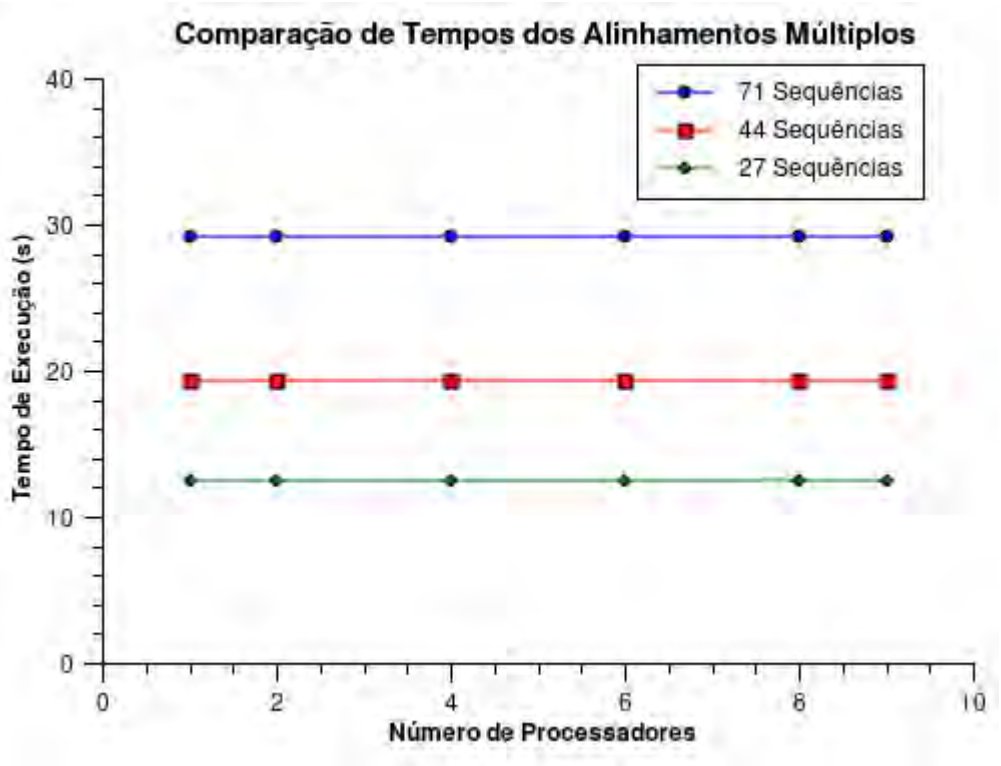

Figura 38: Gráfico que compara número de processadores com tempo de execução em relação aos tempos do alinhamento múltiplo para quantidades diferentes de sequências para o algoritmo utilizando Colônias de Formigas.

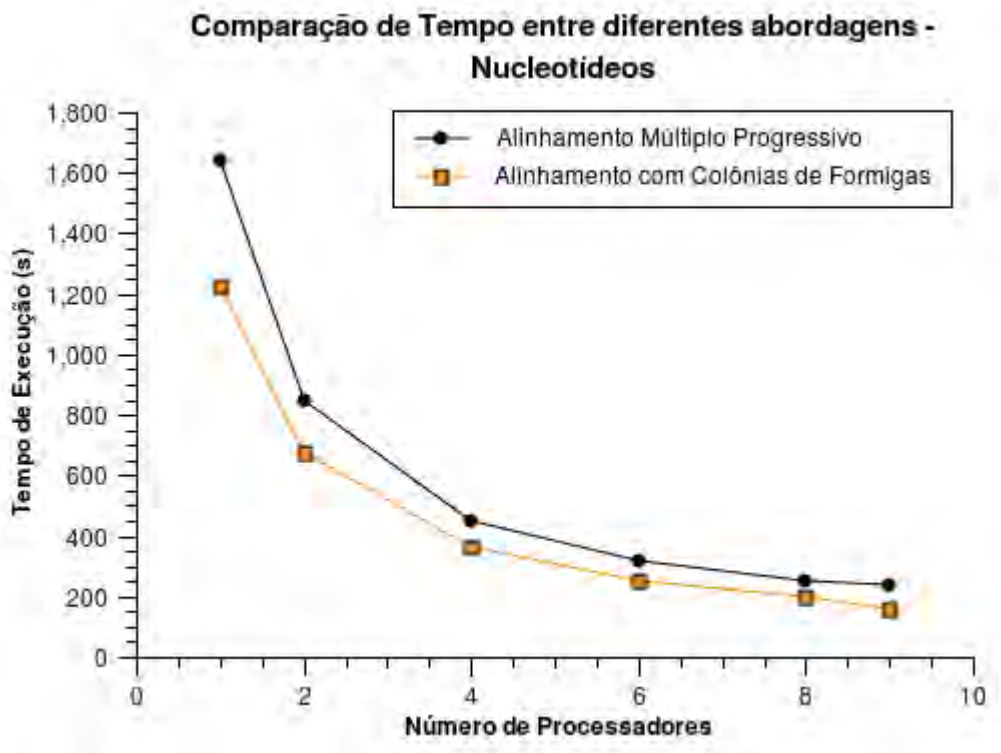

Figura 39: Gráfico que compara número de processadores com tempo de execução em relação aos tempos total dos alinhamento para as diferentes abordagens de algoritmos para nucleotídeos.

no gráfico da figura 39, com a redução de tempo de execução do algoritmo otimizado, em relação ao puramente progressivo. No entanto, na figura 40 é mostrado um gráfico dos tempos de execução para sequências de aminoácidos.

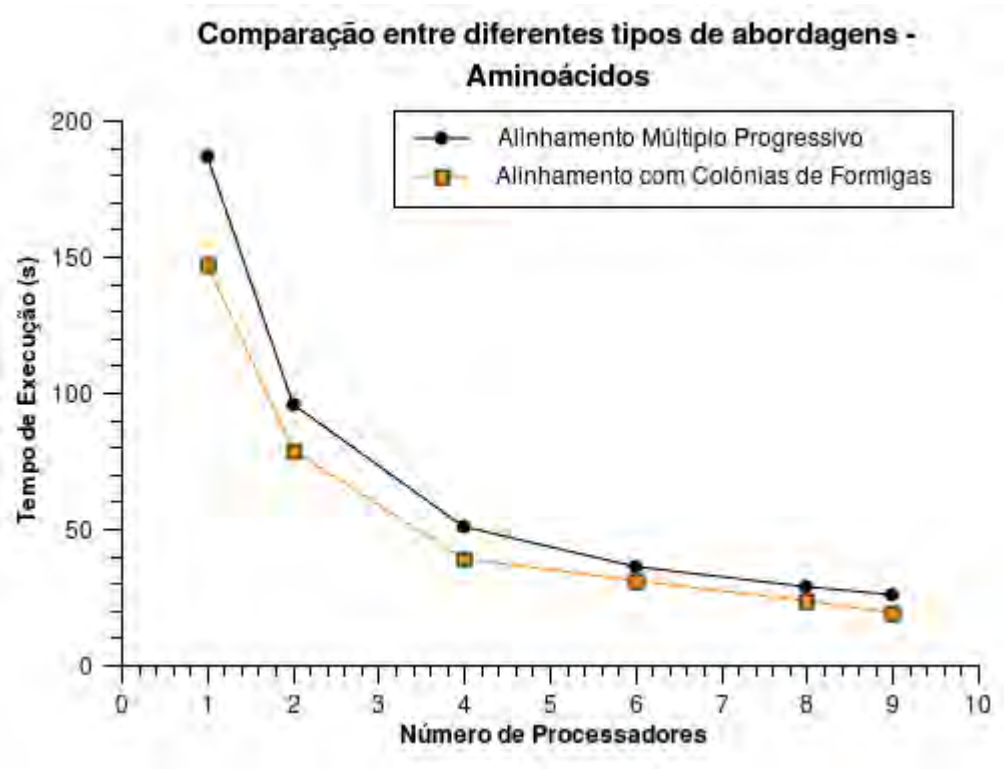

Figura 40: Gráfico que compara número de processadores com tempo de execução em relação aos tempos total dos alinhamento para as diferentes abordagens de algoritmos para aminoácidos.

Tanto no gráfico da figura 39, quanto no gráfico da figura 40, a convergência do valores de tempo de execução é a esperada quando o número de processadores é aumentado, verificando-se que as proporções dos tempos de execução entre os algoritmos se mantém em ambos os gráficos.

Assim, encerra-se a parte de testes de desempenho para o algoritmo otimizado, restando apenas a parte de testes da qualidade dos alinhamentos realizados por ele.

4.2.2 Qualidade dos Alinhamentos

Esta seção, tem por objetivo mostrar a qualidade dos alinhamentos produzidos, assim como feito na seção 4.1.2.

As mesmas sequências utilizadas para a análise na seção 4.1.2 foram utilizadas nesta seção e juntamente com o programa *bali_score* [3] para se realizar a análise comparativa.

Para essa comparação, criou-se a tabela 4 com resultados de análises feitas, novamente, por programas de alinhamentos múltiplos bastante conhecidos, como o Dialign [60], o ClustalX [61] e o Saga [62] que foram mostrados anteriormente na tabela 3. Substituiu-se a coluna relativa à proposta de algoritmo puramente progressivo, pelos resultados da abordagem otimizada.

Percebe-se pela tabela 4 que o algoritmo otimizado também acompanha a tendência da proposta puramente progressiva que é manter uma característica mais uniforme nos resultados, quando comparado com outros programas de alinhamentos e, além disso, com todos os valores acima de 70% de similaridade em relação aos do *BaliBase* [59] [4]. Esses resultados vêm para reafirmar que os programas de alinhamentos mostrados se comportam bem para determinados casos, mas para outros não. Com isso, mostra-se que a abordagem otimizada também analisa de forma mais global as biossequências, através de uma heurística mais generalizada, não buscando por uma determinada característica.

Tabela 4: Tabela comparativa com resultados de alguns programas de alinhamento múltiplo e da abordagem otimizada.

Genes	**Num. Seq.**	**Otimização**	**Dialign**	**ClustalX**	**Saga**
1ad3_ref1	4	0,929	0,998	1,000	0,877
1cpt_ref1	9	0,787	0,234	0,000	0,520
1fmb_ref4	9	0,882	1,000	0,866	0,851
1ubi_ref3	48	0,797	0,000	0,650	0,000
1thm_ref5	49	0,794	0,543	0,234	0,634

Para finalizar a parte de comparações da qualidade dos alinhamentos, apresenta-se na tabela 5 a comparação dos resultados obtidos pelas duas abordagens algorítmicas, com relação à qualidade de seus alinhamentos. Pode-se verificar que na maior parte dos casos a abordagem otimizada conseguiu uma pequena melhora, da ordem de 0,5%, na porcentagem de similaridade do alinhamento em relação a abordagem puramente progressiva.

[3] http://www-bio3d-igbmc.u-strasbg.fr/balibase/BalibaseDownload/
[4] http://www-bio3d-igbmc.u-strasbg.fr/balibase/

Tabela 5: Tabela comparativa com resultados das duas abordagens algorítmicas propostas.

Genes	**Num. Seq.**	**Otimização**	**Proposta Progressiva**
1ad3_ref1	4	0,929	0,899
1cpt_ref1	9	0,787	0,845
1fmb_ref4	9	0,882	0,852
1ubi_ref3	48	0,797	0,786
1thm_ref5	49	0,794	0,775

Os testes de qualidade realizados compararam os resultados das abordagens propostas, com os resultados de programas de alinhamento múltiplo de sequências bastante conhecidos e utilizados pela comunidade científica. No entanto, isso não significa que não existam outros programas de alinhamentos múltiplos de sequências e que os resultados das abordagens propostas também possam ser comparados com eles.

5 Conclusões

Este trabalho apresentou uma visão de como se encontra a bioinformática, mais especificadamente com relação às técnicas para alinhamentos múltiplos progressivos e otimizações utilizadas para melhorar essas técnicas.

Assim, o presente trabalho procurou descrever um algoritmo de alinhamento múltiplo progressivo convencional e o uso de uma dessas técnicas de otimização, que no caso é a de colônias de formigas, utilizando-a para produzir uma otimização em algoritmos de alinhamentos múltiplos, mas mantendo a qualidade dos resultados finais dos alinhamentos. Além disso, realizou-se algumas melhorias que podem ser utilizadas também nos algoritmos de alinhamentos múltiplos progressivos.

Os testes mostraram que o algoritmo proposto sem a otimização se comporta bem quando o número de processadores é aumentado, gerando através de um paralelismo de grão grosso, uma redução bastante significativa no tempo do alinhamento. Os gráficos apresentados na seção de testes mostram bem essas características descritas. O uso da técnica de otimização com colônias de formigas e estimativas de pontuação contribuiu para a melhora do desempenho do algoritmo, reduzindo o tempo de execução para a realização dos alinhamentos, como também mostrado nos gráficos da seção de testes.

Quanto a qualidade dos alinhamentos de ambas as abordagens propostas, as comparações com o outros programas de alinhamento, utilizando o *bali_score*, mostraram que essas abordagens se enquadram bem com os propósitos dos alinhamentos de biossequências e, através de suas heurísticas mais generalizadas, obtêm soluções mais generalizadas, não se focando em analises bem particulares como uma boa parte dos programas de alinhamentos

múltiplos existentes.

Com todo o aparato algorítmico apresentado neste trabalho, conclui-se que a estratégia otimizada do algoritmo de alinhamento múltiplo conseguiu bons resultados e pode tornar-se fonte de estudos para outros trabalhos.

5.1 Trabalhos Futuros

Como um dos trabalhos futuros, uma primeira ideia é produzir uma interface gráfica amigável para essa ferramenta, pois na versão atual ela se encontra totalmente em modo texto. Em geral, os principais usuários desse tipo de ferramenta são biólogos, e proporcionar uma interface agradável a eles é algo que estimulará bastante o seu uso. Além disso, a interface pode oferecer mais algumas funcionalidades de visualização dos resultados, sem ser somente através de um arquivo de texto plano como ela figura atualmente.

Uma outra ideia que pode ser considerada como trabalho futuro é portar todo o código criado em C++ com MPI, para uma linguagem multiplataforma, como JAVA e testar o seu desempenho utilizando as bibliotecas de troca de mensagem implementadas em JAVA.

Além dessas duas possibilidades colocadas anteriormente, pode-se acrescentar à ferramenta um mecanismo de busca de padrões, que agregado ao alinhamento múltiplo, pode contribuir ainda mais para os biólogos, ou para quem vier a trabalhar com ela. Essa busca auxiliará na identificação dos chamados *hot spots* (pontos quentes), que são pontos que podem representar uma determinada característica do gene. Evita-se com isso, que o biólogo tenha que realizar esse reconhecimento de padrões sobre o arquivo alinhado.

Referências

[1] LEMOS, M.; CASANOVA, M. A. *Algoritmos para análise de seqüências*. [S.l.], 2000.

[2] MOSS, J.; JOHNSON, C. G. An ant colony algorithm for multiple sequence alignment in bioinformatics. *Artificial Neural Networks and Genetic Algorithms*, p. 182–186, 2003.

[3] BACKER, B. de; FURNON, V.; SHAW, P. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Metaheuristics*, n. 6, p. 501–523, 2000.

[4] EDGAR, R. C.; BATZOGLOU, S. Multiple sequence alignment. *Current Opinion in Structural Biology*, n. 16, p. 368–373, 2006.

[5] LIEW, A. W.-C.; YAN, H.; YANG, M. Pattern recognition for the emerging field of bioinformatics: a review. *Pattern Recogntion, Science Direct*, n. 38, p. 2055–2073, 2005.

[6] AL, D. Z. et. Separation of near full-length hepatitis c virus quasispecies variants from a complex population. *Journal of Virological Methods*, n. 141, p. 220–224, 2007.

[7] WALLACE, I. M.; BLACKSHIELDS, G.; HIGGINS, D. G. Multiple sequence alignment. *Current Opinion in Structural Biology*, n. 15, p. 261–266, 2005.

[8] 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*, n. 48, p. 443–453, 1970.

[9] LEMOS, M.; ARAGãO, M. V. S. P. de; CASANOVA, M. A. *Padrões em biosseqüências*. [S.l.], 2003.

[10] EIDHAMMER, I.; JONASSEN, I.; TAYLOR, W. R. Structure comparison and structure patterns. *Journal of Computational Biology*, n. 7, p. 685–716, 2000.

[11] LEMOS, M.; BASíLIO, A.; CASANOVA, M. A. *Um estudo dos algoritmos de montagem de fragmentos de DNA*. [S.l.], 2003.

[12] RECH, D.; PILATTI, R. *992align - Uma ferramenta para alinhamento múltiplo de seqüências de DNA e proteínas*. [S.l.], 2004.

[13] BRAZMA, A. et al. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, p. 2740–2746, 1998.

[14] ROUCHKA, E. C. Aligning dna sequences using dynamic programming. *The ACM Student Magazine*, v. 3, n. 1, 2006.

[15] COHEN, J. Bioinformatics - an introduction for computer scientists. *ACM Computing Surveys*, v. 36, n. 2, p. 122–158, 2004.

[16] HENIKOFF, S.; HENIKOFF, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America - PNAS*, v. 89, n. 22, p. 10915–10919, 1992.

[17] BALDI, P. et al. Hidden markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the United States of America - PNAS*, v. 91, n. 3, p. 1059–1063, 1994.

[18] DOOLITTLE, R. F. *Molecular Evolution: Computer Analisys of Protein and Nucleic Acid Sequences*. [S.l.]: Academic Press, 1990.

[19] LIPMAN, D. J.; PEARSON, W. R. Rapid and sensitive protein similarity search. *Science*, n. 227, p. 1435–1441, 1985.

[20] PEARSON, W. R.; LIPMAN, D. J. Improved tools for biological sequence comparison. In: *Proceedings of the National Academy of Sciences of the U.S.A.* [S.l.: s.n.], 1988. p. 2444–2448.

[21] ALTSCHUL, S. F. et al. A basic local alignment search tool. *Journal of Molecular Biology*, 1990.

[22] ALTSCHUL, S. F. et al. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, n. 25, p. 3389–3402, 1997.

[23] KROGH, A. *Computational Methods in Molecular Biology*. [S.l.]: Elsevier, 1998.

[24] SIEPEL, A.; HAUSSLER, D. Combining phylogenetic and hidden markov models in biosequence analysis. In: *Proceedings of the seventh annual international conference on Research in computational molecular biology - RECOMB '03*. [S.l.: s.n.], 2003. p. 277–286.

[25] FENG, D.-F.; DOOLITTLE, R. F. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, n. 60, p. 351–360, 1987.

[26] NEI, M.; SAITO, N. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, n. 4, p. 406–425, 1987.

[27] ZOMAYA, A. Y.; ERCAL, F.; OLARIU, S. *Solutions to parallel and distributed computing problems - Lessons from biological sciences*. [S.l.]: John Wiley & Sons, 2001.

[28] QUINN, M. J. *Parallel Programming in C with MPI and OpenMP*. [S.l.]: McGraw Hill, 2004.

[29] TAN, W. B.; STRAZDINS, P. The analysis and optimization of collective communications on a beowulf cluster. In: *Proceedings of the 9th International Conference on Parallel and Distributed Systems (ICPADS'02)*. [S.l.: s.n.], 2002. p. 659– 666.

[30] HALL, M. W. et al. Detecting coarse-grain parallelism using an interprocedural parallelizing compiler. In: *Proceedings of IEEE/ACM SC95 Conference*. [S.l.: s.n.], 1995.

[31] RINARD, M. C. Effective fine-grain synchronization for automatically parallelized programs using optimistic synchronization primitives. *ACM Transactions on Computer Systems (TOCS)*, 1999.

[32] KRANZLMüLLER, D. et al. (Ed.). *Recent Advances in Parallel Virtual Machine and Massage Passing Interface*, v. 2474 de *Lecture Notes in Computer Science*, (Lecture Notes in Computer Science, v. 2474). 9th European PVM/MPI User's Group Meeting, [S.l.]: Springer, 2002.

[33] PACHECO, P. S. *Parallel Programming with MPI*. [S.l.]: Morgan Kaufmann, 1997.

[34] KIM, J.; PRAMANIK, S.; CHUNG, M. J. Multiple sequence alignment using simulated annealing. *Oxford Journals - Bioinformatics*, v. 10, p. 419–426, 1994.

[35] RIAZ, T.; WANG, Y.; LI, K.-B. Multiple sequence alignment using tabu search. In: *Proceedings of the 2nd conference on Asia-Pacific bioinformatics*. [S.l.: s.n.], 2004. p. 223–232.

[36] GUPTA, S. K.; KECECIOGLU, J. D.; SCHAFFER, A. A. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, v. 2, n. 3, p. 459–472, 1995.

[37] HO, S. L. et al. A modified ant colony optimization algorithm modeled on tabu-search methods. *IEEE Transactions on Magnetics*, v. 42, n. 4, 2006.

[38] COMBS, W. et al. Tho course scheduling problem as a source of student project. In: *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. [S.l.: s.n.], 2005.

[39] DORIGO, M.; CARO, G. D.; GAMBARDELLA, L. M. Ant algorithms for discrete optmization. *Artificial Life*, p. 137–172, 1999.

[40] GLOVER, F.; LAGUNA, M. *Tabu Search*. [S.l.]: Kluwer Academic Publishers, 1997.

[41] KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optmization by simulated annealing. *Science*, n. 220, p. 671–680, 1983.

[42] KALOS, M. H. Monte carlo methods in th physical sciences. In: *Proceedings of the 2007 Winter Simulation Conference*. [S.l.: s.n.], 2007. p. 266–271.

[43] SCHWARTZ, A. S.; PATCHER, L. Multiple alignment by sequence annealing. *Bioinformatics*, v. 23, p. 24–29, 2006.

[44] WIESE, K. C.; DESCHENES, A. A.; HENDRIKS, A. G. Rnapredicted - an evolutionary algorithm for rna secondary structure prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 5, n. 1, p. 25–41, 2008.

[45] DORIGO, M.; BLUM, C. Ant colony optimization theory: A survey. *Theoretical Computer Science*, v. 344, p. 243–278, 2005.

[46] BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308, 2003.

[47] HO, S. L. et al. An improved ant colony optimization algorithm and its application to electromagnetic devices designs. *IEEE Transactions on Magnetics*, v. 41, n. 5, 2005.

[48] DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, 1997.

[49] PAPADIMITRIOU, C.; STEIGLITZ, K. *Combinatorial Optimization - Algorithms and Complexity*. [S.l.]: Dover Publications Inc., 1982.

[50] COOK, S. A.; MITCHELL, D. G. Finding hard instances of the satisfiability problem: A survey. *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, v. 35, p. 1–17, 1997.

[51] HOROWITZ, E.; SAHNI, S. Computing partitions with applications to the knapsack problem. *Journal of ACM*, v. 21, 1974.

[52] ZHANG, L.; MALIK, S. Conflict driven learning in a quantified boolean satisfiability solver. In: *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design - ICCAD '02*. [S.l.: s.n.], 2002. p. 442–449.

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

[54] KUMNORKAEW, P.; RUENGLERTPANYAKUL, W.; KU, H.-M. Application of ant colony optimization to evolutionary tree construction. In: *Proceedings of the 15th Annual Meeting of the Thai Society for Biotechnology*. [S.l.: s.n.], 2004.

[55] MUHMMAD, R. B. Parallelization of local search for euclidean steiner tree problem. In: *Proceedings of the 44th annual Southeast Regional Conference*. [S.l.: s.n.], 2006.

[56] 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.

[57] GRAS, R. et al. Cooperative metaheuristics for exploring proteomic data. *Artificial Intelligence Review*, v. 20, n. 1-2, p. 95–120, 2003.

[58] CHEN, Y. et al. Partitioned optimization algorithms for multiple sequence alignment. In: *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA '06)*. [S.l.: s.n.], 2006.

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

[60] MORGENSTERN, B. et al. Dialign: finding local similarities by multiple sequence alignment. *Bioinformatics*, v. 14, n. 3, p. 290–294, 1998.

[61] JEANMOUGIN, F. et al. Multiple sequence alignment with clustal x. *Trends in Biochemical Sciences*, v. 23, n. 10, p. 403–405, 1998.

[62] NOTREDAME, C.; HIGGINS, D. Saga: sequence alignment by genetic algorithm. *Nucleic Acid Research*, v. 24, n. 8, p. 1515–1524, 1996.