UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

ALINHAMENTO DE SEQUÊNCIAS BIOLÓGICAS EM AMBIENTE DISTRIBUÍDO

EDSON JOSÉ SAVI JÚNIOR

BLUMENAU
2007

2007/2-9

EDSON JOSÉ SAVI JÚNIOR

ALINHAMENTO DE SEQUÊNCIAS BIOLÓGICAS EM AMBIENTE DISTRIBUÍDO

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo de Tarso Mendes Luna, Orientador

BLUMENAU
2007

2007/2-9

ALINHAMENTO DE SEQUÊNCIAS BIOLÓGICAS EM AMBIENTE DISTRIBUÍDO

Por

EDSON JOSÉ SAVI JÚNIOR

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo de Tarso Mendes Luna, Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva, FURB
Membro:	Prof. Paulo César Rodacki Gomes, FURB

Blumenau, 20 de novembro de 2007

Dedico este trabalho a minha família, amigos e aos profissionais da área de bioinformática.

AGRADECIMENTOS

A Deus, por tudo o que ele representa.

À minha família, em especial aos meus pais e irmão, que sempre me apoiaram em todos os momentos.

Aos meus amigos, que sempre se apresentaram dispostos a ajudar.

Ao meu orientador, Paulo de Tarso Mendes Luna, que possibilitou o contato com profissionais da área de bioinformática e me ajudou de diversas formas para que os objetivos deste trabalho fossem alcançados.

Ao professor Paulo Fernando da Silva, que sempre se mostrou disposto a ajudar nas questões referente a sistemas distribuídos.

Aos pesquisadores Henrique César Lemos Jucá e Guilherme Arthur Gerônimo, da Universidade Federal de Santa Catarina, que me orientaram e auxiliaram nas questões referentes a bioinformática.

Passe a vida sorrindo, mesmo que isso lhe custe lágrimas.

Danilo de Oliveira

RESUMO

O alinhamento de seqüências biológicas é uma das principais operações da bioinformática, pois serve como base para outros processos nesta área. Por meio do alinhamento de seqüências é possível compará-las em busca de similaridades em suas estruturas. Essa comparação permite inferir sobre as propriedades de determinada molécula, baseando-se em propriedades conhecidas de outras. Entretanto, esta tarefa exige alto poder de processamento, sendo necessários, em geral, computadores de alto desempenho. O objetivo deste trabalho foi o de adaptar uma ferramenta de alinhamento de seqüências biológicas *open source*, conhecida por JAligner, de forma a permitir a realização do processamento do alinhamento das seqüências em ambiente distribuído, possibilitando com isso a utilização de uma infraestrutura computacional de menor custo. Para viabilizar o processamento em ambiente distribuído foram utilizadas as bibliotecas JPVM e JOMP.

Palavras-chave: Bioinformática. Seqüências biológicas. Sistemas distribuídos.

ABSTRACT

The biological sequences alignment is one of the most important operations in bioinformatics because it is suitable as a base for other processes in this area. Through the sequences alignment it is possible to compare them looking for similarities in its structures. This comparison allows the user infer on certain molecule properties, based on properties known from other molecules. However this task needs a high processing power, and its needed high performance computers. In this paper it was intended to adapt an open source biological sequences alignment tool, known by JAligner, objecting to accomplish the sequences alignment processing in a distributed environment. To accomplish the processing in a distributed environment, JPVM and JOMP parallel programming libraries were used.

Key words: Bioinformatics. Biological sequences. Distributed systems.

LISTA DE ILUSTRAÇÕES

Quadro 1 – Fragmento do gene do fator IX de coagulação humano 17
Quadro 2 – Os vinte aminoácidos naturais 18
Quadro 3 – Exemplo de possibilidades de alinhamento entre a última base de DNA das seqüências AAAC e AGC 19
Figura 1 – Inicialização da matriz do alinhamento 22
Quadro 4 – Função MAX utilizada no preenchimento da matriz do alinhamento 22
Figura 2 – Preenchimento da primeira célula da matriz 23
Figura 3 – Matriz completamente preenchida 23
Figura 4 – *Traceback* completo do alinhamento das seqüências GAATTCAGTTA e GGATCGA ... 24
Quadro 5 – Alinhamento resultante e pontuação do alinhamento entre as seqüências GAATTCAGTTA e GGATCGA 24
Figura 5 – Ferramenta JAligner 25
Quadro 6 – Matriz de pontuação BLOSUM45 26
Quadro 7 – Sintaxe de uma diretiva JOMP 30
Quadro 8 – Sintaxe da diretiva JOMP para definir uma região paralela 31
Quadro 9 – Sintaxe da diretiva JOMP para definir um *loop* paralelo 31
Quadro 10 – Sintaxe da diretiva JOMP para definir seções paralelas 31
Quadro 11 – Compilação do arquivo com as diretivas JOMP 31
Figura 6 – Fluxograma de uso do sistema JOMP 32
Quadro 12 – Parâmetro para informar quantidade de *threads* 32
Figura 7 – Casos de usos para manipulação da fila de seqüências 35
Quadro 13 – Detalhamento do caso de uso UC01 36
Quadro 14 – Detalhamento do caso de uso UC02 36
Quadro 15 – Detalhamento do caso de uso UC03 36
Figura 8 – Casos de usos relacionados com a execução do alinhamento em ambiente distribuído 37
Quadro 16 – Detalhamento do caso de uso UC04 37
Quadro 17 – Detalhamento do caso de uso UC05 38
Quadro 18 – Detalhamento do caso de uso UC06 38
Figura 9 – Caso de uso relacionados com o processamento do alinhamento em um nó do cluster 39

Quadro 19 – Detalhamento do caso de uso UC07 39
Figura 10 – Diagrama de classes 40
Quadro 20 – Funcionalidade das classes do protótipo 41
Figura 11 – Diagrama de atividades do alinhamento ambiente distribuído 42
Quadro 21 – Comando para executar o Daemon JPVM 44
Quadro 22 – Comando para executar o *console* do JPVM 44
Quadro 23 – Adicionando um computador na máquina paralela virtual 44
Figura 12 – Filas de seqüências biológicas em destaque 45
Quadro 24 – Trecho do código responsável por carregar uma seqüência biológica na fila de seqüências 46
Figura 13 – Campos *threads number* e *result directory* em destaque 47
Quadro 25 – Trecho do código da classe SmithWatermanGotoh onde são criadas as tarefas JPVM 48
Quadro 26 – Trecho do código da classe SmithWatermanCore onde é aguardado o recebimento de um pacote JPVM 48
Quadro 27 – Trecho do código da classe SmithWatermanGotoh que envia pacote para um computador da máquina virtual 49
Quadro 28 – Paralelização da fase inicial do algoritmo Smith-Waterman 50
Quadro 29 – Criando tarefa JPVM com número de *threads* específico 51
Figura 14 – Resultado do alinhamento de seqüências em ambiente distribuído 52
Quadro 30 – Exemplo de resultado do alinhamento entre seqüências 53
Figura 15 – Gráfico com resultados dos testes de *performance* do alinhamento com múltiplas *threads* 55
Quadro 31 – Configuração dos computadores utilizados nos testes 55
Figura 16 – Gráfico com resultados dos testes de *performance* do alinhamento em ambiente distribuído 56
Quadro 32 – Comparação entre as ferramentas JAligner, BLAST e FASTA 57

Lista de tabelas

Tabela 1 – Resultados dos testes de *performance* do algoritmo de alinhamento 54
Tabela 2 – Resultado dos testes de alinhamento em ambiente distribuído 56

LISTA DE SIGLAS

API – *Application Programming Interface*

BLAST – *Basic Local Alignment and Search Tool*

DNA – *Deoxyribonucleic Acid*

FASTA – *Fasta Sequence Comparison*

GB – *Gigabyte*

GHz – *Gigahertz*

JOMP – Java *Open Multi-Processor*

JPVM – Java *Parallel Virtual Machine*

MBps – *Megabit* Por Segundo

NCBI – *National Center for Biotechnology Information*

PVM – *Parallel Virtual Machine*

RAM – *Random Access Memory*

RF – Requisito Funcional

RNA – *Ribonucleic Acid*

RNF – Requisito Não Funcional

SGBD – Sistema de Gerenciamento de Banco de Dados

UML – *Unifield Modeling Language*

WU-BLAST – *Washington University Basic Local Alignment and Search Tool*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS DO TRABALHO 14
1.2 ESTRUTURA DO TRABALHO 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 BIOINFORMÁTICA 16
2.2 ALINHAMENTO DE SEQUÊNCIAS BIOLÓGICAS 17
2.3 ALGORITMO DE SMITH-WATERMAN 21
2.3.1 Inicialização da matriz de alinhamento 21
2.3.2 Preenchimento da matriz de alinhamento 22
2.3.3 Traceback 23
2.4 A FERRAMENTA JALIGNER 25
2.5 BANCO DE DADOS BIOLÓGICOS 27
2.6 SISTEMAS DISTRIBUÍDOS 28
2.7 JPVM 28
2.8 JOMP 30
2.9 TRABALHOS CORRELATOS 32
3 DESENVOLVIMENTO 34
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 34
3.2 ESPECIFICAÇÃO 34
3.2.1 Diagrama de casos de uso 35
3.2.1.1 Criação da fila de seqüências 35
3.2.1.2 Alinhamento de seqüências em ambiente distribuído 37
3.2.2 Diagrama de classes 39
3.2.3 Diagrama de atividades 41
3.3 IMPLEMENTAÇÃO 43
3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS 43
3.3.2 ARQUIVOS E CONFIGURAÇÕES DE AMBIENTE 43
3.3.3 CONFIGURAÇÃO DO CLUSTER 44
3.3.4 MONTAGEM DA FILA DE SEQÜÊNCIAS E CONFIGURAÇÃO DA FERRAMENTA JALIGNER 45
3.3.5 REALIZANDO PROCESSAMENTO EM AMBIENTE DISTRIBUÍDO 47

3.3.6 VISUALIZAÇÃO DOS RESULTADOS DOS ALINHAMENTOS 51
3.4 RESULTADOS E DISCUSSÃO 54
4 CONCLUSÕES **58**
4.1 EXTENSÕES 59
REFERÊNCIAS BIBLIOGRÁFICAS 60

1 INTRODUÇÃO

No dia 23 de fevereiro de 2003, o consórcio internacional que constitui o Projeto Genoma Humano anunciou oficialmente a conclusão do seqüenciamento dos três bilhões de bases de DNA[1] da espécie humana. O Projeto Genoma Humano é uma das maiores façanhas da história da humanidade e levou cerca de 15 anos para ser concluído. Ele é traduzido como um esforço da pesquisa internacional para seqüenciar e mapear todos os genes dos seres humanos, que no seu conjunto são conhecidos como genoma. Integrado a este projeto, esforços semelhantes vêm sendo empregados para a caracterização de genomas de vários outros organismos, uma vez que a maioria dos organismos vivos apresenta muitos genes que são similares, ou seja, com funções semelhantes.

A identificação das seqüências e das funções dos genes de outros organismos se traduz no potencial para explicar a homologia dos genes nos seres humanos, portanto, podendo ser usados como modelo. Segundo Gibas e Jambeck (2001, p. 4), um exemplo comum pode ser visto com as moscas-das-frutas, que é muito estudada como modelo na pesquisa sobre evolução de animais. Por isso, seus genes são bastante conhecidos. Ela tem um gene denominado *eyeless* que, se for retirado do genoma, resulta em moscas-das-frutas sem olhos. É evidente portanto, que o gene *eyeless* tem função importante no desenvolvimento do olho. O ser humano apresenta um gene denominado aniridia que também parece ter um papel fundamental no desenvolvimento de olhos. Essa dedução começou a se formar a partir da observação de que o ser humano que não tem esse gene ou em que o gene sofreu uma mutação suficiente para que o produto protéico parasse de funcionar corretamente, os olhos se desenvolvem sem a íris.

Gibas e Jambeck (2001, p. 5) citam que a mentalidade dedutiva dos cientistas os levou a fazer o questionamento do que poderia ocorrer ao inserir o gene aniridia em uma mosca-da-fruta sem olho. O que acontece é que a aniridia promove a produção de olhos normais na mosca. Poderia haver alguma similaridade em como o *eyeless* e a aniridia funcionam, apesar de moscas e seres humanos serem organismos extremamente diferentes. Para saber como o *eyeless* e a aniridia funcionam, juntos, é possível comparar as seqüências de DNA de cada gene.

---

[1] DNA é a abreviatura em Inglês para *Deoxyribonucleic Acid* (em português, ADN: ácido desoxirribonucléico). A sigla DNA ou ADN refere-se a uma molécula orgânica que reproduz o código genético. Esta molécula é responsável pela transmissão das características hereditárias de cada espécie de todos os seres vivos.

Uma vez representadas as seqüências de DNA pode-se então compará-las em busca de similaridades em suas estruturas. Essa comparação permite inferir sobre as propriedades de uma determinada molécula baseando-se em propriedades conhecidas da outra. O processo de comparação de seqüências denomina-se alinhamento de seqüências. Alinhar seqüência é colocar uma seqüência sobre a outra de forma que a correspondente entre elas fique clara.

O foco atual baseia-se no seqüenciamento de outros organismos, principalmente pela iniciativa privada, que investe nesses projetos pelo lucro que as pesquisas podem trazer, especialmente para as indústrias farmacêuticas. Em geral dirigem a pesquisa para os genes específicos, buscando através da comparação do DNA de diferentes indivíduos os genes "defeituosos", que causam doenças.

Entretanto, para a realização da comparação de seqüências biológicas desses organismos é preciso computadores de alto desempenho. Uma solução para tal problema é distribuir o processo de alinhamento de seqüências biológicas entre vários computadores de menor custo. Neste trabalho, pretende-se apresentar um sistema capaz de realizar o alinhamento de seqüências biológicas em ambiente distribuído, utilizando-se o poder de processamento de várias máquinas para realizar tal tarefa.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é aperfeiçoar e desenvolver novas rotinas em uma ferramenta de alinhamento de seqüências biológicas *open source*, no caso a ferramenta JAligner, para que o sistema resultante seja capaz de realizar o processamento em ambiente distribuído.

Os objetivos específicos do trabalho são:

a) disponibilizar rotinas para realizar o alinhamento entre seqüências biológicas em ambiente distribuído;
b) armazenar o resultado do alinhamento entre seqüências em um banco de dados (arquivo simples);
c) demonstrar o grau de similaridade entre as seqüências na ferramenta;
d) comparar o tempo de processamento consumido ao realizar o alinhamento de seqüências em ambiente distribuído com o tempo consumido na mesma tarefa em ambiente não distribuído.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está divido em quatro capítulos. O capítulo um apresenta uma introdução ao tema abordado e os objetivos. O capítulo dois apresenta a fundamentação teórica dos conhecimentos necessários para a realização deste trabalho, tal como trabalhos correlatos. O capítulo três aborda a especificação do protótipo, aspectos técnicos do desenvolvimento e implementação do mesmo. Por fim, no capítulo quatro são descritas as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados assuntos relacionados com o foco do trabalho proposto, enfatizando aspectos utilizados para sua definição e implementação. Os assuntos a serem abordados são bioinformática, alinhamento de seqüências biológicas, o algoritmo de Smith-Waterman, a ferramenta JAligner, banco de dados biológicos, sistemas distribuídos e as bibliotecas para computação paralela JPVM e JOMP. Também são abordados trabalhos correlatos ao assunto tratado.

2.1 BIOINFORMÁTICA

A bioinformática é uma das áreas científicas de maior crescimento na última década. Foca-se na criação e no uso de ferramentas para a organização e a análise de dados biológicos através dos computadores. É uma afluência da biologia, da informática e da matemática e com o avanço da tecnologia, tornou-se essencial ao desenvolvimento científico, principalmente em áreas como a biologia molecular, biotecnologia, bioquímica, farmácia, medicina e a biologia evolutiva (GARCÊS, 2007).

> A bioinformática é a ciência de usar as informações para entender a biologia. [...] Estritamente falando, a bioinformática é um subconjunto de um campo maior da biologia computacional, aplicação de técnicas analíticas quantitativas à modelagem de sistemas biológicos. [...] O campo da bioinformática conta muito com o trabalho de especialistas em métodos estatísticos e reconhecimento de padrões. Os pesquisadores na área de bioinformática são originalmente de muitos campos, incluindo matemática, ciência da computação e lingüística. (GIBAS; JAMBECK, 2001, p. 3).

Segundo Garcês (2007), o uso e o desenvolvimento de ferramentas para bioinformática tem aumentado, principalmente nas últimas duas décadas, a partir da fusão da informática e de ciências biológicas, para ajudar os cientistas a organizar e analisar a enorme quantidade de dados gerados. Gibas e Jambeck (2001, p. 4) citam que a bioinformática possibilita a realização de tarefas instigantes e complexas, como comparar seqüências biológicas e gerar resultados potencialmente significativos.

A bioinformática desenvolveu-se para enfrentar os resultados das iniciativas de seqüenciamento de genes, que produzem uma quantidade cada vez maior de dados sobre

proteínas, DNA e RNA. Desse modo, os biólogos moleculares passaram a utilizar métodos estatísticos capazes de analisar grandes quantidades de dados biológicos, a predizer funções dos genes e a demonstrar relações entre genes e proteínas (VOGT, 2003).

2.2 ALINHAMENTO DE SEQUÊNCIAS BIOLÓGICAS

Antes de adentrar ao assunto de alinhamento de seqüências biológicas, é conveniente explicar o que é uma seqüência biológica.

Uma seqüência é uma sucessão ordenada de caracteres de um alfabeto. Em bioinformática interessam seqüências compostas de caracteres que representam bases de DNA, RNA ou proteínas. As bases de DNA são formadas pelos nucleotídeos[2] Adenina, Citosina, Guanina e Timina. Sendo que as letras que representam esses nucleotídeos são, respectivamente, A, C, G, T. As bases que formam o RNA são praticamente as mesmas que formam o DNA, a única diferença é que a base Uricil substitui a Timina. Desta forma, as letras que representam o RNA são A, C, G, U. No quadro 1 é possível ver um exemplo de seqüência biológica (MORONY, 2006, p. 19).

```
ATGCAGCGCGTGAACATGATCATGGCAGAATCACCAGGCCTCATCACCATCTGCCTTTTAGG
ATATCTACTCAGTGCTGAATGTACAGTTTTTCTTGATCATGAAAACGCCAACAAAATTCTGA
ATCGGCCAAAGAGGTATAATTCAGGTAAATTGGAAGAGTTTGTTCAAGGGAACCTTGAGAGA
GAATGTATGGAAGAAAAGTGTAGTTTTGAAGAAGCACGAGAAGTTTTTGAAAACACTGAAAG
AACAACTGAATTTTGGAAGCAGTATGTTGATGGAGATCAGTGTGAGTCCAATCCATGTTTAA
ATGGCGGCAGTTGCAAGGATG
```

Fonte: Viveiros (2006, p. 51).

Quadro 1 – Fragmento do gene do fator IX de coagulação humano

As proteínas são formadas por aminoácidos. Como no caso das seqüências de DNA, grande parte das propriedades de uma proteína podem teoricamente ser deduzidas a partir da seqüência de aminoácidos que a compõem. Vinte aminoácidos ocorrem naturalmente em proteínas, eles podem ser vistos no quadro 2 juntamente com sua designação simbólica por letras e abreviação em inglês. (MEIDANIS; SETÚBAL, 1994, p. 7).

[2] Nucleotídeos são compostos ricos em energia e que auxiliam os processos metabólicos nas células.

AMINOÁCIDOS NATURAIS			
	Letra	**Abrev.**	**Nome**
1	A	Ala	Alanina
2	C	Cys	Cisteína
3	D	Asp	Ácido aspártico
4	E	Glu	Ácido glutâmico
5	F	Phe	Fenilalanina
6	G	Gly	Glicina
7	H	His	Histidina
8	I	Ile	Isoleucina
9	K	Lys	Lisina
10	L	Leu	Leucina
11	M	Met	Metionina
12	N	Asn	Asparagina
13	P	Pro	Prolina
14	Q	Gln	Glutamina
15	R	Arg	Arginina
16	S	Ser	Serina
17	T	Thr	Treonina
18	V	Val	Valina
19	W	Trp	Triptofano
20	Y	Tyr	Tirosina

Quadro 2 – Os vinte aminoácidos naturais

O processo de comparação de seqüências biológicas denomina-se alinhamento de seqüências. Segundo Meidanis e Setúbal (1994, p. 16), a comparação de seqüências é sem dúvida a operação primitiva mais importante na área de biologia computacional, servindo de base para muitas outras manipulações mais elaboradas.

Grosso modo, esta operação consiste em encontrar trechos semelhantes nas seqüências de entrada. Gibas e Jambeck (2001, p. 179) citam que antes de comparar seqüências biológicas é necessário realizar um alinhamento de seqüências. O conceito básico para uma seleção de uma boa seqüência de alinhamento é simples. As duas seqüências são combinadas aleatoriamente. A qualidade da combinação é avaliada e pontuada. Em seguida, uma seqüência é movida sobre a outra e a combinação é pontuada novamente, até que seja obtida a melhor pontuação de alinhamento. No quadro 3 é possível verificar as possibilidades de alinhamento da última base de DNA das seqüências `AAAC` e `AGC`.

```
Seqüências AAAC e AGC

      alinhamento ótimo entre (AAA )C
                              ( AG )C

      alinhamento ótimo entre (AAAC)-
                              ( AG )C

      alinhamento ótimo entre (AAA )C
                              (AGC )-
```

Fonte: Meidanis e Setúbal (1994, p. 19).

Quadro 3 – Exemplo de possibilidades de alinhamento entre a última base de DNA das seqüências AAAC e AGC

Contudo, por trás desta aparente simplicidade, esconde-se uma vasta gama de problemas distintos, com formalizações diversas, muito deles exigindo algoritmos e estruturas de dados próprias para a sua execução eficiente. O processo de alinhamento de seqüências exige o uso de computadores de alto desempenho.

> O uso de computadores é plenamente justificável em aplicações que manipulam grande volume de dados, como é o caso das buscas em bases de bioseqüências. Porém, mesmo em casos que poderiam ser resolvidos à mão é mais conveniente e seguro usar um computador [...]. (MEIDANIS; SETÚBAL, 1994, p. 17).

Segundo Gibas e Jambeck (2001, p. 166), a comparação de seqüências por pares é o modo principal de unir a função biológica ao genoma[3] e de transmitir informações conhecidas de um genoma ao outro. Normalmente, o fato de uma elevada semelhança entre seqüências implica também numa elevada semelhança entre funcionalidades ou estruturas dos genes.

A grande utilidade da computação no processo de alinhamento entre duas seqüências reside no fato que a quantidade de símbolos em uma seqüência pode ser muito grande, e pelo fato de que a busca por um alinhamento ótimo pode levar a uma explosão combinatória, tornando o trabalho extremamente longo. Além disso, mesmo para seqüências curtas é possível errar fazendo a mão, enquanto que um computador corretamente programado não erra (VIVEIROS, 2006, p. 36).

Morony (2006, p. 21) descreve que os principais métodos de alinhamento de seqüências genéticas dividem-se em duas categorias, alinhamento de pares de seqüências e alinhamento de múltiplas seqüências.

A primeira categoria, alinhamento de pares de seqüências é composta pelos seguintes métodos:

a) matriz de pontos, utilizada para comparar duas seqüências genéticas buscando

[3] Genoma corresponde ao conjunto de genes que contém as informações de uma dada espécie. É toda a informação hereditária de um organismo que está codificada em seu DNA.

possíveis alinhamentos de caracteres entre as seqüências, através da comparação de seqüências genômicas pareadas e seqüências repetitivas e inversões;

b) programação dinâmica, que é uma técnica de projeto de algoritmos, cujo método computacional calcula o melhor alinhamento possível entre as seqüências genéticas. Resolve-se o problema utilizando a solução de problemas menores, que tem como objetivo guardar todas as subsoluções numa tabela (matriz) e determinar quais subsoluções são necessárias para a solução global, tendo que existir essa tabela para que não seja preciso recalcular as subsoluções;

c) dicionário de palavras ou k-tuplas, realiza a comparação local através do uso de heurísticas. Suas implementações mais conhecidas são os softwares BLAST e o WU-BLAST.

A segunda categoria, chamada de alinhamento de múltiplas seqüências possui o grande problema de alinhar simultaneamente mais de duas seqüências e é também importante porque existe uma necessidade muito ampla de definir famílias de seqüências. Essa definição permite caracterizar e procurar novos membros da família e entender a estrutura, a evolução e a função dessas seqüências.

Conforme descrito por Prosdocimi et al (2003, p. 21), o alinhamento de seqüências ainda pode ser grosseiramente classificado em dois tipos, o alinhamento global e o alinhamento local. Segundo Viveiros (2006, p. 36), no alinhamento global, as seqüências envolvidas devem ser alinhadas de um extremo ao outro, dando origem a apenas um resultado. No alinhamento local é realizado em fragmento de uma seqüência, não precisa se estender do começo ao fim de duas seqüências a serem alinhadas. Para cada alinhamento é associada uma pontuação. Se a pontuação acumulativa em algum ponto da seqüência for negativa, o alinhamento pode ser interrompido e um novo alinhamento é iniciado. Ele também pode ser finalizado em qualquer local da seqüência.

O primeiro algoritmo projetado para detectar alinhamento global ótimo foi o de Needleman-Wunsch. Depois, uma pequena variante foi proposta, chamada de algoritmo de Smith-Waterman, que encontra o alinhamento local ótimo entre duas seqüências. Estes dois algoritmos requerem tempo de execução proporcional ao produto dos comprimentos das duas seqüências que estão sendo comparadas (LEMOS, 2004, p. 28).

Como as similaridades entre as seqüências de proteínas e DNA freqüentemente se localizam em apenas segmentos das seqüências envolvidas, os programas de busca de similaridades mais populares são os baseados no algoritmo de Smith-Waterman. No entanto, devido à complexidade desses algoritmos, eles tornam-se muito lentos para a maior parte dos

usuários, que não dispõe dos recursos necessários para utilizá-los (LEMOS, 2004, p. 28).

2.3 ALGORITMO DE SMITH-WATERMAN

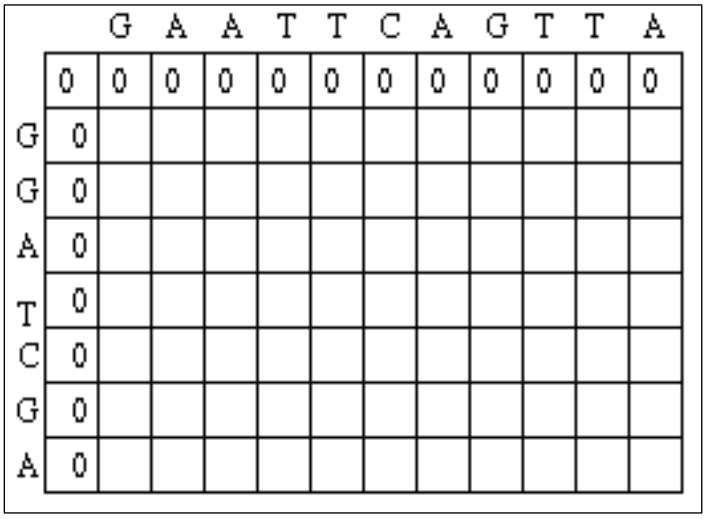
O algoritmo de Smith-Waterman é um algoritmo bem construído para executar o alinhamento local de seqüências, isto é, para determinar regiões similares entre seqüências de DNA ou seqüências de proteínas. Em vez de olhar a seqüência total, o algoritmo compara segmentos de todos os tamanhos possíveis. O algoritmo foi proposto por Smith Temple e por Michael Waterman. Este algoritmo exige recursos de tempo e de memória razoavelmente consideráveis, sendo que a fim de alinhar duas seqüências de comprimentos *m* e *n*, o tempo e o espaço requerido é de $O(mn)$ (SMITH-WATERMAN, 2006, tradução nossa).

Conforme descreve Morony (2006, p. 46), algoritmo de Smith-Waterman é baseado na técnica de programação dinâmica e é composto em três fases. A fase de inicialização da matriz de alinhamento está descrita na seção 2.3.1. A fase de preenchimento da matriz de alinhamento está descrita na seção 2.3.2. A última fase, denominada *traceback*, está descrita na seção 2.3.3.

2.3.1 Inicialização da matriz de alinhamento

O primeiro passo da programação dinâmica no alinhamento é criar uma matriz com número de colunas igual ao tamanho da primeira seqüência mais um e com o número de linhas igual ao comprimento da segunda seqüência mais um. A primeira linha e coluna da matriz são preenchidas com zero. A figura 1 apresenta esta fase no alinhamento das seqüências `GAATTCAGTTA` e `GGATCGA` (MORONY, 2006, p. 46).

Um simples sistema de pontuação é assumido. Em bioinformática, é conhecido por *match* quando um caractere da primeira seqüência for igual ao caractere da segunda seqüência. Quando um caractere da primeira seqüência for diferente do caractere da segunda seqüência é conhecido como *mismatch.* Para que as seqüências possuam o mesmo tamanho e para que haja correspondência entre os caracteres das seqüências, pode haver a necessidade de inserção de espaços. Esses espaços são conhecidos por *gaps* e entende-se por *gap penalty* uma proposta para penalizar espaços no alinhamento (MORONY, 2006, p. 46).



Fonte: Souto (2004).

Figura 1 - Inicialização da matriz do alinhamento

2.3.2 Preenchimento da matriz de alinhamento

Tendo o valor do *match*, *mismatch* e *gap* definidos, o valor de cada célula da matriz é calculado através da função MAX, conforme descrito no quadro 4. Esta função retorna a pontuação máxima para cada célula da matriz (MORONY, 2006, p. 47-48).

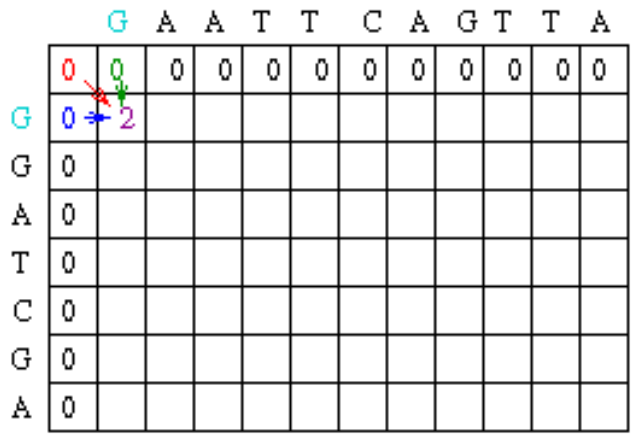
```
  Ai,j = MAX (
              Ai - 1, j - 1 + a(ai,bj),
              Ai, j - 1 + w,
              Ai - 1, j + w,
              0)
  Onde:
    Ai refere-se ao índice da primeira seqüência;
    j refere-se ao índice da segunda seqüência;
    a(ai,bj) refere-se a comparação entre os caracteres das seqüências,
resultando o valor definido para match/mismatch;
    w refere-se ao valor definido para um gap;
```

Fonte: Morony (2006, p. 54).

Quadro 4 – Função MAX utilizada no preenchimento da matriz do alinhamento

O preenchimento da matriz prossegue até que todas as células da matriz estejam preenchidas. Utilizando o valor de *match* igual a 2, *mismatch* igual –1 e *gap* igual –2, o valor da célula referente à linha 1 e coluna 1 da matriz será igual a 2, conforme exemplifica a figura 2.

```
M(1,1) = MAX[M(0,0) + 2, M(1,0) - 2, M(0,1) - 2]
       = MAX[2, -2, -2]
       = 2
```

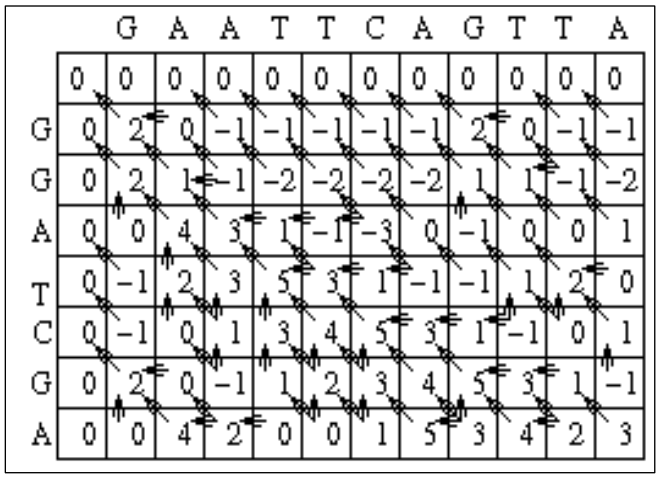


Fonte: Souto (2004).

Figura 2 - Preenchimento da primeira célula da matriz

A segunda fase do algoritmo é finalizada com o preenchimento de todas as células da matriz. É a fase do algoritmo de maior tempo de processamento.

A matriz totalmente preenchida pode ser visualizada na figura 3.



Fonte: Souto (2004).

Figura 3 – Matriz completamente preenchida

2.3.3 Traceback

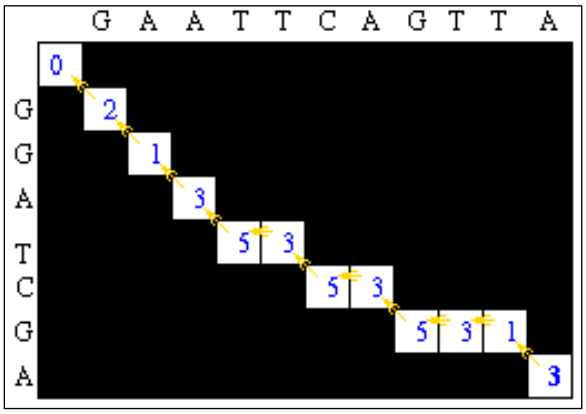
Após o preenchimento da matriz de alinhamento, dá-se início ao procedimento chamado de *traceback*. Este procedimento identifica os segmentos bem como produz o alinhamento correspondente que determinará qual é o resultado do sistema de pontuação

máxima da matriz de alinhamento, denominado de melhor alinhamento entre as duas seqüências ou alinhamento ótimo (MORONY, 2006, p. 49).

A pontuação desse alinhamento inicia-se na posição `A(m,n)`, sendo `A` referente a matriz de alinhamento e `m` e `n`, o tamanho das seqüências alinhadas. A pontuação máxima é obtida pela soma das pontuações das células da matriz, iniciando na célula `A(m,n)` até chegar à posição `A(0,0)`.

Segundo Morony (2006, p. 49), o *traceback* inicia na célula da posição `A(m,n)` e verifica as células vizinhas que poderiam ser suas predecessoras, ou seja, células que podem dar continuidade para obter o melhor alinhamento entre as duas seqüências da matriz `A`. Isso significa que a célula `A(m,n)` deve verificar o vizinho acima, o vizinho da diagonal e por último o seu vizinho da esquerda. Portanto, verifica-se no máximo três possibilidades segundo uma ordem preferencial, visto que cada seta indica um alinhamento, conforme na figura 3. Se houver mais de uma possibilidade, então existe mais de um alinhamento ótimo.

O *traceback* completo do alinhamento das seqüências `GAATTCAGTTA` e `GGATCGA` pode ser visualizado na figura 4. No quadro 5 é apresentado o alinhamento resultante e sua respectiva pontuação.



Fonte: Souto (2004).

Figura 4 – *Traceback* completo do alinhamento das seqüências `GAATTCAGTTA` e `GGATCGA`

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

Pontuação: 2-1+2-2+2+2-2+2-2-2+2 = 3
```

Fonte: Souto (2004).

Quadro 5 – Alinhamento resultante e pontuação do alinhamento entre as seqüências `GAATTCAGTTA` e `GGATCGA`

2.4 A FERRAMENTA JALIGNER

A ferramenta JAligner, desenvolvida por Ahmed Moustafa, é uma implementação *open source* do algoritmo de Smith-Waterman para alinhamento locais de pares de seqüências biológicas, utilizando o modelo de penalização de *gaps*. A ferramenta possui uma interface simples e amigável, o que facilita o uso. Outro item importante a destacar desta ferramenta é o fato de possui vários formatos padrões para visualização do resultado do alinhamento (MOUSTAFA, 2003, tradução nossa).

Conforme visto na figura 5, a ferramenta JAligner permite a seleção de uma matriz de pontuação para ser utilizada no alinhamento. Existem mais de 70 matrizes para escolha. As matrizes são utilizadas para saber discernir se o alinhamento é aleatório ou significativo. Se o alinhamento for significativo, estima-se a sua importância.

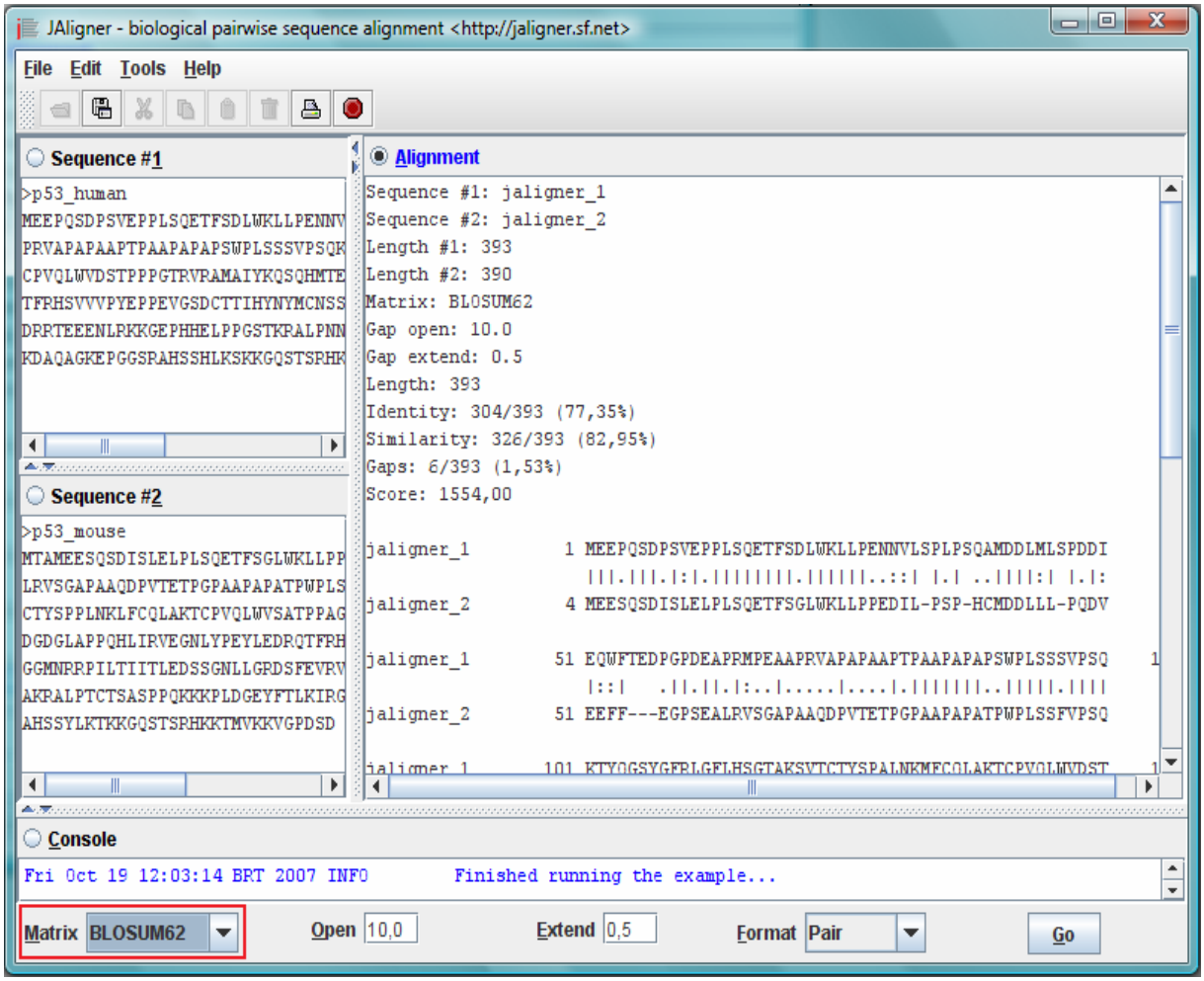


Figura 5 – Ferramenta JAligner

A matriz de pontuação é uma tabela de valores que descreve a probabilidade de ocorrer um par de resíduos em um alinhamento. Estas probabilidades são retiradas a partir de

amostras de alinhamentos de seqüências reais, comprovadamente válidas. O quadro 6 apresenta a matriz BLOSUM45, muito usada para substituição de aminoácidos (GIBAS; JAMBECK, 2001, p. 182).

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-2	-2	0	-1	-1	0	-5
R	-2	7	0	-1	-3	1	0	-2	0	-3	-2	3	-1	-2	-2	-1	-1	-2	-1	-2	-1	0	-1	-5
N	-1	0	6	2	-2	0	0	0	1	-2	-3	0	-2	-2	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-1	2	7	-3	0	2	-1	0	-4	-3	0	-3	-4	-1	0	-1	-4	-2	-3	5	1	-1	-5
C	-1	-3	-2	-3	12	-3	-3	-3	-3	-3	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-2	-3	-2	-5
Q	-1	1	0	0	-3	6	2	-2	1	-2	-2	1	0	-4	-1	0	-1	-2	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-2	0	-3	-2	1	-2	-3	0	0	-1	-3	-2	-3	1	4	-1	-5
G	0	-2	0	-1	-3	-2	-2	7	-2	-4	-3	-2	-2	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-5
H	-2	0	1	0	-3	1	0	-2	10	-3	-2	-1	0	-2	-2	-1	-2	-3	2	-3	0	0	-1	-5
I	-1	-3	-2	-4	-3	-2	-3	-4	-3	5	2	-3	2	0	-2	-2	-1	-2	0	3	-3	-3	-1	-5
L	-1	-2	-3	-3	-2	-2	-2	-3	-2	2	5	-3	2	1	-3	-3	-1	-2	0	1	-3	-2	-1	-5
K	-1	3	0	0	-3	1	1	-2	-1	-3	-3	5	-1	-3	-1	-1	-1	-2	-1	-2	0	1	-1	-5
M	-1	-1	-2	-3	-2	0	-2	-2	0	2	2	-1	6	0	-2	-2	-1	-2	0	1	-2	-1	-1	-5
F	-2	-2	-2	-4	-2	-4	-3	-3	-2	0	1	-3	0	8	-3	-2	-1	1	3	0	-3	-3	-1	-5
P	-1	-2	-2	-1	-4	-1	0	-2	-2	-2	-3	-1	-2	-3	9	-1	-1	-3	-3	-3	-2	-1	-1	-5
S	1	-1	1	0	-1	0	0	0	-1	-2	-3	-1	-2	-2	-1	4	2	-4	-2	-1	0	0	0	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-1	-1	2	5	-3	-1	0	0	-1	0	-5
W	-2	-2	-4	-4	-5	-2	-3	-2	-3	-2	-2	-2	-2	1	-3	-4	-3	15	3	-3	-4	-2	-2	-5
Y	-2	-1	-2	-2	-3	-1	-2	-3	2	0	0	-1	0	3	-3	-2	-1	3	8	-1	-2	-2	-1	-5
V	0	-2	-3	-3	-1	-3	-3	-3	-3	3	1	-2	1	0	-3	-1	0	-3	-1	5	-3	-3	-1	-5
B	-1	-1	4	5	-2	0	1	-1	0	-3	-3	0	-2	-3	-2	0	0	-4	-2	-3	4	2	-1	-5
Z	-1	0	0	1	-3	4	4	-2	0	-3	-2	1	-1	-3	-1	0	-1	-2	-2	-3	2	4	-1	-5
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-2	-1	-1	-1	-1	-1	-5
*	-5	–5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Fonte: Gibas e Jambeck (2001, p. 182).

Quadro 6 – Matriz de pontuação BLOSUM45

A ferramenta JAligner permite informar o valor da penalidade para *gaps* abertos (novo espaço) e *gaps* estendidos (espaços consecutivos). Esses valores influenciam na pontuação do alinhamento. Através do campo `Format` é possível definir o formato de visualização do alinhamento.

Além de apresentar o alinhamento gerado, são apresentadas duas outras informações muito importantes sobre o alinhamento, a identidade e a similaridade. A identidade de uma seqüência refere a presença do mesmo ácido nucléico ou aminoácido na mesma posição em duas seqüências alinhadas. A similaridade é dada pela melhor pontuação do alinhamento dentre todos os possíveis entre as seqüências a serem comparadas e posteriormente alinhadas. Reflete o qual semelhantes são as seqüências comparadas (GIBAS; JAMBECK, 2001, p. 181).

Vários pacotes de ferramentas da área de bioinformática incorporaram o JAligner, entre eles destacam-se BioWeka, RDPquery, STRAP e EMBOSS. Esses pacotes contém ferramentas especificas para determinadas tarefas da área de bioinformática (MOUSTAFA, 2003, tradução nossa).

2.5 BANCO DE DADOS BIOLÓGICOS

Em conseqüência da grande quantidade de informações de seqüências de nucleotídeos e de aminoácidos que são produzidas atualmente, principalmente nos projetos Genoma, Transcriptoma e Proteoma, o uso dos bancos de dados vem assumindo uma importância crescente na bioinformática (PROSDOCIMI et al, 2003, p. 19-20).

Segundo Viveiros (2006, p. 21), um banco de dados pode ser considerado uma coleção de dados inter-relacionados, projetado para suprir as necessidades de um grupo específico de aplicações e usuários. Um banco de dados organiza e estrutura as informações de modo a facilitar consultas, atualizações e exclusão de dados. A grande maioria dos bancos de dados é atrelado a um sistema denominado sistema de gerenciamento de banco de dados (SGBD). Este sistema é responsável por intermediar os processos de construção, manipulação e administração do banco de dados solicitados pelos usuários ou por outras aplicações.

Gibas e Jambeck (2001, p. 361) ressaltam que existem dois tipos de gerenciamento de banco de dados, o de indexação de arquivos simples e os bancos de dados relacionais. Um terceiro tipo, o banco de dados orientado a objetos, está começando a ficar mais popular.

Os bancos de dados de arquivos simples são o tipo de banco de dados que os não-especialistas entendem com mais facilidade. Um banco de dados de arquivos simples não é realmente um banco de dados, é simplesmente uma coleção ordenada de arquivos semelhantes, geralmente em conformidade com um formato padrão de conteúdo. A ênfase na formatação de dados para um banco de dados de arquivos simples está no nível dos caracteres, isto é, no nível de como os dados apareceriam se fossem impressos em uma página (GIBAS; JAMBECK, 2001, p. 361).

Segundo Gibas e Jambeck (2001, p. 363), em um banco de dados relacional, as informações são guardadas em um conjunto de tabelas. Todas as tabelas são etiquetadas com a identidade da proteína que descrevem, de forma que é possível fazer conexões entre elas. Assim como os bancos de dados de arquivos simples, os bancos de dados relacionais são apenas uma forma de coletar todas as informações sobre algo e armazená-las em um computador.

Bancos de dados biológicos são banco de dados, de arquivos simples ou relacionais, que armazenam informações de caráter biológico, como por exemplo, seqüências de DNA. Este tipo de banco vem aumentando de importância devido a grande quantidade de seqüências biológicas produzidas por diversos projetos da área de bioinformática.

2.6 SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos podem ser definidos como uma coleção de computadores independentes que se apresentam ao usuário como um sistema único e consistente. Interligados por uma rede, os computadores disponibilizam os mesmos recursos e fontes a diferentes usuários, causando a impressão de que trabalham a partir de um sistema único, quando na verdade, o ambiente é compartilhado (TANENBAUM, 2003, p. 413-414).

A este propósito é possível mencionar como exemplo de sistemas distribuídos os *clusters*. Um *cluster* é um sistema que compreende dois ou mais computadores ou sistemas que trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma que os usuários que os utilizam tenham a impressão que somente um único sistema responde para eles, criando assim uma ilusão de um recurso único (PITANGA, 2003).

Existem vários tipos de *clusters*, entretanto o tipo utilizado para alinhar as seqüências biológicas é o modelo de processamento distribuído ou, como também é conhecido, processamento paralelo. Isto porque, segundo Pitanga (2003), este modelo de *cluster* aumenta a disponibilidade e *performance* para as aplicações, particularmente para as grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida em pequenas tarefas que são distribuídas ao redor das estações, como se fosse um supercomputador massivamente paralelo.

2.7 JPVM

JPVM é uma biblioteca relativamente pequena e implementada inteiramente em Java que segue a estrutura básica do PVM. PVM é uma abreviatura para *parallel virtual machine*, ou máquina paralela virtual, na qual permite que um conjunto de máquinas heterogêneas seja visto como uma única máquina paralela virtual. O PVM lida transparentemente com o roteamento de mensagens, conversão de dados e o escalonamento de processos em uma rede formada por arquiteturas antes incompatíveis (FERREIRA, 1998).

Segundo Ferreira (1998), usuários do PVM desenvolvem suas aplicações como uma série de tarefas PVM que trabalham juntas na resolução de um problema. As tarefas têm acesso aos recursos do PVM através de uma interface padrão composta de rotinas

implementadas em uma biblioteca de funções que é compilada juntamente com o código do programa. Este método permite a comunicação entre vários computadores com memória própria, formando um único recurso computacional. Esses computadores se comunicam por troca de mensagens através de uma rede de computadores, respeitando as regras dos protocolos de comunicação.

Contudo, o JPVM apresenta um nível de portabilidade e interoperabilidade superior ao do PVM padrão e também oferece recursos favoráveis à implementação de programas concorrentes, devido a características próprias da linguagem Java. Isso ocorre pelo fato da linguagem Java oferecer características que estimulam o seu uso na construção de ambientes paralelos heterogêneos (FERRARI, 1998).

A biblioteca JPVM fornece uma interface semelhante à oferecida em C/C++ e Fortran pelo PVM, mas com a sintaxe e semântica utilizada pela linguagem Java. Essa similaridade facilita o uso para os usuários já familiarizados com o PVM. A instalação da biblioteca é simples e não necessita de privilégios especiais, ou seja, não é necessário acessar a máquina como superusuário para instalar o JPVM. Para a utilização do JPVM, não é necessária a prévia instalação do PVM, como acontece em algumas distribuições do PVM para outras linguagens (OLIVEIRA, 2005, p. 34).

O JPVM é formado por dois componentes, o jpvmDaemon e o jpvmEnvironment. O jpvmDaemon é um processo que é executado em segundo plano em todos os nós do *cluster*, formando a máquina virtual paralela. Esta é responsável por coordenar as tarefas em execução e realizar a comunicação entre todos os processos criados pela aplicação JPVM. O jpvmEnvironment é uma biblioteca que implementa um conjunto completo de primitivas que permitem codificar, enviar e receber mensagens, criar e eliminar processos, sincronizar tarefas e modificar a máquina virtual. Também é possível alterar a máquina virtual paralela através do *console*, que acompanha o pacote JPVM (FERRARI, 1998).

Segundo Oliveira (2005, p. 35), as aplicações, escritas em Java, podem ser paralelizadas utilizando as primitivas para troca de mensagens disponibilizadas pelo JPVM. O modelo computacional do PVM, utilizado pelo JPVM, assume que qualquer processo pode enviar uma mensagem para qualquer outro processo PVM. Com a implementação de troca de mensagens, os vários processos que formam a aplicação podem cooperar para resolver um problema em paralelo.

2.8 JOMP

JOMP é um projeto de pesquisa cujo objetivo é definir e implementar uma plataforma para programação de processamento paralelo com memória compartilhada em Java. Consiste de um conjunto de diretivas, bibliotecas usadas em tempo de execução e variáveis de ambiente (BULL; KAMBITES, 2000, tradução nossa).

JOMP utiliza o modelo de paralelização de tarefas em tempo de execução. Um programa que utiliza o JOMP começa sua execução numa única *thread* (chamada de *master thread*). A *master thread* executa de forma serializada até que a primeira região construída para ser executada de forma paralela é encontrada. A *master thread* então cria um time de *threads*, na qual ela mesma está incluída. Então cada *thread* executa o código da região paralela (BULL; OBDRZALEK, 2000, tradução nossa).

Bull e Kambites (2000, tradução nossa), destacam que existe a possibilidade de escrever programas com memória compartilhada utilizando modelo de *threads* nativas do Java. Entretanto, o sistema de diretivas do JOMP tem uma série de vantagens sobre abordagem de utilização de *threads* nativas. Primeiramente, o código resultante é muito mais parecido com o código da versão seqüencial do mesmo programa. Isto torna o desenvolvimento e manutenção do código significativamente mais fácil. Outro problema em usar *threads* nativas do Java é que para a máxima eficiência no uso de arquiteturas paralelas com memória compartilhada, é necessário o uso de uma *thread* por processador e é preciso que essas *threads* permaneçam ativas durante toda a execução do programa. A sincronização entre as *threads* é trabalhosa e requerem atenção na implementação.

Através do uso de diretivas, o JOMP facilita a implementação de programas de processamento paralelo. A sintaxe de uma diretiva do JOMP é especificada informalmente, conforme quadro 7.

```
//omp directive-name [clause[ clause] ...]
[//omp [ clause[clause] ...]
...
```

Fonte: Bull e Obdrzalek (2000).

Quadro 7 – Sintaxe de uma diretiva JOMP

Segundo Bull e Obdrzalek (2000, tradução nossa), as principais diretivas disponíveis para uso no JOMP e que foram utilizadas neste trabalho estão listadas a seguir:

a) `//omp parallel` é a diretiva que define uma região paralela, ou seja, a região que será executada por múltiplas *threads* em paralelo. Esta diretiva é fundamental para

ativar a execução paralela. A sintaxe desta diretiva é apresentada no quadro 8.

```
//omp parallel [clause[ clause] ...]
structured-block
```

Fonte: Bull e Obdrzalek (2000).

Quadro 8 – Sintaxe da diretiva JOMP para definir uma região paralela

b) `//omp for` é a diretiva que especifica que as iterações do *loop* serão executadas em paralelo. As iterações do *loop* serão distribuídas através do time corrente de *threads*. A sintaxe desta diretiva é apresentada no quadro 9.

```
//omp for [clause[ clause] ... ]
for-loop
```

Fonte: Bull e Obdrzalek (2000).

Quadro 9 – Sintaxe da diretiva JOMP para definir um *loop* paralelo

c) `//omp sections` é a diretiva `que` especifica as seções de código que deverão ser executadas paralelamente pelo time de *threads*. A sintaxe desta diretiva é apresentada no quadro 10.

```
//omp sections [clause[ clause] ...]
  {
     //omp section
     structured-block
     [//omp section
     structured-block
     .
     .
     .]
  }
```

Fonte: Bull e Obdrzalek (2000).

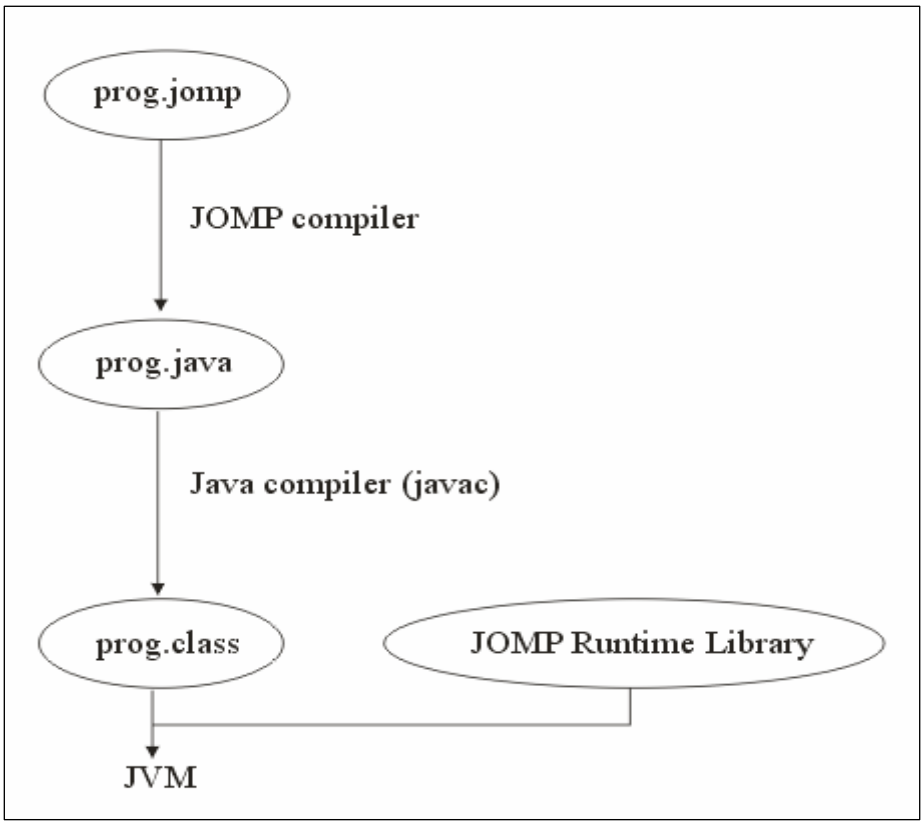
Quadro 10 – Sintaxe da diretiva JOMP para definir seções paralelas

O arquivo na qual são informadas as diretivas deve possuir a extensão `jomp`. Este arquivo contém tanto as diretivas do JOMP como o código Java. Após informar as diretivas, deve-se submeter o arquivo ao compilador do JOMP, conforme quadro 11.

```
java jomp.compiler.Jomp MyFile
```

Quadro 11 – Compilação do arquivo com as diretivas JOMP

Como resultado da compilação, caso não forem encontrados erros, será gerado um fonte Java, preparado para executar em paralelo as rotinas previamente definidas pelas diretivas. Este código gerado pode ser compilado com o compilador do Java (`javac`). O fluxograma do sistema JOMP pode ser visto na figura 6.



Fonte: Bull et al (2000).

Figura 6 – Fluxograma de uso do sistema JOMP

Para especificar a quantidade de *threads* que formarão o time de *threads* deve-se inicializar o programa com o parâmetro `-Djomp.threads`. Através deste parâmetro é possível indicar o número de *threads* que serão criadas durante a execução do programa (BULL; OBDRZALEK, 2000, tradução nossa).

O quadro 12 exemplifica o uso do parâmetro para indicar o número de *threads*.

```
java -Djomp.threads=4…
```

Quadro 12 – Parâmetro para informar quantidade de *threads*

2.9 TRABALHOS CORRELATOS

Dentre as ferramentas de bioinformática utilizadas para o alinhamento de seqüências pode-se destacar o Basic Local Alignment Search Tool (BLAST) e o FASTA Sequence Comparison (FASTA). Dentre os trabalhos acadêmicos, pode-se destacar o trabalho de conclusão de curso desenvolvido por Felipe Fernandes Albrecht, sobre reconstrução filogenética em ambiente distribuído.

Segundo Viveiros (2006, p. 40), o BLAST é uma ferramenta de alinhamento que, dada

uma seqüência de consulta, apresenta como resultado a extensão e qualidade da similaridade da seqüência. Em cada um dos alinhamentos encontrados, apresenta uma descrição da seqüência e a pontuação do alinhamento obtido. É uma ferramenta desenvolvida pelo National Center for Biotechnology Information (NCBI).

Segundo Morony (2006, p. 29), existe uma modificação da ferramenta BLAST original para execução em ambientes distribuídos. Trata-se do mpiBLAST. O mpiBLAST fraciona o banco de seqüências e no processo de pesquisa por seqüências similares, atribui as frações a cada um dos computadores do ambiente distribuído através do padrão de computação paralela *Message Passing Interface* (MPI).

A ferramenta FASTA também tem como objetivo encontrar regiões similares entre seqüências de proteínas ou DNA. Como o BLAST, o FASTA pode ser utilizado para inferir funcionalidades entre seqüências. Destaca-se o fato de que o BLAST, assim como o FASTA, não utilizam programação dinâmica e sim heurísticas. Esses softwares fazem o alinhamento local das seqüências genéticas, porém, não há garantias de que vão encontrar o melhor alinhamento (MORONY, 2006, p. 29)

Na FURB, Felipe Fernandes Albrecht desenvolveu um trabalho visando otimizações para um *workflow* de filogenias de proteínas homólogas distantes e um algoritmo paralelo para inferência de árvores filogenéticas utilizando o método de *Least Squares*. O objetivo da análise filogenética de seqüências biológicas é observar a relação entre vários grupos de seqüências e traçar uma linha evolutiva entre elas.

Antes de iniciar a construção da árvore filogenética torna-se necessário realizar uma pesquisa por seqüências similares no banco de dados e realizar um alinhamento entre as seqüências encontradas. Para esta tarefa foi utilizada a ferramenta mpiBLAST. Através do uso de um algoritmo paralelo utilizando o método de *Least Squares*, buscou-se a diminuição do tempo total consumido na construção de árvores filogenéticas (ALBRECHT, 2006, p. 7-90).

3 DESENVOLVIMENTO DO PROTÓTIPO

A seguir serão descritos os requisitos do sistema, a especificação e a implementações realizadas no protótipo. Na seção posterior é explanada a implementação executada para atingir os objetivos deste trabalho. Em seguida é apresentada a operacionalidade das implementações desenvolvidas e também uma discussão sobre procedimentos adotados no desenvolvimento.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A proposta do sistema a ser desenvolvido atenderá a algumas especificações e necessidades, tais como:

a) carregar seqüências biológicas, armazenando-as em uma fila para processamento (Requisito Funcional - RF);
b) realizar o alinhamento das seqüências biológicas com as demais seqüências armazenadas na fila, em ambiente distribuído (RF);
c) apresentar o grau de similaridade, identidade e pontuação, resultante do alinhamento entre as seqüências biológicas (RF);
d) armazenar o resultado dos alinhamentos em um banco de dados de arquivos simples (RF);
e) ser implementado na linguagem Java (Requisito Não Funcional - RNF);
f) utilizar a biblioteca JPVM (RNF);
g) utilizar a biblioteca JOMP (RNF).

3.2 ESPECIFICAÇÃO

A especificação das rotinas implementadas foi realizada com a utilização da Unified Modeling Language (UML), através de diagramas de casos de uso, de classes e de atividades. Para a construção dos diagramas utilizou-se a ferramenta Enterprise Architect.

3.2.1 Diagrama de casos de uso

Diagramas de casos de uso documentam os requisitos funcionais de um sistema, fornecendo assim uma visão do papel que o sistema deve ter, sempre com foco nas necessidades do usuário.

Nas seções 3.2.1.1 e 3.2.1.2, são retratados os casos de uso existentes entre o usuário e a ferramenta JAligner, em relação as rotinas implementadas para realizar o alinhamento de seqüências em ambiente distribuído.

3.2.1.1 Criação da fila de seqüências

A figura 7 representa os casos de usos relacionados com a manipulação da fila de seqüências biológicas.

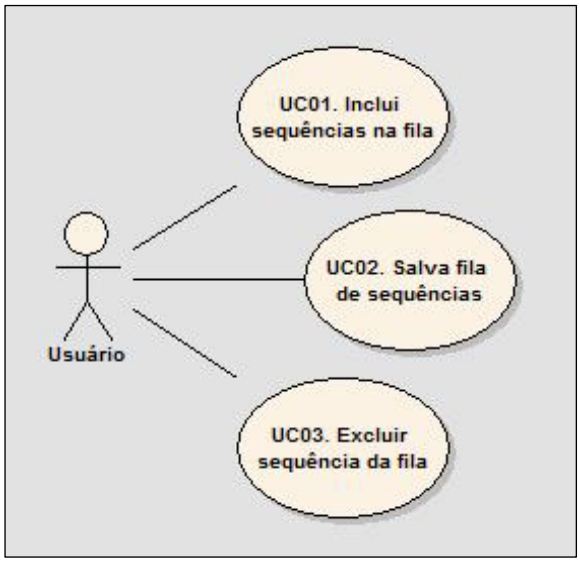


Figura 7 – Casos de usos para manipulação da fila de seqüências

O Quadro 13 apresenta o detalhamento do caso de uso `UC01`, apresentado na figura 7.

UC01: Inclui seqüências na fila	
Resumo	Usuário inclui seqüências biológicas numa fila.
Seqüência de ações	1. Usuário seleciona fila de seqüência na qual deseja incluir uma nova seqüência. 2. Usuário seleciona botão *Open* na ferramenta JAligner. 3. Usuário seleciona o arquivo que contém seqüências biológicas. 4. Arquivos selecionados são importados para fila de seqüências.
Exceções	No passo 4, caso a seqüência informada contenha caracteres que não façam parte de uma cadeia de DNA, RNA ou proteínas, a mesma não é incluída na fila de seqüência. Uma mensagem de alerta é gerado para o usuário no *console* da ferramenta JAligner.

Quadro 13 – Detalhamento do caso de uso UC01

O detalhamento do caso de uso `UC02` da figura 7 é apresentado no quadro 14.

UC02: Salva fila de seqüências	
Resumo	Usuário salva fila de seqüências biológicas para uso posterior em outro alinhamento.
Seqüência de ações	1. Usuário seleciona fila de seqüência que deseja salvar. 2. Usuário seleciona botão *Save* na ferramenta JAligner. 3. Usuário informa local onde deseja salvar o arquivo com a lista de seqüências biológicas. 4. Arquivo com a lista de seqüências biológicas é gerado.

Quadro 14 – Detalhamento do caso de uso UC02

O detalhamento do caso de uso `UC03` da figura 7 é apresentado no quadro 15.

UC03: Excluir seqüência da fila	
Resumo	Usuário exclui uma seqüência da fila.
Seqüência de ações	1. Usuário seleciona seqüência que deseja excluir da fila. 2. Usuário seleciona botão *Delete*. 3. Seqüência biológica selecionada é excluída da fila.

Quadro 15 – Detalhamento do caso de uso UC03

3.2.1.2 Alinhamento de seqüências em ambiente distribuído

A figura 8 representa os casos de usos relacionados com a execução do alinhamento de seqüências em ambiente distribuído.

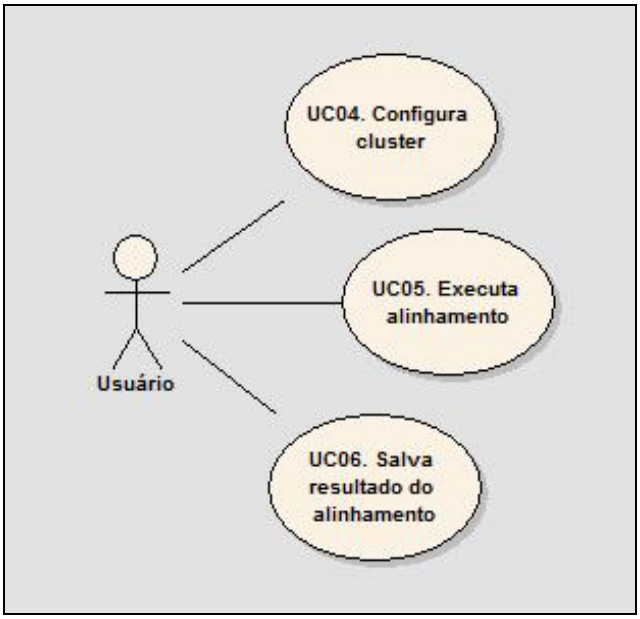


Figura 8 – Casos de usos relacionados com a execução do alinhamento em ambiente distribuído

O Quadro 16 apresenta um detalhamento do caso de uso `UC04`, apresentado na figura 8.

UC04: Configura cluster	
Resumo	Usuário configura cluster.
Seqüência de ações	1. Usuário informa quais computadores formam a máquina virtual paralela, ou seja, os nós do cluster. 2. Usuário informa a quantidade de *threads* que serão utilizadas no processamento de cada tarefa nos nós do cluster. 3. Usuário informa em qual diretório serão salvos os alinhamentos realizados.

Quadro 16 – Detalhamento do caso de uso UC04

O detalhamento do caso de uso `UC05` da figura 8 é apresentado no quadro 17.

UC05: Executa alinhamento	
Resumo	Usuário executa alinhamento.
Seqüência de ações	1. Usuário cria filas de seqüências biológicas. 2. Usuário configura cluster. 3. Usuário seleciona matriz de pontuação a ser utilizada no alinhamento. 4. Usuário informa valor da penalidade para os espaços abertos e consecutivos que podem vir a surgir no alinhamento. 5. Usuário informa o formato de saída do alinhamento. 6. Usuário pressiona botão *Go* na ferramenta JAligner. 7. Tarefas de alinhamento são distribuídas pelos nós do cluster. 8. Aguarda término do processamento. 9. Apresenta resultado geral dos alinhamentos.

Quadro 17 – Detalhamento do caso de uso UC05

O detalhamento do caso de uso `UC06` da figura 8 é apresentado no quadro 18.

UC06: Salva resultado do alinhamento	
Resumo	Usuário salva resultado geral do alinhamento das seqüências biológicas.
Seqüência de ações	1. Usuário seleciona quadro *Aligment*, na ferramenta JAligner. 2. Usuário seleciona botão *Save*. 3. Usuário informa local onde deseja salvar o arquivo com o resultado do alinhamento. 4. Arquivo com resultado geral do alinhamento é gerado.

Quadro 18 – Detalhamento do caso de uso UC06

A figura 9 representa o caso de uso relacionado com o processamento do alinhamento de seqüências biológicas em um nó do cluster.

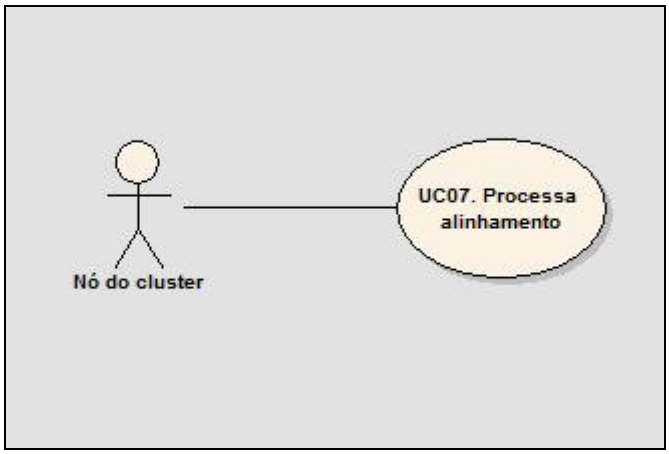


Figura 9 – Caso de uso relacionados com o processamento do alinhamento em um nó do cluster

O quadro 19 apresenta o detalhamento do caso de uso UC07, da figura 9.

UC07: Processa alinhamento	
Resumo	Nó do cluster realiza processamento do alinhamento de seqüências biológicas.
Seqüência de ações	1. Aguarda o recebimento do pacote contendo seqüências biológicas para alinhar. 2. Realiza o alinhamento através do algoritmo de Smith-Waterman, dividindo o processamento conforme o número de *threads* pré-definidos na ferramenta JAligner. 3. Monta pacote contendo resultado do alinhamento das seqüências. 4. Envia resultado para nó do cluster que disparou o processamento do alinhamento.

Quadro 19 – Detalhamento do caso de uso UC07

3.2.2 Diagrama de classes

Pilone e Pitman (2006, p. 5) destacam que os diagramas de classes utilizam classes e interfaces para documentar detalhes sobre as entidades que formam o sistema e as relações estáticas entre elas.

A figura 10 apresenta as principais classes do protótipo.

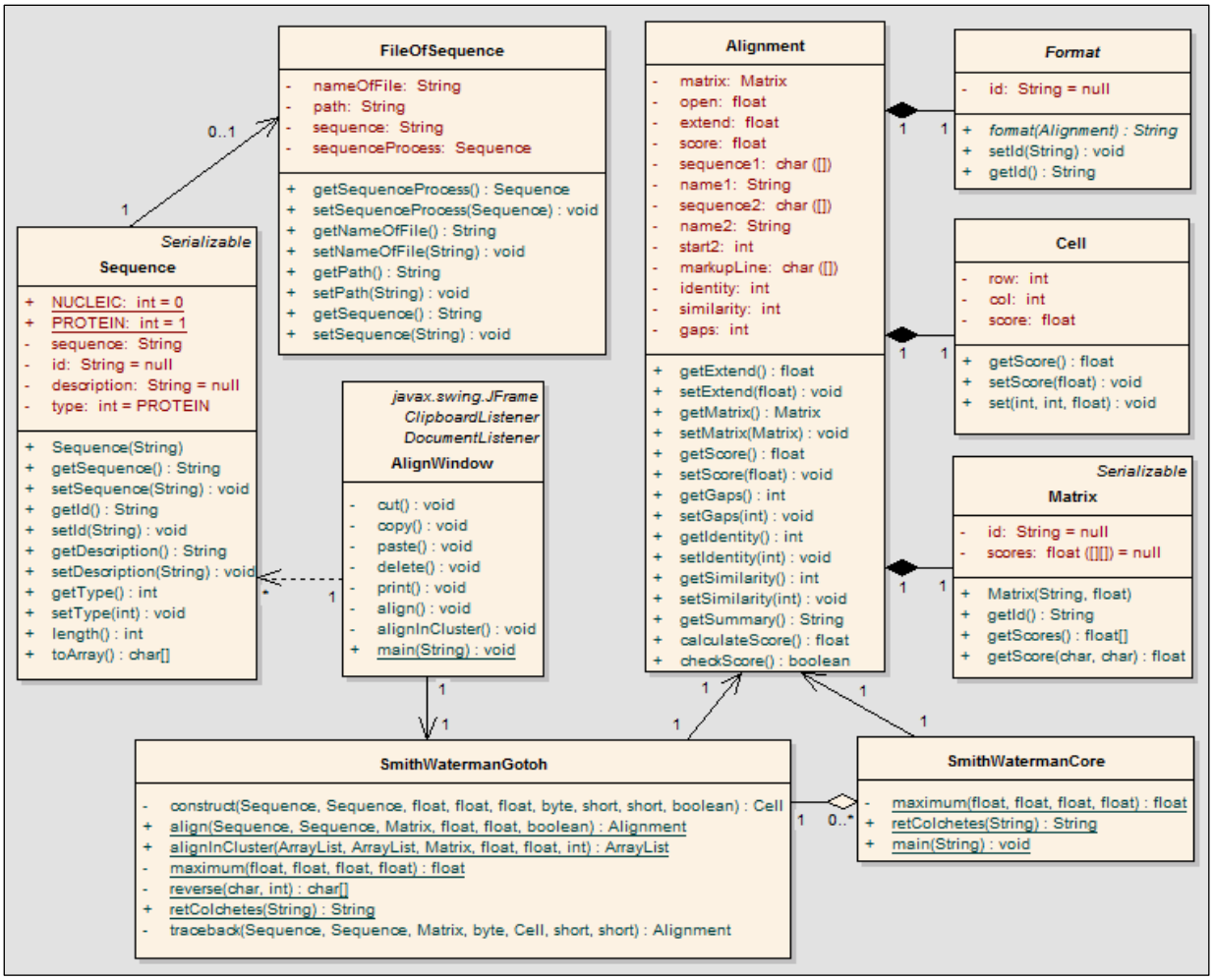


Figura 10 – Diagrama de classes

O quadro 20 apresenta uma descrição das funcionalidades de cada uma destas classes.

Classe	Descrição
AlignWindow	Responsável pelas interações gráficas do protótipo com o usuário.
Sequence	Classe que guarda as informações da seqüência.
FileOfSequence	Utilizada para manipulação do arquivo que contém a seqüência biológica.
SmithWatermanGotoh	Realiza o alinhamento das seqüências através do algoritmo de Smith-Waterman. Quando o alinhamento é feito em ambiente distribuído, é responsável por dividir a tarefa em partes e envia-las para processamento nos nós escravos do *cluster*.
SmitWatermanCore	Utilizado em alinhamento de seqüências em ambiente distribuído. Processa alinhamento ao qual foi designado e envia resultado para o nó mestre do *cluster*.
Aligment	Classe que contém atributos do alinhamento e métodos para cálculo da pontuação.
Format	Define o formato do resultado do alinhamento.
Cell	Guarda informações relacionadas ao alinhamento ótimo.
Matrix	Classe para manipulação das matrizes de pontuação.

Quadro 20 – Funcionalidade das classes do protótipo

3.2.3 Diagrama de atividades

Os diagramas de atividades são utilizados para documentar o fluxo de uma atividade ou comportamento. Sendo assim, conceitualmente muito similares a um fluxograma (PILONE; PITMAN, 2006, p. 6).

A figura 11 demonstra, através de um diagrama de atividades, os passos realizados pelo usuário e pelos nós do cluster para processar o alinhamento de seqüências biológicas. O cluster apresentado na figura 11 é formado por dois computadores, sendo denominado `nó mestre do cluster` o computador na qual foi iniciado o alinhamento das seqüências na ferramenta JAligner e por `nó escravo do cluster` o computador que realiza o processamento do alinhamento.

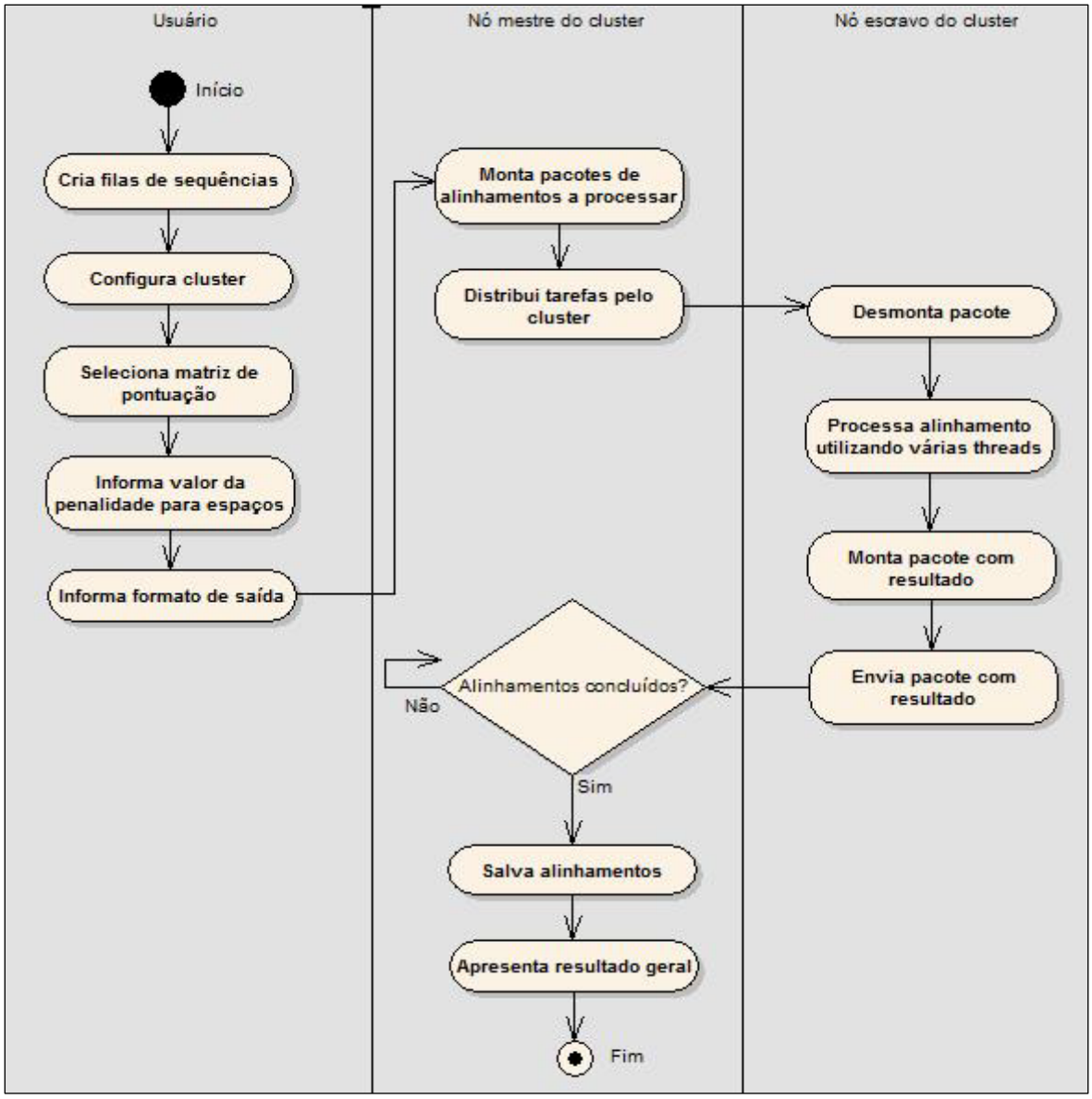


Figura 11 – Diagrama de atividades do alinhamento em ambiente distribuído

Sendo assim, observa-se que antes de iniciar o processamento dos alinhamentos, o usuário deverá selecionar a matriz de pontuação, informar o valor da penalidade para espaços abertos e consecutivos no alinhamento, tal como seu formato de saída.

O `nó mestre do cluster` realiza a tarefa de montar os pacotes contendo as seqüências a alinhar e demais parâmetros necessários. Ele distribui os pacotes entre os computadores que formam a máquina virtual. Os computadores que realizam de fato o alinhamento, também conhecidos como `nós escravos do cluster`, desmontam o pacote contendo as seqüências e iniciam o processamento. Após o término do processamento, um pacote contendo o resultado é montado e enviado ao `nó mestre do cluster`.

Após o término do processamento de todos os alinhamentos, o `nó mestre do cluster` salva o resultado de cada alinhamento em um diretório especificado pelo usuário. Em seguida é apresentado na ferramenta JAligner o resultado geral do alinhamento.

3.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

A linguagem de programação utilizada no desenvolvimento do protótipo é Java, sendo que o ambiente de programação utilizado foi o Eclipse. Porém, em relação as implementações de interface do protótipo, as mesmas foram realizadas através da ferramenta Netbeans.

Para possibilitar que o processamento seja realizado em ambiente distribuído foi utilizada a biblioteca JPVM. Para otimizar o processamento dos alinhamentos de seqüências biológicas, de forma que o processamento ocorresse de forma paralela, foi utilizada a biblioteca para programação de processamento paralelo com memória compartilhada chamada JOMP.

3.3.2 Arquivos e configurações de ambiente

Para que o alinhamento de seqüências biológicas ocorra de forma distribuída é necessário que dois ou mais computadores estejam interligados entre si, ou seja, formem uma rede de computadores. Esses computadores precisam estar previamente configurados para que os mesmos formem um cluster.

Primeiramente, deve-se instalar o protótipo em todos os computadores que farão parte do cluster. Para instalar basta descompactar o arquivo de instalação ou copiar os arquivos do protótipo para o diretório de preferência.

Em seguida, deve-se configurar as variáveis de ambiente do sistema operacional, informando o diretório onde se encontra a máquina virtual do Java e onde está instalado o protótipo. Para isto devem ser configuradas as variáveis de ambiente `java` e `classpath`.

Após, é necessário executar o jpvmDaemon. Somente os computadores em que o jpvmDaemon estiver executando é que poderão formar a máquina virtual paralela. O jpvmDaemon está disponível no diretório em que o protótipo foi instalado, sendo que o

mesmo pode ser executado através do comando apresentado no quadro 21.

```
java jaligner.jpvmDaemon
```

Quadro 21 – Comando para executar o Daemon JPVM

Para que o jpvmDaemon execute logo que o computador for ligado, é possível criar um arquivo de lote (.bat) e associa-lo a inicialização do sistema operacional. Neste arquivo pode-se incluir o comando para executar o jpvmDaemon.

3.3.3 Configuração do cluster

Conforme apresentado na seção 3.3.2, para formar a máquina virtual paralela é preciso que todos os computadores que fazem parte da mesma estejam executando o jpvmDaemon. Entretanto, o cluster não é formado apenas por estar executando o jpvmDaemon em diversos computadores. É preciso dizer quais computadores irão formar os nós do cluster. Isto é feito através do jpvmConsole.

O jpvmConsole é maneira de configurar o cluster. Ele está disponível no mesmo diretório em que a ferramenta JAligner está instalada, podendo ser executado através do comando descrito no quadro 22.

```
java jaligner.jpvmConsole
```

Quadro 22 – Comando para executar o *console* do JPVM

O comando executado no *console* para adicionar um novo computador na máquina paralela virtual é `add`, onde são informados o nome do computador e a porta de conexão, conforme apresentado no quadro 23. O processamento dos alinhamentos ocorrerá nos nós do cluster, através de tarefas JPVM.

```
jpvm> add
      Host name   : edson
      Port number : 1234
```

Quadro 23 – Adicionando um computador na máquina paralela virtual

3.3.4 Montagem da fila de seqüências e configuração da ferramenta JAligner

A ferramenta JAligner realiza o alinhamento de seqüências através do algoritmo de

Smith-Waterman. Este algoritmo realiza o alinhamento ótimo entre duas seqüências biológicas, exigindo recursos de tempo e de memória razoavelmente consideráveis. Isto gera um gargalo e não facilita o seu uso na busca por similaridade entre genes.

Desta forma, foi implementado para que seja possível criar duas filas de seqüências biológicas. As seqüências da primeira fila serão alinhadas com todas as seqüências da segunda fila. Estas filas possibilitam o processamento de diversas seqüências de forma paralela, o que torna a ferramenta JAligner mais funcional.

A figura 12 apresenta em destaque as filas de seqüência, sendo que a primeira fila contém apenas uma seqüência biológica informada e na segunda fila existem três seqüências informadas.

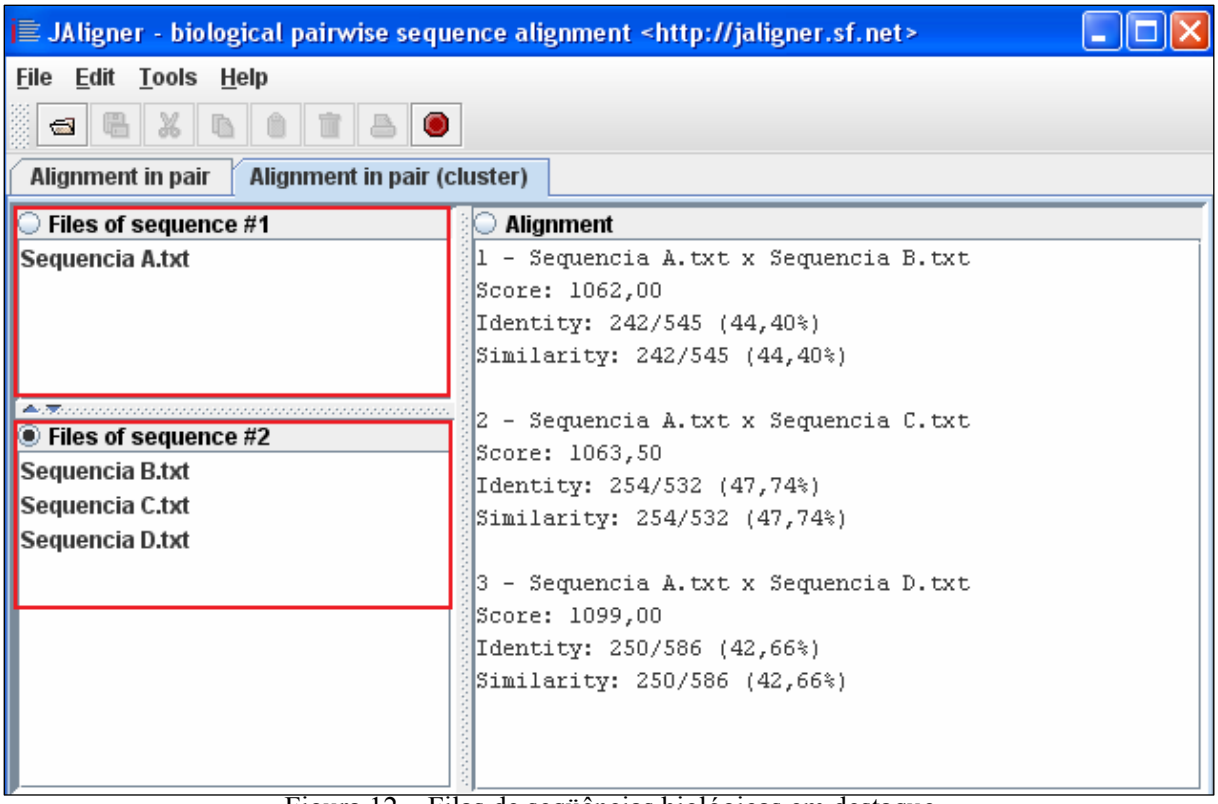


Figura 12 – Filas de seqüências biológicas em destaque

No quadro 24 é apresentado trecho de código responsável por carregar uma seqüência biológica na fila de seqüências.

```
// Cria objeto de arquivo de seqüência
FileOfSequence fs = new FileOfSequence();

// Seta nome da seqüência
fs.setNameOfFile(arqs[0].getName());

// Seta o caminho do arquivo de seqüência biológica no objeto
fs.setPath(arqs[0].getPath());

// Seta os caracteres que formam a seqüência
bf = new BufferedReader(new FileReader(arqs[0]));
buffer = new StringBuffer();
linha = bf.readLine();
while (linha != null){
  buffer.append(linha);
  linha = bf.readLine();
}
fs.setSequence(buffer.toString());

// Guarda objeto conforme fila de seqüência selecionada
if (currentlist == 1){
  modelSequence1.addElement(fs.getNameOfFile());
  listOfSequence1.add(fs);
}else{
  modelSequence2.addElement(fs.getNameOfFile());
  listOfSequence2.add(fs);
}
```

Quadro 24 – Trecho do código responsável por carregar uma seqüência biológica na fila de seqüências

O processamento do alinhamento, de cada tarefa JPVM, é realizado de forma paralela em cada nó do cluster. Isto é feito através do processamento paralelo com memória compartilhada, utilizando-se *threads*. Em torno disto, é preciso configurar quantas *threads* serão abertas para processar o alinhamento. No protótipo, a quantidade de threads deve ser informada através do campo *threads number*.

Os resultados dos alinhamentos são guardados em um diretório especificado pelo usuário. Este diretório deve possuir permissão para gravação de arquivos e deve ser especificado no campo *result directory*.

A figura 13 apresenta em destaque os campos *threads number* e *result directory*.

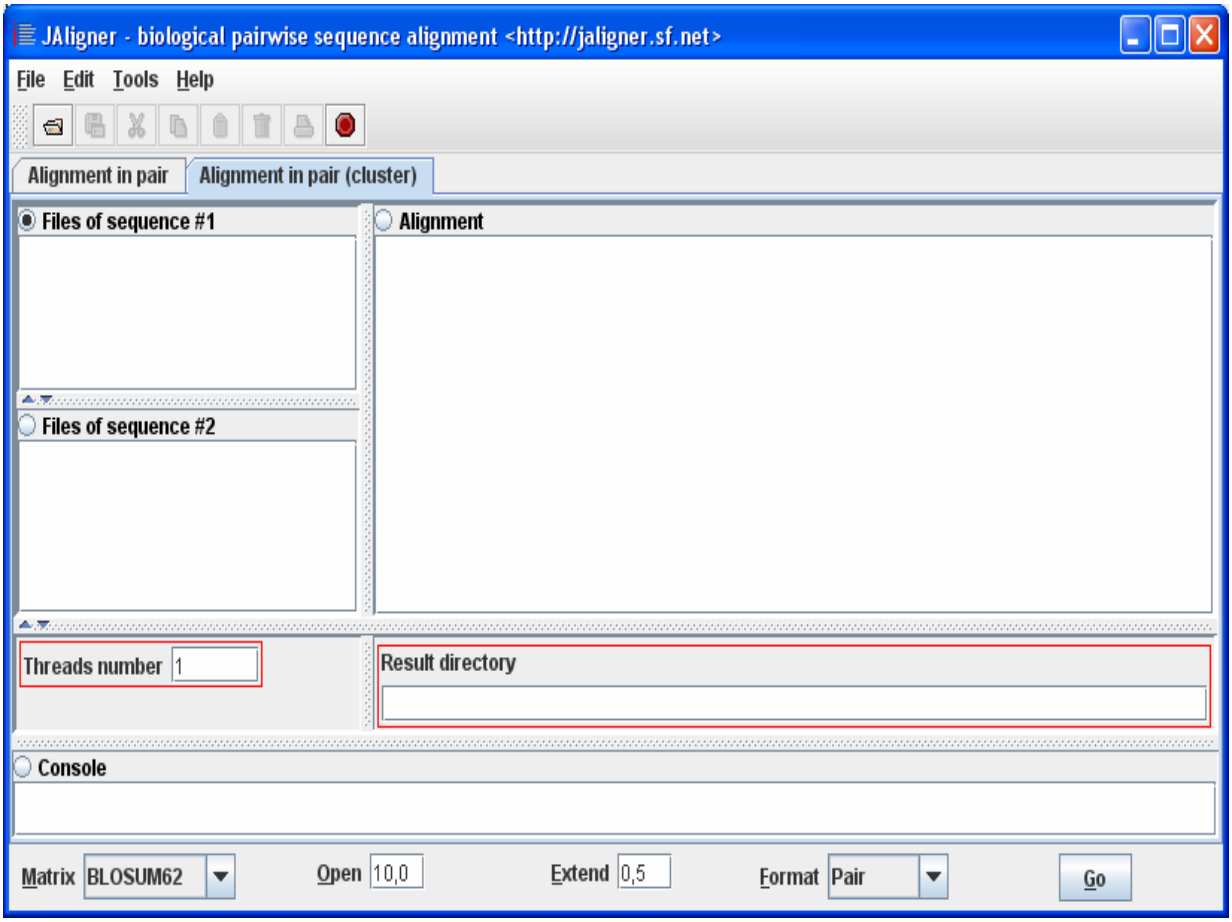


Figura 13 - Campos *threadsnumber* e *result directory* em destaque

3.3.5 Realizando processamento em ambiente distribuído

Para realizar o alinhamento de seqüências em ambiente distribuído e otimizar o processamento, foram utilizadas duas bibliotecas de programação paralela.

A primeira biblioteca, o JPVM, busca distribuir tarefas pelos nós do cluster. Cada tarefa tem o objetivo de realizar o alinhamento de um par de seqüências biológica. As seqüências que compõem as tarefas são obtidas das filas de seqüências, sendo que uma seqüência é pega de cada fila. Nunca são geradas tarefas repetidas, ou seja, cada tarefa contém uma combinação de seqüências específicas.

Quando as tarefas JPVM são criadas, em cada nó do cluster é instanciado a classe `SmithWatermanCore`. Esta classe contém implementado o algoritmo de Smith-Waterman, que irá realizar o alinhamento das seqüências. Após instanciar a classe, a mesma fica aguardando o recebimento de um pacote JPVM. Este pacote contém diversas informações essenciais para o alinhamento, como por exemplo, as seqüências que serão alinhadas e a matriz de pontuação.

O trecho do código fonte em que as tarefas JPVM são criadas pode ser visualizado no

quadro 25. No quadro 26 é apresentado trecho do código da classe `SmithWatermanCore` em que é aguardado o recebimento de uma tarefa de alinhamento, através do comando `jpvm.pvm_recv()`.

```
/* Registra-se na máquina paralela virtual */
jpvmEnvironment jpvm = new jpvmEnvironment();

/* Define quantidade de tarefas JPVM serão criadas */
jpvmTaskId tids[] = new jpvmTaskId[num_workers];

/* Instancia a classe SmithWatermanCore nos nós que
 * formam a máquina virtual paralela  */
jpvm.pvm_spawn("jaligner.SmithWatermanCore",num_workers,tids);
```

Quadro 25 – Trecho do código da classe `SmithWatermanGotoh` onde são criadas as tarefas JPVM

```
/* Registra-se na máquina paralela virtual */
jpvmEnvironment jpvm = new jpvmEnvironment();

/* Obtem ID do nó mestre */
jpvmTaskId parent = jpvm.pvm_parent();

/* Aguarda o recebimento de uma pacote JPVM que
 * contém sequências biológicas a alinhar */
jpvmMessage message = jpvm.pvm_recv();

/* Desempacota tarefa JPVM */
String recebe = message.buffer.upkstr();
```

Quadro 26 – Trecho do código da classe `SmithWatermanCore` onde é aguardado o recebimento de um pacote JPVM

A montagem do pacote JPVM ocorre no nó mestre do cluster, ou seja, no computador em que o protótipo do JAligner foi iniciado. Cada pacote é montado com um par de seqüências biológicas especificas e é enviado para um nó do cluster através do comando `jpvm.send`. O quadro 27 apresenta a montagem do pacote JPVM e o envio do mesmo para um nó do cluster.

```
/* Monta pacote contendo informações para alinhamento s*/
buf.pack(
    s1.getSequence() +"#"+
    s2.getSequence() +"#"+
    lin +"#"+
    col +"#"+
    Arrays.toString(serialMatrixScores) +"#"+
    o +"#"+
    e +"#"+
    Arrays.toString(pointers) +"#"+
    Arrays.toString(sizesOfVerticalGaps) +"#"+
    Arrays.toString(sizesOfHorizontalGaps));

/* Envia o pacote para um nó da máquina virtual paralela */
jpvm.pvm_send(buf, tids[iTids], 12345);
```

Quadro 27 - Trecho do código da classe `SmithWatermanGotoh` que envia pacote para um computador da máquina virtual

O pacote JPVM é composto por dados do tipo *string*. Desta forma, para facilitar a leitura das informações contidas no pacote, foi incluído entre cada informação um separador de texto. O separador utilizado foi o caractere `#`. Através da classe `StringTokenizer`, da linguagem java, foi possível separar facilmente as informações contidas no pacote em cada nó do cluster.

A segunda biblioteca utilizada foi o JOMP, onde foi possível paralelizar o processamento do alinhamento em cada nó do cluster. A paralelização foi obtida através do uso de seções, em determinados trechos do código, que podem ser executados simultaneamente.

A fase de inicialização do algoritmo de Smith-Waterman, descrita na seção 2.3.1 deste trabalho, pode ser paralelizada através do uso das diretivas `//omp parallel section` e `//omp parallel for`, conforme quadro 28.

```
  /* Inicializa matriz de alinhamento e demais matrizes utilizadas no processamento
*/

  //omp parallel sections
  {

    //omp section
    {
       String xToken = null;
       for (int y = 0; y < lin; y++) {
          for (int u = 0; u < col; u++) {
              xToken = strtokMatrix.nextToken();

              matrix[y][u] = Float.parseFloat(xToken);
          }
       }
     }

     //omp section
     {
       StringTokenizer strtokPointers = new StringTokenizer(sPointers, ",");
       for (int t = 0; t < pointers.length; t++) {
           pointers[t] = (byte)Float.parseFloat(strtokPointers.nextToken());
       }
     }

     //omp section
     {
       StringTokenizer strtokVerGap = new StringTokenizer(VerticalGap, ",");
       for (int t = 0; t < sizesOfVerticalGaps.length; t++) {
           sizesOfVerticalGaps[t] =
(short)Float.parseFloat(strtokVerGap.nextToken());
       }
     }

     //omp section
     {
       StringTokenizer strtokHorGap = new StringTokenizer(HorizontalGap, ",");
       for (int t = 0; t < sizesOfHorizontalGaps.length; t++) {
           sizesOfHorizontalGaps[t] =
(short)Float.parseFloat(strtokHorGap.nextToken());
       }

       g[0] = Float.NEGATIVE_INFINITY;
       h = Float.NEGATIVE_INFINITY;
       v[0] = 0;
     }

  }

  //omp parallel for
  for (int j = 1; j < n; j++) {
      g[j] = Float.NEGATIVE_INFINITY;
      v[j] = 0;
  }
```

Quadro 28 – Paralelização da fase inicial do algoritmo Smith-Waterman

Após o término da inclusão das diretivas, é necessário submeter o fonte ao compilador JOMP, com intuito de gerar o fonte na linguagem Java. O código gerado pelo compilador é pouco legível.

O número de *threads* abertas no processamento do alinhamento em cada nó do cluster é definido no campo *threads number*, do protótipo. Este valor é utilizado ao instanciar a classe `SmithWatermanCore`, através de um novo parâmetro na rotina de criação de tarefas JPVM. No quadro 29 é apresentada a rotina para criação de tarefas com o novo parâmetro.

```
// Create a thread to execute the new task
String args[] = new String[8];
args[0] = java_exec;
args[1] = "-Djpvm.daemon="+jpvm.pvm_mytid().getPort();
args[2] = "-Djpvm.parhost="+parent.getHost();
args[3] = "-Djpvm.parport="+parent.getPort();
args[4] = "-Djpvm.taskname="+name;
args[5] = "-Djpvm.regnum="+nextCreateOrder;
/* Parâmetro que informa quantidade de threads a serem criadas */
args[6] = "-Djomp.threads="+threadsNumber;
args[7] = name;
if(debug_on)
    log("exec( "+args[0]+" "+args[1]+" "+args[2]+" "+
        args[3]+" "+args[4]+" "+args[5] +" "+args[6] +" "+args[7] + " )");

jpvmExecTaskThread spawnThread;
spawnThread = new jpvmExecTaskThread(jpvm, client,
    args,createOrders[nextCreateOrder]);
nextCreateOrder++;
spawnThread.start();
```

Quadro 29 – Criando tarefa JPVM com número de *threads* específico

Após o processamento pelo nó do cluster, o mesmo envia um novo pacote contendo o resultado do alinhamento. O resultado do alinhamento ainda não é definitivo, é preciso executar a fase de *traceback*, ou seja, a terceira fase do algoritmo de Smith-Waterman. Na fase de *traceback* os resultados referentes à similaridade, identicidade e pontuação de cada alinhamento são obtidos.

3.3.6 Visualização dos resultados dos alinhamentos

Os valores resultantes do alinhamento do processamento são exibidos na ferramenta JAligner. São listados os valores referentes a similaridade, identicidade e pontuação. Não há uma ordenação especifica por alinhamento que apresentou maior pontuação. Os resultados são apresentados pela ordem em que os alinhamentos foram realizados, conforme as filas de seqüências. A figura 14 demonstra o resultado de um alinhamento de quatro seqüências.

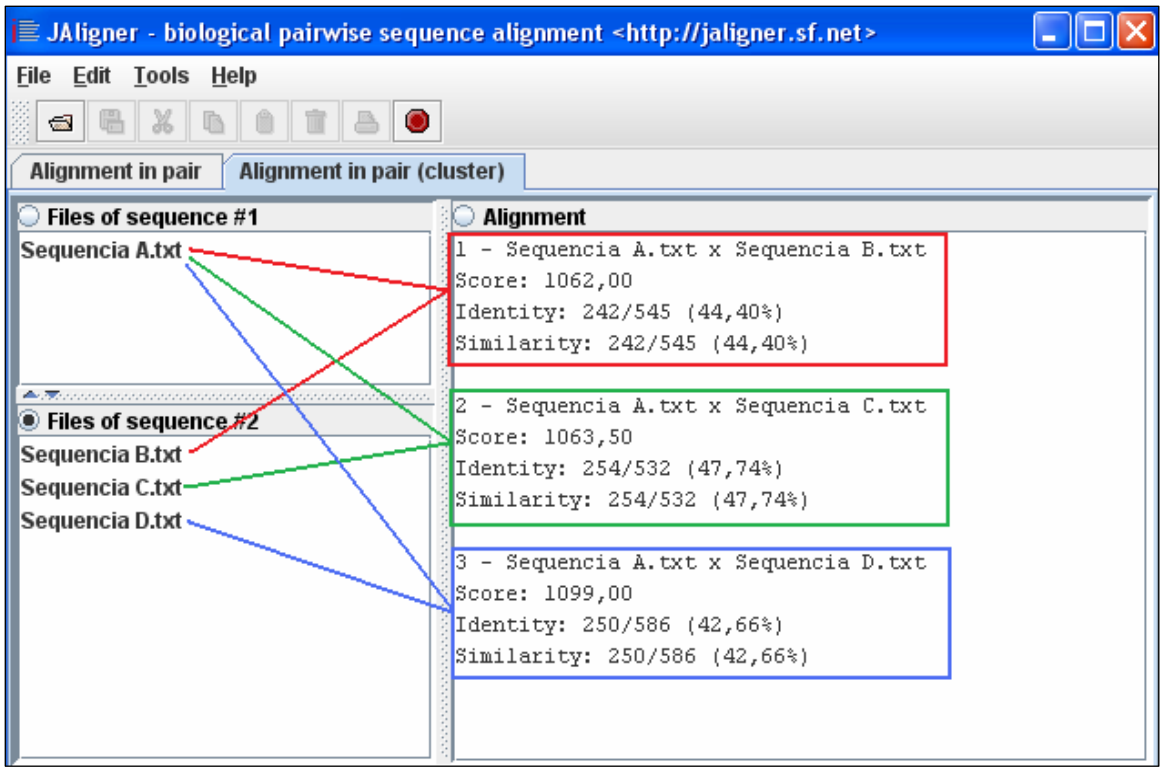


Figura 14 – Resultado do alinhamento de seqüências em ambiente distribuído

Os alinhamentos resultantes são armazenados no diretório especificado no campo *result directory*, no protótipo. Este diretório é utilizado como um banco de dados de arquivos simples, sendo conhecido como banco de dados biológico. Para cada alinhamento é gerado um arquivo específico, no formato definido antes de iniciar o alinhamento. Conforme resultados apresentados, o usuário pode optar por visualizar determinado alinhamento. O quadro 30 demonstra o resultado do alinhamento entre as seqüências `A` e `B`.

```
Sequence #1: Seqüência A.txt
Sequence #2: Sequencia B.txt
Length #1: 545
Length #2: 545
Matrix: BLOSUM62
Gap open: 10.0
Gap extend: 0.5
Length: 545
Identity: 242/545 (44,40%)
Similarity: 242/545 (44,40%)
Gaps: 116/545 (21,28%)
Score: 1062,00
Sequencia A      3 TAGAGCTCATTCCCTACGCCCCGACTCTGTCCTGGACAGCGTGCCCACCA   52
                   ||| |.||||..|..|.||..||     |.|||      |||..||..|.
Sequencia B      2 TAG-GATCATGGCAGAAGCATCG-----GGCCT------CGTCACCGTCT   39

Sequencia A     53 GCCATGGCGGGGCCCCGGGGCCTCC--TCCCACTCTGCC-----TCCTGG   95
                   |||.|...||....|.    .|||.  .||.|.|.|||.     ||.||.
Sequencia B     40 GCCTTTTAGGATATCT----ACTCAGTGCCGAATGTGCAGTTTTTCTTGA   85

Sequencia A     96 CCTT----CTGCCTGGCAGGCTTCAGCTTCGTCAGGGGGCAGGTGCTGTT   141
                   .|.|    .|||| ..||...|||.|..|||.|....|  ||||....||
Sequencia B     86 TCGTGAAAATGCC-ACCAAAATTCTGAGTCGGCCAAAG--AGGTATAATT   132

Sequencia A    142 CAAAGGCTGTGATGTGAAAACCACGTTTGTCACTCATGTACCCTGCACCT   191
                   ||.     ||.|..||.||   ..||||||   ||..|     .|.||||
Sequencia B    133 CAG-----GTAAACTGGAA---GAGTTTGT---TCGAG-----GGAACCT   166

Sequencia A    192 CGTGCGCGGCCATCAAGAAGCAGACGTGTCCCTCAGGCTGGCTGCGGGAG   241
                   .|.|.|.|....|..||||| |.|.|||     |||..|   ||..|.||
Sequencia B    167 TGAGAGAGAATGTATAGAAG-AAAAGTG-----CAGTTT---TGAAGAAG   207

Sequencia A    242 CTC---CCGGATCAGATAACCCAG-GACTGCCGCT-----------ACGA   276
                   |.|   ..|..|..||.|||.|.| .|...||.||           .|.|
Sequencia B    208 CACGGGAAGTTTTTGAAAACACTGAAAAAACCACTGAATTTTGGAAGCAA   257

Sequencia A    277 AGTACAGCTGGGGGGCTCTATGG----TGTCCATG------AGCGG---C   313
                   ..|...|.|||.|..|..|.||.    ..||||||      ..|||   .
Sequencia B    258 TATGTTGATGGAGATCAATGTGAATCCAATCCATGTTTAATGACGGTGTA   307

Sequencia A    314 TGCA-----GACGGAAGTGCC----GGAAGCAAG--TGGTGCAGAAGGCC   352
                   ||||     |||...|.|.||    .|.|.||.|  .|..|||...||||
Sequencia B    308 TGCAAGGATGACATTAATTCCTAATAGGATCATGGCAGAAGCATCGGGCC   357

Sequencia A    353 T-GCTGCC--CTGGCTACTGGGG----TTCCCGGTGCCATGAATGC----   391
                   | |...||  |||.||..|.||.    |.|.|.|||||.....|||
Sequencia B    358 TCGTCACCGTCTGCCTTTTAGGATATCTACTCAGTGCCGAATGTGCAGTT   407

Sequencia A    392 ---CCTGGGGGCGCTGAGACCCCATGCAATGGCCACGGGACCTGCTTGGA   438
                      |.||...|.|...|..||.|....|.|     |.|...|.||.....
Sequencia B    408 TTTCTTGATCGTGAAAATGCCACCAAAATT-----CTGAGTCGGCCAAAG   452

Sequencia A    439 TGGCATGG-ACAGG--AATGGGACCTGTGTGTGCCAGGAAAACTT       480
                   .||.||.. .||||  ||..|||...||.|||.|.|||.||.|||
Sequencia B    453 AGGTATAATTCAGGTAAACTGGAAGAGTTTGTTCGAGGGAACCTT       497
```

Quadro 30 – Exemplo de resultado do alinhamento entre seqüências

3.4 RESULTADOS E DISCUSSÃO

Através da implementação de rotinas utilizando bibliotecas para programação paralela, tornou-se possível realizar o alinhamento de seqüências biológicas em ambiente distribuído. Para verificar o ganho de *performance* com a utilização de bibliotecas de processamento paralelo, foram realizados testes específicos com o algoritmo de alinhamento e com o processo de alinhamento de várias seqüências em ambiente distribuído.

Nos testes realizados para verificar a *performance* do algoritmo, foram realizados alinhamentos com um mesmo par de seqüências, sendo que a primeira seqüência utilizada possuía 2641 caracteres e a segunda 2649 caracteres. O computador utilizado nos testes tem como configuração um processador Intel Core 2 Duo T5500 1.66 GHz e 2 GB de memória RAM. Os testes foram realizados alterando-se a quantidade de *threads* utilizadas na execução da classe `SmithWatermanCore`. Esta classe é responsável por realizar o alinhamento nos computadores que formam o cluster e utiliza a biblioteca JOMP.

Os resultados obtidos nos testes podem ser visualizados na tabela 1. A figura 15 apresenta de forma gráfica os resultados obtidos.

Tabela 1 - Resultados dos testes de *performance* do algoritmo de alinhamento

Número de *threads*	**Tempo (em milisegundos)**
1	7062
2	4984
3	4688
4	4468
5	4422
6	5438
7	5453
8	4891
9	5359
10	5734

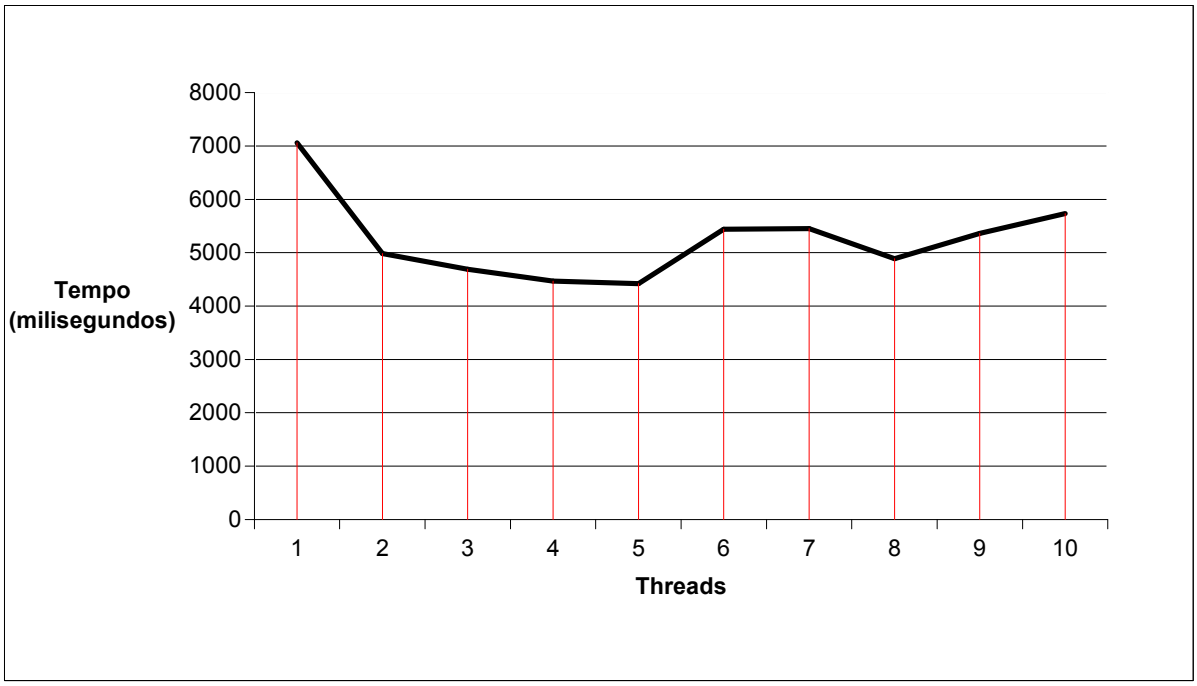


Figura 15 – Gráfico com resultados dos testes de *performance* do alinhamento com múltiplas *threads*

Analisando o resultado dos testes, é possível notar que existe um ganho de *performance* ao executar o alinhamento com mais de uma *thread*, ou seja, de forma paralela. Entretanto, nota-se que o ganho de *performance* é crescente ao utilizar até cinco *threads*. Isto ocorre por que o número máximo de seções que podem executar paralelamente é igual a quatro, conforme implementação realizada no algoritmo de alinhamento com a biblioteca JOMP, e a inicialização da matriz de alinhamento ocorre em um *loop* paralelizável, que se beneficia do uso de várias *threads*. Ao informar mais do que cinco *threads* existe uma perda crescente de *performance* devido ao tempo necessário para criar as *threads* e dividir o processamento. Mesmo assim, é possível notar que existe um ganho de *performance* considerável, mesmo ao executar o alinhamento com dez *threads*.

Nos testes para verificar a *performance* do alinhamento de várias seqüências em ambiente distribuído foi criado um cluster com até três computadores. Esses estavam conectados através de uma rede *wireless*, de 54 MBps. O quadro 31 apresenta a configuração dos computadores utilizados.

Computador	Processador	Memória RAM
1	SEMPROM 3400 1.8 GHz	1 GB
2	ATHLON X2 4000+ 2.0 GHz	2 GB
3	INTEL CORE 2 Duo T5500	2 GB

Quadro 31 – Configuração dos computadores utilizados nos testes

Foram realizados testes de alinhamento de quatro seqüências, onde a primeira seqüência foi alinhada com outras três seqüências. Em cada teste realizado, eram alterados os

tamanhos das seqüências. Inicialmente, foram realizadas testes com seqüências de 1000 caracteres, sendo que em cada teste foi adicionado um computador ao cluster. Em seguida, os mesmos testes foram repetidos, porém alterando o tamanho das seqüências para 1500 caracteres e posteriormente para 2000 caracteres.

Como o objetivo dos testes era verificar a *performance* ao realizar o alinhamento de várias seqüências em ambiente distribuído, o número de *threads* utilizado pela classe `SmithWatermanCore` sempre permaneceu igual a um.

Os resultados obtidos nos testes podem ser visualizados na tabela 2 e de forma gráfica através da figura 16.

Tabela 2 - Resultado dos testes de alinhamento em ambiente distribuído

Tamanho das seqüências (caracteres)	**Quantidade de computadores**		
	1	**2**	**3**
	Tempo (milisegundos)		
1000	109955	69656	63781
1500	371922	219750	180578
2000	882953	419031	335547

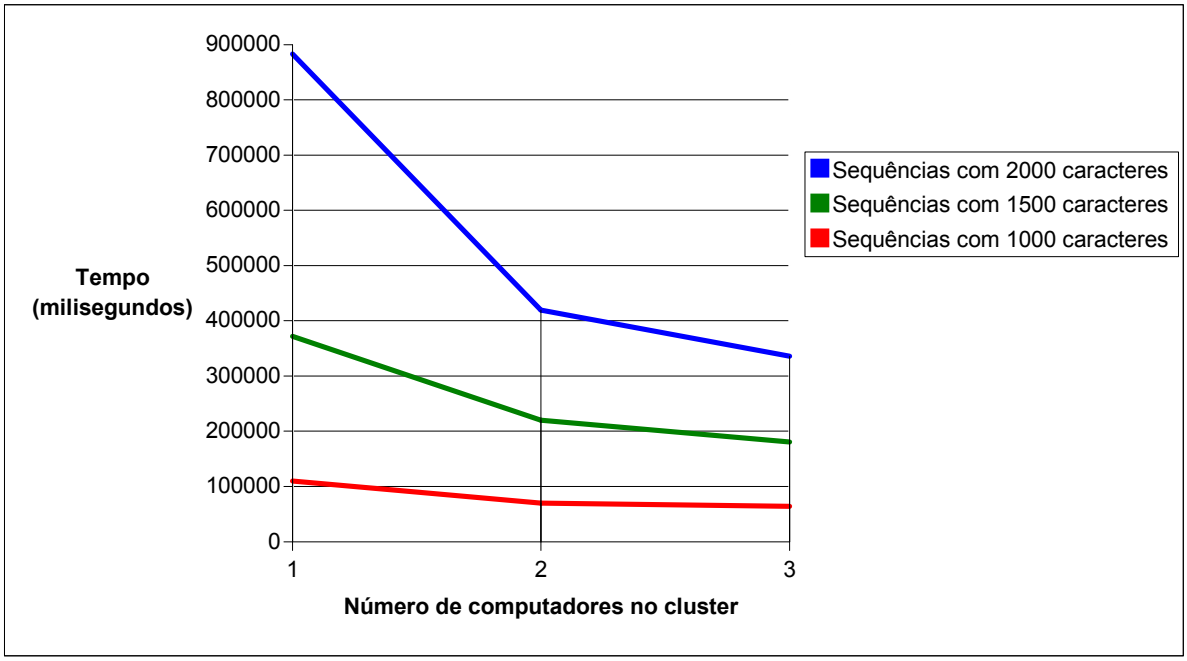


Figura 16 – Gráfico com resultados dos testes de *performance* do alinhamento em ambiente distribuído

Através dos testes realizados tornou-se evidente o ganho de *performance* obtido ao realizar o alinhamento das seqüências em ambiente distribuído. É possível notar que o ganho de *performance* cresce conforme o aumento do tamanho das seqüências e a quantidade de computadores utilizados no cluster.

Durante os testes foi possível notar o uso elevado da memória RAM. Isto ocorre devido a necessidade de armazenar várias matrizes na memória, utilizadas para realizar o alinhamento. O tamanho dessas matrizes aumenta conforme o tamanho das seqüências. Com o alinhamento em ambiente distribuído foi possível se beneficiar não só do poder de processamento mas também do uso da memória dos computadores do cluster.

Comparando o protótipo desenvolvido com os programas BLAST e FASTA, destaca-se o fato de que esses programas possuem heurísticas durante o processamento de alinhamentos de seqüências, o que os torna mais rápidos. Entretanto, o alinhamento resultante pode não ser o alinhamento ótimo, ou seja, pode não ser o melhor alinhamento possível entre duas seqüências biológicas analisadas. A ferramenta JAligner não possui heurísticas e sempre resulta o alinhamento ótimo.

O quadro 32 apresenta características das ferramentas JAligner, BLAST e FASTA.

Características	Trabalho proposto	Trabalhos correlatos	
	JAligner	**BLAST**	**FASTA**
Fila de seqüências biológicas	X	X	X
Alinhamento ótimo	X		
Programação dinâmica	X		
Uso de heurísticas		X	X

Quadro 32 – Comparação entre as ferramentas JAligner, BLAST e FASTA

4 CONCLUSÕES

Os objetivos propostos neste trabalho foram atingidos plenamente. Através do uso de bibliotecas de programação paralela foi possível adaptar uma ferramenta de alinhamento de seqüências biológicas *open source* para que realize o processamento em ambiente distribuído.

A tarefa de alinhar seqüências é de grande importância na área da bioinformática. Através do alinhamento de seqüências é possível inferir sobre as propriedades de determinada molécula, baseando-se em propriedades conhecidas de outras. Esta é uma tarefa que necessita de computadores de alto desempenho. A possibilidade de criar uma máquina virtual paralela é de grande utilidade. Proporciona o uso de diversos computadores, antes ociosos, para realizar a tarefa de alinhar seqüências biológicas.

As bibliotecas de programação paralela utilizadas demonstraram-se adequadas para que os objetivos propostos neste trabalho fossem alcançados. Em relação à biblioteca JOMP, verificou-se que o uso de diretivas para paralelizar algoritmos facilita a manutenção do código fonte. Esta biblioteca demonstrou-se eficiente, possibilitando um melhor aproveitamento dos recursos computacionais. Torna-se claro que seu uso pode se estender para ferramentas de outras áreas, como por exemplo, para ferramentas de geoprocessamento na qual necessitam de alto poder de processamento.

A biblioteca JPVM demonstrou-se uma boa alternativa para a busca de alto desempenho. A *performance*obtida com seu uso foi satisfatória. Esta biblioteca se destaca pela portabilidade oferecida, na qual é possível utilizar computadores de diversas arquiteturas para formar a máquina virtual paralela.

A medida que novas seqüências são descobertas, cresce a necessidade do desenvolvimento de ferramentas para bioinformática. Essas ferramentas possibilitam a realização de tarefas instigantes e complexas, gerando resultados potencialmente significativos.

A ferramenta JAligner se destaca por realizar o alinhamento ótimo entre seqüências. A possibilidade de realizar o processamento em ambiente distribuído, criar filas de seqüências e armazenar os resultados obtidos de forma automatizada, facilita a manipulação de seqüências biológicas pelos pesquisadores desta área.

4.1 EXTENSÕES

O alinhamento de seqüências é uma tarefa básica da área de bioinformática e sempre será necessária para inferir funcionalidades em novas seqüências descobertas. No entanto para inferir a evolução entre espécies é preciso realizar um alinhamento entre várias seqüências simultaneamente. Esta tarefa é chamada de alinhamento múltiplo em bioinformática e necessita de elevado poder de processamento computacional. Uma sugestão para trabalho futuro é adaptar a ferramenta JAligner para realizar alinhamentos múltiplos.

Em relação à manutenção da máquina virtual paralela, seria interessante incorporar a ferramenta JAligner meios de gerenciamento do cluster. Possibilitar a inclusão ou exclusão de determinadas máquinas no cluster e a visualização da distribuição de tarefas no cluster. Outra sugestão interessante é analisar a viabilidade e, se necessário, modificar a ferramenta para utilizar a plataforma de g*rid* computacional. A *grid* computacional possibilita o uso de grande quantidade de recursos heterogêneos e amplamente distribuídos.

Outra sugestão proposta é em relação ao resultado do alinhamento. Embora seja visualizado em um formato pré-determinado, seria interessante disponibilizar a opção para apresentar os resultados obtidos de forma gráfica.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBRECHT, F. F. **Reconstrução filogenética em ambiente distribuído**. 2006. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BULL, J. M. et al. **Towards openMP for java.** [S.l.], 2000. Disponível em: <http://www.compunity.org/events/pastevents/ewomp2000/BullSlides.pdf>. Acesso em: 10 out. 2007.

BULL, J. M.; KAMBITES, M. E. **JOMP an openMP-like interface for java**. [S.l.], 2000. Disponível em: <http://www.ukhec.ac.uk/publications/reports/acm2kprint.pdf>. Acesso em: 10 out. 2007.

BULL, J. M.; OBDRZALEK, J. **JOMP application program interface.** [S.I.], 2000. Disponível em: <www.lrr.in.tum.de/~gerndt/praktikumss01/jompapi.ps>. Acesso em: 11 out. 2007.

FERREIRA, V. J. M. F. **Uma introdução ao PVM**: como a paralelização pode ajudar a resolver problemas complexos de otimização. Rio de Janeiro, [1998?]. Disponível em: <http://www.dep.fem.unicamp.br/unisim/publicacoes/pesq_oper_pvm_1998.pdf>. Acesso em: 27 out. 2007.

GARCÊS, S. **Bionauta:** alinhamento de seqüências. [S.l.], 2007. Disponível em: <http://www.bionauta.com/as.html>. Acesso em: 6 abr. 2007.

GIBAS, C.; JAMBECK, P. **Desenvolvendo bioinformática:** ferramenta de software para aplicações em biologia. Rio de Janeiro: Campus, 2001.

LEMOS, M. **Workflow para bioinformática**. 2004. 239 f. Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

MEIDANIS, J.; SETÚBAL, J. C. **Uma introdução à biologia computacional**. Recife: UFPE-DI, 1994.

MORONY, C. S. **Implementação e desempenho em hardware dos algoritmos global e local de sequenciamento de gens**. 2006. 166 f. Dissertação (Mestrado em Ciências da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília.

MOUSTAFA, A. **JAligner**. [S.l.], [2003?]. Disponível em: <http://jaligner.sourceforge.net/>. Acesso em: 27 out. 2007.

OLIVEIRA, D. C. **Implementação de aplicações paralelas com o pacote JPVM**. 2005. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro Universitário Luterano de Palmas, Palmas.

PILONE, Dan; PITMAN, Neil. **UML 2**: rápido e prático. Rio de Janeiro: Alta Books, 2006.

PITANGA, M. **Computação em cluster**. [S.l.], 2003. Disponível em: <http://www.clubedohardware.com.br/artigos/153>. Acesso em: 10 out. 2007.

PROSDOCIMI, F. et al. Bioinformática: manual do usuário. **Biotecnologia ciência e desenvolvimento**, [S.l.], n. 29, p. 18-31, jan. 2003. Disponível em: <http://www.biotecnologia.com.br/revista/bio29/bioinf.pdf>. Acesso em: 27 maio 2007.

SMITH-WATERMAN Algorithm. In: WIKIPEDIA, the free encyclopedia . [S.l.]: Wikimedia Foundation, 2006. Disponível em: <http://en.wikipedia.org/wiki/Smith-Waterman_algorithm>. Acesso em: 4 nov. 2007.

SOUTO, M. C. P. **Banco de dados biológico.** [S.l.], [2004?]. Disponível em: <http://www.dimap.ufrn.br/~marcilio/BIOINFORMATICA/BIO2004.1/BIO-aula-05-busca.ppt>. Acesso em: 15 out. 2007.

TANENBAUM, A. S. **Sistemas operacionais modernos**. 2. ed. Rio de Janeiro: Prentice Hall do Brasil, 2003. 695 p.

VIVEIROS, P. N. **Uma introdução a bioinformática:** interpretando o algoritmo e o programa BLAST. 2006. 63 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Curso de Ciências da Computação, Centro Universitário Anhanguera, São Paulo.

VOGT, C. **Bioinformática, genes e inovação**. [S.l.], 2003. Disponível em: <http://www.comciencia.br/reportagens/bioinformatica/bio01.shtml>. Acesso em: 10 out. 2007.