UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS BIOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM PARASITOLOGIA

DESENVOLVIMENTO DE FERRAMENTAS PARA A IDENTIFICAÇÃO DE MARCADORES MOLECULARES E IMUNOLÓGICOS A PARTIR DE DADOS GENÔMICOS COMO ALVO PARA O DIAGNÓSTICO DE DOENÇAS PARASITÁRIAS

Robson da Silva Lopes

Belo Horizonte
2015

Robson da Silva Lopes

DESENVOLVIMENTO DE FERRAMENTAS PARA A IDENTIFICAÇÃO DE MARCADORES MOLECULARES E IMUNOLÓGICOS A PARTIR DE DADOS GENÔMICOS COMO ALVO PARA O DIAGNÓSTICO DE DOENÇAS PARASITÁRIAS

Tese apresentada ao Programa de Pós-graduação em Parasitologia do Instituto de Ciências Biológicas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do título de Doutor em Ciências - Parasitologia.

Área de concentração: Imunoparasitologia

Orientadora: Prof.ª Dr.ª Daniella Castanheira Bartholomeu

Universidade Federal de Minas Gerais
Instituto de Ciências Biológicas
Departamento de Parasitologia

Belo Horizonte
2015

DEDICATÓRIA

À minha família

AGRADECIMENTOS

Agradeço

A Deus por minha vida.

A meus pais Aluísio Lopes Ricardo e Elzenice Nunes da Silva, bem como minhas irmãs e sobrinhos pela dedicação, confiança, apoio, carinho e amor incondicional em todos os momentos.

A minha esposa Pollianna Alves da Silva e minha querida filha Eduarda Alves Lopes pelo carinho e amor durante toda esta etapa da minha vida.

A todos meus companheiros de profissão na UFMT e colegas de DINTER que durante todo o período do doutorado compartilharam de muitos momentos de alegria e, também, de muito trabalho árduo.

Aos pesquisadores/alunos e amigos que fiz no Laboratório de Imunologia e Genômica de Parasitos (LIGP) da UFMG, Tiago Antônio de Oliveira Mendes, Gabriela Rodrigues-Luiz, João Luís Reis Cunha e Rodrigo de Almeida e Lourdes. Pessoas estas com quem aprendi muito, se mostraram companheiros e que direta ou indiretamente contribuíram para a realização deste trabalho. A Michele Silva de Mato pela gentileza, carinho e amizade com que sempre me recebeu e atendeu no LIGP.

As pessoas que me encaminharam, incentivaram e ajudaram a concluir mais uma etapa na minha formação, Dr. Eduardo Luzia França, Dr.ª Adenilda Cristina Honório-França, Dr.ª Marly Augusta Lopes de Magalhães e a Me. Glaucia Margareth Rocha Olivieri (*in memoriam*).

A CAPES, pelo apoio financeiro durante o desenvolvimento do doutorado.

Principalmente, à professora Dr.ª Daniella Castanheira Bartholomeu pela orientação, amizade e profissionalismo no desenvolver deste trabalho, pela confiança em orientar alguém que veio de outra área de conhecimento e no mais pelo apoio, paciência e amizade.

EPÍGRAFE

“...

É preciso acelerar

Seminários, pesquisas, provas, professor que nunca informa a nota

Vou pirar...

E assim, em meio a tantos atropelos,

Aparadas as arestas

Finalmente

Cerimônia, fotos, família, que maravilha vamos comemorar!”

(Gláucia Rocha Olivieri)

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AAR	Amino acid ration
AFLP	Amplified Fragment Length Polymorphism
AUC	Area Under a Curve
BEEPro	B-cell epitope prediction by evolutionary information and propensity scales
DNA	Deoxyribonucleic Acid
EBV	Epstein-Barr Virus
ELISA	Enzyme-Linked Immuno Sorbent Assay
ELISPOT	Enzyme-Linked ImmunoSpot
EuPathDB	Eukaryotic pathogen database
GI	genInfo identifier
HIV	Human Immunodeficiency Virus
HLTV	Human T-cell Lymphotropic Virus type
IEDB	Immune Epitope Database
KW	Kruskal-Wallis
LCDE	Lista circular duplamente encadeada
LIBSVM	Library for SVM
MCC	Matthews correlation coefficient
MHC	Major histocompatibility complex
NCBI	National Center for Biotechnology
PCR	Polymerase Chain Reaction
ProGeRF	Proteome and Genome Repeat Finder
PSSM	Position-specific scoring matrix
RAM	Random Access Memory
RAPD	Random Amplified Polymorphic DNA
RBF	Radial Basis Kernel
RFLP	Restriction Fragment of Length Polymorphism
RNA	Redes Neurais Artificiais
RNA	Ribonucleic Acid
ROC	Receiver Operating Characteristics
SBP	SVM BeeproPipe
SCAR	Sequence Characterized Amplified Regions
SSR	Simple Sequence Repeats
STR	Short Tandem Repeats

STS	Sequence Tagged Sites
SVM	Support Vector Machine
UPEC	Uropatógeno Escherichia Coli

RESUMO

Doenças infecciosas causadas por parasitos protozoários apresentam-se como um grande problema de saúde pública, principalmente, em países pobres e em desenvolvimento, causando milhões de mortes por ano. Para o controle e monitoramento efetivo dessas doenças é essencial que se disponha de métodos de diagnósticos cada vez mais precisos. Identificar marcadores imunológicos e moleculares, como epítopos e microssatélites, permite uma rápida seleção de potenciais alvos que podem ser utilizados em diagnósticos, protocolos vacinais e como imunoterapêuticos. Métodos experimentais de identificação de marcadores imunológicos e moleculares apresentam elevado custo e requerem longos períodos de experimentação. Uma alternativa empregada é a utilização de métodos *in silico* de identificação destes marcadores dada à grande quantidade sequências genômicas e proteômicas disponíveis em bancos de dados públicos. Muitos métodos e ferramentas foram desenvolvidos nos últimos anos com o objetivo de identificar marcadores imunológicos e moleculares. No que se refere à preditores de epítopos, ferramentas como PREDITOP, PEOPLE, BEPITOPE, BepiPred, ABCpred, BCPred, BayesB e BEST foram desenvolvidas utilizando técnicas de inteligência artificial que permitem aos algoritmos aprender com informações já existente, isto é, aprendizado de máquina. No entanto, estas ferramentas baseadas em técnicas de aprendizado ainda apresentam uma baixa taxa de acurácia na predição quando aplicadas em dados de protozoários, devido à baixa quantidade de dados de protozoários usada no treinamento. Para identificação de marcadores moleculares do tipo repetições em tandem, existem várias ferramentas disponíveis, tais como: IMEx, MISA, Mreps, SciRoKo, Sputnik e TROLL. No entanto, além de identificar as repetições faz-se necessário que as ferramentas apresentem outros recursos que auxiliem os pesquisadores no reconhecimento e análise dos marcadores moleculares. Diante disto, este trabalho vem apresentar o desenvolvimento de ferramentas computacionais para a identificação de marcadores moleculares e imunológicos a partir de dados genômicos, a fim de buscar novos alvos para o diagnóstico de doenças parasitárias. Na primeira parte deste trabalho foi desenvolvido uma ferramenta web e stand-alone, intitulada *Proteome and Genome Repeat Finder (ProGeRF)*, capaz de identificar marcadores

de diagnóstico do tipo repetições em tandem. Em uma segunda etapa, buscou-se verificar se a etapa de treinamento de uma ferramenta de predição de epítopos lineares de células B com dados de diferentes táxons (bactérias, vírus e protozoários) pode influenciar, significativamente, na predição de epítopos de protozoários. A ferramenta ProGeRF proposta apresenta-se como uma ferramenta eficiente, rápida e precisa, seja nos modos *stand-alone* ou web, sendo a única que proporciona visualização gráfica e buscas com filtros nos resultados. Além de ser capaz de ser executada tanto em dados genômicos quanto em dados proteômicos e em grandes arquivos quando comparada com as ferramentas MISA, TROLL, TRF, Sputnik, GMATo e SciRoKo, ProGeRF consegue identificar uma quantidade maior de elementos repetitivos que as demais ferramentas em um tempo, consideravelmente, mais rápido. No que diz respeito à predição de epítopos lineares de células B em dados de protozoários, ferramentas treinadas apenas com dados de bactérias de maneira geral, conduz a uma predição, em média, próxima do aleatório. No entanto, quando treinado apenas com dados de vírus, algumas espécies de protozoários apresentaram significativo grau de eficiência. Todavia, melhores resultados foram apresentados quando a base de treinamento continha dados balanceados dos três táxons ou apenas dados de protozoários o que sugere uma real influência da base de dados de treinamento na acurácia de predição de epítopos lineares de células B de protozoários.

ABSTRACT

Infectious diseases caused by protozoan parasites is a major public health problem, especially in poor or developing, causing millions of deaths annually. For effective monitoring and control of these diseases it is essential to develop precise diagnostic methods. The identification of immunological and molecular markers, such as microsatellites and epitopes, allows rapid selection of potential targets that can be used in diagnostics, vaccination protocols and immunotherapeutic. Experimental methods for the identification of immunological and molecular markers have high costs and require long periods of experimentation. Given the large amount of genomic and protein sequences available in public databases, *in silico* methods for identifying these markers have been employed as an alternative approach. Recently, many methods and tools to identify immunological and molecular markers have been developed. Epitope prediction tools such as PREDITOP, PEOPLE, BEPITOPE, BepiPred, ABCpred, BCPred, BayesB and BEST were developed using machine learning techniques. These tools have better results in comparison to tools that do not use this approach. However, they are not accurate enough when applied to protozoa data, mainly due to the small protozoan datasets used for training. To identify tandem repeat markers, there are several available tools, such as IMEX, MISA, Mreps, SciRoKo, Sputnik and TROLL. However, besides identifying repetitive regions, other features have to be displayed to assist researchers in the recognition and analysis of molecular markers. In this work, we have developed tools for the identification of molecular and immunological markers from genomic data in order to seek new targets for the diagnosis of parasitic diseases. In the first part of this work, it presents a web and stand-alone tool, entitled ProGeRF (Proteome and Genome Repeat Finder), and developed to identify tandem repeats as molecular markers. The second part of this work aimed to verify whether the performance of *in silico* prediction tools of linear B-cell epitope could be impacted by using distinct training datasets (bacteria, viruses and protozoa). The ProGeRF tool is an efficient, fast, accurate, easy to use, either in stand-alone or web tool, provides a graphic display and allows filtering the results. Besides, it is able to run in large genomic and proteomic data. When compared with MISA, TROLL, TRF, Sputnik, SciRoKo and GMATo, ProGeRF can identify a larger number of repetitive elements and is faster than the majority of the other tools. Regarding the prediction of linear

B-cell epitope from protozoa data, tools trained with bacteria data generally leads to random predictions. However, when trained only with virus data, some species of protozoa showed a significant degree of efficiency during B-cell epitope prediction. The best results were obtained when the training dataset contained balanced data from the three taxon or data only from protozoa, highlighting the importance of optimizing the performance of tools for B-cell linear epitope using adequate training datasets.

1 INTRODUÇÃO

Doenças infecciosas causadas por parasitos protozoários, como doença de Chagas, malária e leishmaniose, têm causado milhões de mortes anualmente. O que faz com que estas apresentam-se como um grande problema de saúde pública, principalmente, em países pobres e em desenvolvimento (Ersfeld 2003, Monzote & Siddiq 2011), e sejam foco de intensos esforços de controle e erradicação dos seus organismos causadores

Para o controle e monitoramento efetivo das doenças provocadas por estes parasitos é essencial que se disponha de métodos de diagnóstico cada vez mais eficiente. Um diagnóstico que apresente alto grau de acurácia torna-se um instrumento vital para se conhecer a real prevalência da doença, avaliar com precisão a cura após intervenções terapêuticas e avaliar o sucesso das estratégias de controle.

Basicamente, pode-se classificar os métodos de diagnóstico em duas formas: a) Métodos de diagnósticos que se baseiam na detecção e identificação do parasito ou suas partes, como ovos ou fragmentos moleculares, sendo utilizados como prova de uma infecção recente ou ativa (Kasper et al. 2009) ou b) por meio de evidências indiretas da infecção parasitária, como pela demonstração de reatividade celular aos antígenos do parasito ou pela detecção dos anticorpos específicos, seja em processo de infecção aguda ou recente (IgM), ou uma infecção passada ou crônica (IgG) (Rabello 1990, Kasper et al. 2009).

Métodos convencionais de diagnóstico laboratorial de parasitos utilizam-se da identificação morfológica destes, como a microscopia óptica, por serem acessíveis e mais econômicos. No entanto, alguns problemas como a dificuldade de identificar morfologicamente o parasito e/ou de identificá-lo na fase crônica de algumas doenças podem diminuir a sensibilidade de tais métodos. Por exemplo, o exame parasitológico de fezes, utilizando-se da técnica de *Kato-Katz*, é a principal forma de diagnosticar a esquistossomose mansoni quando na fase aguda. No entanto, na fase crônica esta já não é tão eficaz (Rabello 1990).

Outro exemplo, a doença de Chagas, a qual tem como agente causador o protozoário *Trypanosoma cruzi*. Dependendo da fase da doença a forma de diagnóstico pode ser clínico-epidemiológico e laboratorial (Penna et al. 1998). Em se tratando da fase aguda da doença, onde existe um elevado número de tripomastigotas circulantes, são priorizados os exames parasitológicos para identificação do *T. cruzi* no sangue periférico pelos métodos de *Strout*, exame a fresco, gota espessa, esfregaço corado, creme leucocitário ou xenodiagnóstico. Todavia, na fase crônica, devido à baixa parasitemia e uma forte e persistente resposta humoral induzida pela infecção do *T. cruzi*, faz-se a pesquisa de anticorpos específicos contra antígenos do parasito (Vexenat et al. 1996, Penna et al. 1998, Mendes 2011).

Anticorpos são capazes de se ligarem de forma seletiva e específica ao antígeno por meio de regiões denominadas epítopos, ou determinantes antigênicos (Kindt et al. 2008). Com isto, as estratégias mais confiáveis de imunodiagnóstico de agentes patogênicos normalmente se baseiam na detecção de anticorpos que reconhecem estes determinantes antigênicos (Carmona et al. 2012).

Embora as interações antígeno-anticorpo sejam altamente específicas, em alguns casos o anticorpo induzido por um antígeno pode reagir de maneira cruzada com um antígeno não relacionado, causando a reatividade cruzada. A reatividade cruzada ocorre quando dois antígenos diferentes compartilham epítopos idênticos, ou quando anticorpos específicos se ligam a um epítopo não relacionado que possuem propriedades químicas similares (Kindt et al. 2008). Consequentemente, nestes casos os métodos de imunodiagnóstico apresentam problemas de especificidade.

Diversos trabalhos relatam que anticorpos nos soros de pacientes com doença de Chagas, Calazar e Leishmaniose muco-cutânea apresentam determinantes antigênicos comuns em várias proteínas dos protozoários causadores destas doenças. Desta forma, o imunodiagnóstico da doença de Chagas apresenta resultados falso-positivos devido à reação cruzada com parasitos do gênero *Leishmania* (Vexenat et al. 1996, Folgueira et al. 2010, Mendes 2011).

Todavia, é essencial que os epítopos sejam conservados em vários isolados de uma mesma espécie e únicos na espécie do patógeno, caso o objetivo seja empregá-los no desenvolvimento de testes de diagnósticos sensíveis e específicos (Hernández-Hernández & Rodríguez 2009).

Uma alternativa possível está no uso de diagnóstico molecular, o qual busca detectar o DNA (do inglês *Deoxyribonucleic Acid*) do parasito através da técnica de Reação em cadeia da polimerase (PCR, do inglês *polymerase chain reaction*) em amostras biológicas, como sangue, urina e soro. Isto é possível, pois embora os organismos compartilhem genes comuns em seu genoma (ortólogos), existem outros genes que não são compartilhados e apresentam sequências específicas de gênero, espécie e cepa, que permitem uma identificação inequívoca.

Marcadores moleculares podem ser definidos como sequências de ácidos nucléicos que diferenciam dois ou mais organismos (Ferreira & Grattapaglia 1995, Ferreira 1998). Repetições em tandem têm sido utilizadas com grande frequência como marcadores moleculares informativos, uma vez que possuem a capacidade de diferenciar agentes parasitários taxonomicamente muito próximos (Goto et al. 2006, 2008, 2011, Mesquita et al. 2010, Carmona et al. 2012).

Atualmente um grande número de genomas de diversas espécies já foram sequenciados e disponibilizados em base de dados públicos, como Banco de dados de patógenos eucarióticos (EuPathDB, do inglês *the eukaryotic pathogen database*), que alberga os genomas de diversos patógenos eucarióticos (Aurrecoechea et al. 2012). Com a disponibilidade destes dados, novas abordagens podem ser aplicadas para a identificação de determinantes antigênicos e marcadores moleculares, sendo que podem ser informativos para o diagnóstico de doenças infecciosas. Para tanto, tornou-se fundamental o desenvolvimento de ferramentas computacionais que permitam realizar a mineração de informações em um grande volume de dados genômicos.

Neste contexto, este trabalho vem apresentar duas novas ferramentas computacionais para extrair marcadores imunológicos e moleculares espécie-específica em dados genômicos e proteômicos para utilização em diagnósticos inequívocos e precisos.

2 FUNDAMENTAÇÃOTEÓRICA

2.1 MARCADORES IMUNOLÓGICOS

Localizar regiões expostas em proteínas que sejam capazes de desencadear uma resposta imune é parte central na compreensão do mecanismo do sistema de autoproteção do corpo (Yao et al. 2012). Logo, a identificação de determinantes antigênicos ou epítopos vem permitir que estes sejam utilizados no desenvolvimento de vacinas e no desenvolvimento de testes de diagnóstico sensíveis e específicos (El-Manzalawy et al. 2008, Ponomarenko & van Regenmortel 2009). Identificado um epítopo de natureza proteica, o peptídeo correspondente pode ser sintetizado quimicamente ou, alternativamente, a proteína que o contém pode ser expressa de forma heteróloga em bactéria ou levedura.

Antígenos recombinantes ou sintetizados são utilizados, preferivelmente, em vacinas e em imunodiagnóstico, devido o menor custo e por serem seguros, em contraste a patógenos atenuados ou moléculas bioquimicamente purificadas que requerem o cultivo dos parasitos e que podem oferecer risco de contaminação para o pesquisador (Ponomarenko & van Regenmortel 2009). Devido a estas razões, têm-se focado na utilização de peptídeos ou proteínas recombinantes como moléculas alvo para o diagnóstico de doenças parasitárias.

2.1.1 EPÍTOPOS

Epítopos são regiões de antígenos que apresentam a capacidade de se ligarem, especificamente, a receptores celulares ou aos anticorpos. Epítopos de células B se ligam a anticorpos e geralmente são compostos por aminoácidos hidrofílicos na superfície da proteína. Estes epítopos estão acessíveis aos anticorpos livres ou ligados a membrana, tendem a se localizar nas regiões flexíveis dos imunógenos e frequentemente apresentam mobilidade (Kindt et al. 2008).

Os epítopos de células B podem ser sequenciais ou descontínuos. Quando o epítopo é constituído por região linear de uma cadeia de aminoácidos, é chamado de epítopo linear. No entanto, se um epítopo é formado por aminoácidos

que não são contínuos na estrutura primária da proteína é chamado de epítopo conformacional.

Estudos cristalográficos de anticorpos indicam que cerca de 90% dos anticorpos produzidos no corpo podem ligar-se a epítopos conformacionais e apenas 10% se ligam a epítopos lineares (Pellequer et al. 1991, Ponomarenko & van Regenmortel 2009).

Atualmente, algumas técnicas experimentais estão disponíveis para mapeamento de epítopos de células B em laboratório, as quais são divididas em duas categorias: estrutural e funcional. Métodos estruturais incluem cristalografia, ressonância magnética e microscopia de elétrons (Kindt et al. 2008). Os métodos funcionais para detecção e caracterização de antígenos utilizam de técnicas como ressonância de plasma de superfície, espectrometria de massa, além de imunoensaios como ELISA (do inglês *Enzyme Linked Immunosorbent Assay*) e ELISpot (do inglês *Enzyme Linked immunospot*), *Western blot* e outros (Ponomarenko & van Regenmortel 2009).

Dadas as diversas abordagens de identificação de epítopos, as variadas formas de ensaios de validação usados e o crescente interesse de grupos de pesquisas no desenvolvimento de vacinas, um grande número de epítopos foram identificados nos últimos anos. A fim de consolidar todas estas informações em um único banco de dados foi criado em 2005 um banco de dados de epítopos de células B e T chamado IEDB (do inglês, *Immune Epitope Databases*) (Vita et al. 2010).

O IEDB utiliza algoritmos baseados em redes Bayesianas e máquinas de vetores suporte para localizar publicações científicas no PubMed que tenham identificado epítopos experimentalmente. As sequências dos epítopos de células B e T identificados nestes estudos são recuperadas (exceto epítopos de HIV, por existir uma base de dados específica) e armazenadas na base de dados do IEDB. Além do próprio epítopo são armazenadas os metadados, o que permite realizar consultas de epítopos pela sua estrutura, organismo fonte, restrição do MHC (do inglês *major histocompatibility complex*), tipo de ensaio ou organismo hospedeiro e outros critérios (Peters et al. 2005, Vita et al. 2010).

Segundo Vita et al. (2010) o propósito do IEDB é catalogar informações de epítopos experimentalmente identificados, seja de referências publicadas como submetida diretamente pelos pesquisadores que geraram o dado. Todavia, identificar experimentalmente epítopos não é tarefa fácil. Para os agentes patogênicos com genomas pequenos - por exemplo, vírus e pequenas bactérias – é possível sintetizar um conjunto de peptídeos que representam grande parte do proteoma e testá-los experimentalmente. Entretanto, quando aplicados a proteomas maiores, como os de parasitos, este tipo de abordagem requer muito tempo de experimentação e altos custos, tornando estas abordagens impraticáveis neste modelo (Carmona et al. 2012).

Com isto, e aliado à publicação de genomas completos de micro-organismos patogênicos, a predição *in silico* de epítopos de células B tem se tornado uma alternativa viável e muito utilizada para esta finalidade, pois permite que se identifiquem antígenos candidatos antes da realização de testes experimentais.

2.1.2 PREDIÇÃO *IN SILICO* DE EPÍTOPOS

A predição de epítopos de células B, basicamente, utiliza diferentes abordagens para cada um dos dois tipos, linear ou conformacional (Ponomarenko & van Regenmortel 2009). A predição de epítopos lineares requer apenas uma sequência de aminoácidos da estrutura primária da proteína, enquanto que a predição de um epítopo conformacional requer cálculo sobre os dados da estrutura 3D da proteína, processo este mais complexo (Davydov & Tonevitsky 2009). Desta forma, a grande maioria das abordagens *in silico* destinadas à predição de epítopos se concentram em epítopos lineares, devido sua menor complexidade (Yang & Yu 2009).

Os estudos de predição de epítopos lineares têm utilizado como principais informações de classificação as propriedades dos aminoácidos presentes nas sequências, tais como hidrofilicidade, acessibilidade solvente, estrutura secundária, e flexibilidade (Davydov & Tonevitsky 2009, Yao et al. 2012). Estas propriedades são empregadas como fonte de treinamento em ferramentas como PREDITOP (Pellequer & Westhof 1993), PEOPLE (Alix 1999), BEPITOPE (Odorico & Pellequer 2003), BepiPred (Larsen et al. 2006), ABCpred (Saha &

Raghava 2006), BCPred (El-Manzalawy et al. 2008), BayesB (Wee et al. 2010a), BEST (Gao et al. 2012) e BEEPro (Lin et al. 2013). Estas utilizam técnicas de aprendizado de máquina, como redes neurais artificiais (RNA)(Saha & Raghava 2006), máquinas de vetores de suporte (El-Manzalawy et al. 2008, Yao et al. 2012, Lin et al. 2013) e modelos ocultos de Markov (Larsen et al. 2006), na identificação de padrões enriquecidos em epítopos.

Vários trabalhos de predição de epítopos (Lobo et al., Flower et al. 2010, He et al. 2010, Carmona et al. 2012) têm utilizado ferramentas distintas que buscam identificar nas proteínas de agentes patogênicos, diferentes características que cruzadas podem contribuir na identificação e mapeamento *in silico* de regiões antigênicas com maior precisão. Dentre estas características, pode-se destacar: abundância de expressão, localização subcelular das proteínas, repetições de aminoácidos em tandem (He et al. 2010), sítios de glicosilação, regiões desordenadas/desestruturadas, regiões de conservação de proteínas entre diferentes genomas, exclusão de sequências presentes em cepas não patogênicas e ortólogas ao hospedeiro (Lobo et al., Carmona et al. 2012, Goodswen et al. 2013).

Muitas destas ferramentas são *softwares* independentes, que não estão integrados, apresentam predições com grau de acurácia distintas e formatos de arquivos de resultados diversos. Tal situação torna custoso o trabalho de cruzar, de maneira não automatizada, os resultados dessas ferramentas para identificação de epítopos. Contudo, alguns grupos de pesquisas (Lobo et al., He et al. 2010, Carmona et al. 2010, 2012) têm desenvolvido *scripts* em linguagem de programação, chamados de *pipeline*, que executam cada ferramenta, armazena os resultados, realiza uma varredura (*parse*) nestes e cruzam os resultados das diferentes predições.

No trabalho de Carmona et al. (2010) foi desenvolvida uma *pipeline,* utilizando o preditor de epítopos de células B BepiPred (Larsen et al. 2006) para identificar antígenos peptídicos no genoma de *T. cruzi* com baixa reatividade cruzada a outras espécies. Para isto, além dessa ferramenta, diversas outras foram executadas para avaliar um conjunto de propriedades moleculares de cada proteína do genoma de *T. cruzi*, tais como a localização subcelular, nível de expressão, sítios de glicosilação, regiões desordenadas/desestruturadas, predição de epítopos de células B, similaridade de sequência com proteínas humanas e de *Leishmania*

(sequências geradoras em potencial de reações cruzada entre estas espécies). Neste trabalho, foram identificados 300 possíveis biomarcadores, sendo que 200 peptídeos foram selecionados para validação experimental. Resultados preliminares revelaram que ~ 25% dos peptídeos candidatos reagiram especificamente com soros de pacientes chagásicos.

He et al. (2010) desenvolveram um sistema *web* para projetos de vacinas chamado Vaxign, que teve como base a predição de epítopos de células T. Esta ferramenta realiza a predição de localização subcelular de proteínas, de hélices transmembranar, proteínas com função de adesina, conservação de proteínas com o ser humano e/ou em rato, epítopo de ligação ao MHC de classe I e classe II, bem como exclusão de sequências de cepas não patogênicas.

A ferramenta foi testada contra o uropatógeno *Escherichia coli* (UPEC) visando identificar possíveis candidatos a vacinas, e os resultados foram comparados a vários estudos experimentais (He et al. 2010). Esta ferramenta mostrou-se precisa e eficiente na identificação de antígenos vacinais. O trabalho de Goodswen et al. (2013) apresentou uma *pipeline* com a mesma finalidade, utilizando-se de um preditor de epítopos descontínuos de células B, e a testaram em 29 proteínas experimentalmente demonstradas serem potenciais candidatos à vacina para a toxoplasmose e neosporose. De 29 proteínas, 21 foram classificadas como potenciais candidatos a vacinas.

Apesar do BepiPred (Larsen et al. 2006) ser uma ferramenta bastante utilizada em estudos em larga escala por permitir que as buscas sejam realizadas em uma máquina local, trabalhos recentes (Wang et al. 2011, Lin et al. 2013, Goodswen et al. 2013) relatam que esta ferramenta apresenta baixa acurácia na predição de epítopos lineares de células B. A baixa acurácia do BepiPred especificamente em dados de protozoários se deve em grande parte ao fato que menos de 10% do conjunto de dados usado no treinamento da ferramenta é derivado deste táxon.

2.2 MARCADORES MOLECULARES

Marcadores moleculares podem ser definidos como todo e qualquer fenótipo molecular oriundo de um gene expresso ou um segmento de DNA que

diferenciam dois ou mais indivíduos e são herdados geneticamente (Ferreira & Grattapaglia 1995, MILACH 1998).

Os marcadores moleculares, basicamente, podem ser classificados em dois tipos: proteínas variantes (aloenzimas) e polimorfismo de sequência de DNA (Schlötterer 2004). As aloenzimas são produtos da expressão gênica sendo muito influenciadas pelo ambiente, o que faz com que apresentem algumas desvantagens com relação às demais técnicas de caracterização molecular utilizadas. Outrora, marcadores de DNA podem revelar diferenças entre os genótipos mais eficientemente, por caracterizarem diretamente o genoma do organismo, podendo não ser diretamente influenciados pelo ambiente.

As técnicas mais utilizadas para identificar marcadores de DNA podem ser classificadas como: os que hibridizam e os que amplificam. Os marcadores moleculares que hibridizam são baseados em polimorfismo em sítios de restrição, apresentam altos custos e são laboriosos. Todavia, seus resultados são bastantes consistentes, exemplo: RFLP (*Restriction Fragment of Length Polymorphism*). Marcadores baseados em amplificação são mais rápidos, mais sensíveis e metodologicamente mais simples, sendo mais utilizados. Entre os o marcadores que amplificam pode-se citar os seguintes tipos: RAPD (*Random Amplified Polymorphic* DNA), SCAR (*Sequence Characterized Amplified Regions*), STS (*Sequence Tagged Sites*) microssatélite ou SSR (*Simple Sequence Repeats*) e AFLP (*Amplified Fragment Length Polymorphism*) (Coutinho et al. 2009, Tavares et al. 2011).

Elementos repetitivos, tipo microssatélite e minissatélites, são encontrados em grande quantidade em genoma de eucariotos seja em regiões codificadoras ou em regiões intercodificadoras (Leclercq et al. 2007). Estas sequências apresentam grande número de alelos por lócus, são de natureza codominante que possibilitam a identificação de todos os alelos (variantes) para um mesmo gene/lócus e apresentam alto nível de polimorfismo genético. Além disso, as mesmas requerem métodos de tipagem simples, podendo ser detectados por PCR, o que fez com que estes se destacasse como poderosos marcadores moleculares (Oliveira et al. 2006, Sharma et al. 2007).

Em seres humanos estas sequências chegam a representar cerca de 7% do tamanho do genoma, com aproximadamente 260.000 repetições (Lim et al.

2013). Em parasitos protistas o número de repetições simples e transposons variam de 11% a 65% da proporção do DNA (Clayton 2010) onde em protozoários como *Theileria parva, Plasmodium* spp*., T. cruzi* e *Toxoplasma gondii* este valor varia entre 4% e 50% de sequências repetidas no genomas (Wickstead et al. 2003, Lopes et al. 2015).

2.2.1 SEQUÊNCIAS REPETITIVAS

Sequências repetitivas podem ser categorizadas em dois grupos: repetições em tandem e repetições dispersas. As repetições dispersas podem ser compostas por transposons, genes de RNA (do inglês *Ribonucleic Acid*) transportador e genes parálogos. As repetições em tandem incluem sequências de DNA ribossômica e DNA satélites (Wickstead et al. 2003).

Normalmente, as repetições em tandem são classificadas pela quantidade de nucleotídeos no motivo repetido, podendo ser mono, di, tri, tetra, penta e hexanucleotídeos e, também, de acordo com o tamanho do motivo repetitivo, podendo ser microssatélites, minissatélites e macrossatélites. Os microssatélites (também conhecidas como *Short Tandem Repeats* - STRs ou *Simple Sequence Repeats* - SSRs) são caracterizadas como pequenos trechos de sequências de DNA (usualmente < 200 bp), com motivos de nucleotídeos que variam de 1 a 6 bases. Os minissatélites são formados por grandes conjuntos repetitivos, com motivos de 5 a 25 bases e os macrossatélites são grandes regiões de repetições com tamanhos maiores que 25 pares de bases (Schlötterer 2000, Wickstead et al. 2003, Mudunuri et al. 2010).

Os microssatélites podem ser classificados como perfeitos, imperfeitos e compostos. Os perfeitos são formados de unidades repetitivas, as quais são idênticas umas com as outras. Por exemplo, CGATCGATCGAT é um microssatélite perfeito com um motivo 'CGAT', repetido três vezes, que é representado por $CGAT_3$. Uma região repetitiva imperfeita compreende unidades repetitivas que apresentam pequenas mutações, que podem ter sido causadas por inserções, deleções ou substituições. Por exemplo, CGATCGAACGAT, é um microssatélite imperfeito, com repetição do motivo CGAT que, no entanto, apresenta uma substituição de T por A na oitava posição. Os compostos compreendem sequências onde duas ou mais repetições são arranjadas

sucessivamente com ou sem nucleotídeos entre eles. Por exemplo, $(ATG)_nGCCTC(GC)_m$ é um microssatélite composto por dois microssatélites, sendo uma sequência de motivos ATG e outra de GC separada por cinco nucleotídeos (Mudunuri et al. 2010).

Schlötterer (2000) relata que microssatélites apresentam uma taxa de mutação de 10^{-6} a 10^{-2} por geração, o que é significativamente mais alta que as taxas de substituições em outras regiões e vem explicar seu alto grau de polimorfismo.

A principal explicação para a alta taxa de mutação de microssatélites é dada pelo modelo de deslizamento (*slippage*) da DNA Polimerase durante a replicação do DNA (Ellegren 2004). No entanto, Kelkar et al. (2004) recentemente mostraram que uma grande proporção da variação da taxa de mutação em microssatélites pode ser explicada pela característica intrínseca do lócus de sequências de microssatélites, tais como número de repetição, tamanho padrão e a própria sequência.

Com o advento da tecnologia de PCR, os microssatélites se tornaram um marcador genético altamente versátil e largamente utilizado. Edvinsson et al. (2006) compararam o gene não repetitivo B1 e o elemento repetitivo (529-bp) de *T. gondii*, os quais têm sido amplamente utilizados como marcadores moleculares em diagnóstico e notaram que o elemento repetitivo apresentou maior sensibilidade e especificidade no diagnóstico. Carnevale et al. (2004) utilizaram repetições em tandem de um segmento de DNA de *Plasmodium vivax*. O DNA de *P. vivax* foi detectado em amostras de sangue e em mosquitos, o que levou os autores a considerarem que tais repetições em tandem seriam bons alvos de diagnóstico molecular de *P. vivax* e análise epidemiológica em áreas de transmissão.

Além de serem marcadores moleculares com altos níveis de especificidade e sensibilidade, proteínas compostas de repetições em tandem são consideradas potentes antígenos de células B (Oliveira-Ferreira et al. 2004, Goto et al. 2006, 2008; daRocha et al., 2002). Certa quantidade de repetições em tandem foi identificada como alvos de respostas de células B em diversos organismos incluindo parasitos protozoários, como *T. cruzi* (Burns et al. 1992, DaRocha et al. 2002, Goto et al. 2008), *Leishmania donovani* (Goto et al. 2007), *Leishmania*

infantum (Goto et al. 2006), *Plasmodium vivax* (Oliveira-Ferreira et al. 2004) e *Plasmodium falciparum* (Dame et al. 1984, Koenen et al. 1984).

Goto et al. (2007) realizaram uma busca por regiões repetitivas no genoma de *L. infantum*, um dos agentes causadores da leishmaniose visceral. De 8191 genes analisados, 64 genes apresentaram repetições em tandem. Desses, 22 genes já haviam sido identificados como antígenos. Além disso, foi confirmado que soro de pacientes com leishmaniose visceral reconheceu algumas das proteínas que continham repetições em tandem ainda não caracterizadas com antígenos.

2.2.2 IDENTIFICAÇÃO DE REPETIÇÕES EM TANDEM

Dada a importância de identificar regiões repetitivas em genomas e proteomas, muitas ferramentas para identificação de regiões repetitivas foram desenvolvidas. Trabalhos conduzidos por Lim et al. (2013), Mudunuri et al.(2010) e Leclercq et al. (2007) revisaram e testaram as principais ferramentas para identificação de repetições em tandem.

As ferramentas mais usadas para extração de repetições em tandem de genomas são: TROLL (Castelo et al. 2002), MISA (Thiel et al. 2003), Mreps (Kolpakov et al. 2003), SciRoKo (Kofler et al. 2007), SSR Locator (da Maia et al. 2008), IMEX (Mudunuri & Nagarajaram 2007), GMATo (Wang et al. 2013) e INVERTER (Wirawan et al. 2010).

Basicamente, as ferramentas desenvolvidas utilizam dois tipos de algoritmos: a) algoritmo de busca combinatória, que realizam busca exaustiva de sequências inteiras e compara subsequências adjacentes para identificar repetições em tandem. Esta abordagem apresenta uma maior complexidade, demanda um tempo computacional exponencial, principalmente, quando as buscas envolvem repetições imperfeitas, degenerada ou sobreposta; b) Outros algoritmos varrem a sequência genômica para detectar regiões que podem ser microssatélites sobre uma determinada heurística ou regra estatística. Uma vez que não realizam uma busca exaustiva, estes algoritmos têm um custo computacional menor.

Além de localizar as regiões repetitivas, as ferramentas de identificação de repetições em tandem devem proporcionar aos pesquisadores que

estudam estas regiões, funções que facilitam a análise e utilização futura das sequências identificadas. Dentre várias funções podemos citar: filtro de repetições extraídas seja pelo motivo ou tamanho, conhecer o lócus da informação (se localiza em região codificadora ou não), visualizar alinhamento, desenhar *primers*, dentre outros. Mudunuri et al. (2010) revisaram as principais ferramentas de extração de microssatélites e listaram seus recursos como podem ser observados na Tabela 1.

Tabela 1 - Lista dos softwares de extração de microssatélites e seus importantes recursos. *Sim* indica que o recurso é parte do software; *Não* indica que o recurso está ausente na ferramenta particular.

Ferramenta de busca	Interface web	Perfeito	Imperfeito	Composto	Tamanho da repetição detectada	Busca repetição/tamanho específicos	Alinhamento	Opção de primer	Flanqueamento da sequência na saída	Informações de Padronização/estatísticas da repetição	Informação de lócus codificante e não codificante	Saída gráfica
ATR Hunter	Sim	Não	Sim	Não	1 – 500 bp	Não	Não	Não	Não	Não	Sim	Não
IMEx	Sim	Sim	Sim	Sim	1 – 6 bp	Repetição e tamanho específicos	Sim	Sim	Sim	Sim	Sim	Não
MISA	Não	Sim	Não	Sim	1 – 6 bp	Tamanho específico	Não	Sim	Não	Sim	Não	Não
Mreps	Sim	Sim	Sim	Não	1 – >100 bp	Não	Não	Não	Não	Não	Não	Não
Msatfinder	Sim	Sim	Sim	Sim	1 – 6 bp	Tamanho específico	Não	Sim	Sim	Sim	Não	Não
Poly	Não	Sim	Não	Não	1 – 4 bp	Não	Não	Não	Não	Sim	Não	Não
Repeat Masker	Sim	Sim	Sim	Não	Short (1-5), Interspersed	Não	Sim	Não	Não	Não	Sim	Não
SciRoKo	Não	Sim	Sim	Sim	1-6	Não	Não	Sim	Não	Sim	Não	Não
Sputnik	Sim	Sim	Sim	Não	1-5 bp	Não	Não	Não	Sim	Não	Não	Não
SSRF	Sim	Sim	Não	Não	1-6 bp	Não	Não	Sim	Sim	Não	Sim	Não
SSRIT	Sim	Sim	Não	Não	2-10 bp	Tamanho específico	Não	Não	Não	Não	Não	Não
STAR	Sim	Não	Sim	Não	1-9 bp	Tamanho específico	Sim	Não	Não	Não	Não	Não
STRING	Sim	Não	Sim	Não	1-150 bp	Não	Não	Não	Sim	Não	Não	Sim
TandemSWAN	Sim	Não	Sim	Não	3-100 bp	Não	Não	Não	Não	Não	Não	Não
TRF	Sim	Não	Sim	Não	1-2000 bp	Não	Sim	Não	Sim	Não	Não	Não
TROLL	Sim	Sim	Não	Não	1-5 bp	Repetição e tamanho específicos	Não	Sim	Sim	Não	Não	Sim

3 JUSTIFICATIVA E MOTIVAÇÃO

Muitas doenças infecto-parasitárias carecem de métodos de diagnóstico confiáveis, rápidos e precisos. Desta forma, a identificação de marcadores de diagnóstico sensíveis e específicos, sejam eles determinantes antigênicos ou moleculares, constitui uma instigante linha de pesquisa.

A identificação experimental destes marcadores é um processo demorado e custoso, uma vez que os genomas de alguns patógenos, como protozoários, são consideravelmente grandes. Neste sentido, foram desenvolvidas ferramentas baseadas em diversas técnicas computacionais com objetivo identificar *in silico* possíveis marcadores.

No que diz respeito a marcadores de imunodiagnóstico, as ferramentas desenvolvidas com o objetivo de predizer epítopos lineares de células B em proteínas têm apresentado uma baixa taxa de acurácia. Em trabalhos desenvolvidos até o momento, baseados em aprendizado de máquinas, quando a ferramenta apresenta boa especificidade, a sensibilidade é baixa e vice-versa (Larsen et al. 2006, Saha & Raghava 2006, Ponomarenko & van Regenmortel 2009, Zhao & Li 2010).

Vários fatores podem influenciar esta baixa acurácia, sendo que um deles está relacionado aos dados de treinamento. Sabe-se que o desempenho de métodos de predição de epítopos, baseados em técnicas de aprendizado de máquina, depende criticamente do conjunto de dados usados para o treinamento. Geralmente, estão sub-representados com dados de protozoários e, consequentemente, as ferramentas não conseguem reconhecer eficientemente os padrões dos dados genômicos e proteômicos destes micro-organismos.

Ademais, os marcadores imunológicos identificados por várias das ferramentas podem não ser espécie-específico, o que podem conduzir a diagnósticos de baixa acurácia devido à utilização de determinantes antigênicos que apresentam reatividade cruzada com outros patógenos.

No que se referem às ferramentas de identificação de marcadores moleculares do tipo repetições em tandem, as principais limitações identificadas

foram dificuldade identificar repetições perfeitas, imperfeitas, inserções, busca exaustivas, tempo de execução em grandes arquivos e impossibilidade de execução em proteínas.

Por exemplo, os programas MISA, Poly, SSRF, SSRIT e TROLL não identificam repetições imperfeitas, já os programas como Mreps e TandemSWAN consideram apenas substituições que causam imperfeições e não consideram inserções. As ferramentas TROLL, STAR e SSR scanner utilizam de motivos predefinidos para realizar a busca por microssatélites no genoma, o que não é conveniente em uma busca mais global. TROLL, TRF e Mreps filtram seletivamente e reportam apenas sequência com sobreposições com características pré-definidas no sistema, não permitindo que o usuário defina tal informação. As ferramentas TRF e Mreps utilizam algumas regras estatísticas para identificar as repetições, fazendo com que algumas sequências possam não ser identificadas por não serem validadas pelos testes estatísticos.

As ferramentas SSR Locator e SciRoKo são dependentes de plataforma e a ferramenta MISA não disponibiliza uma interface gráfica. Além disso, apenas duas apresentam saída gráfica, STRING e TROLL. No entanto, na primeira o tamanho da sequência de entrada é limitado a 150 caracteres e não trata inserções e repetição perfeitas, e a segunda identifica repetição de motivos entre um e cinco caracteres, não identifica imperfeições e inserções. Vale considerar que nenhuma das ferramentas disponíveis realiza busca de repetições tanto em dados proteômicos.

Há de se considerar, também, o tempo gasto no processo de identificação de repetições, principalmente em grandes arquivos, uma vez que este é utilizado como parte de processos maiores. Desta forma, é importante que a identificação de regiões repetitivas encontre a maior quantidade de repetições sejam elas perfeitas ou imperfeitas de forma rápida.

Assim, desenvolver uma ferramenta de identificação de regiões repetitivas que seja rápida, precisa, de fácil utilização, que realize busca exaustiva seja de repetições perfeitas bem como imperfeitas em dados genômicos, proteômicos e em grandes arquivos, aliado com uma ferramenta de predição *in silico* de determinantes antigênicos mais precisa em dados de protozoários podem contribuir na identificação de marcadores imunológicos espécie-específico.

4 OBJETIVO GERAL

Desenvolver ferramentas computacionais capazes de identificar marcadores de diagnóstico espécie-específico, sejam eles determinantes antigênicos ou marcadores moleculares do tipo repetições em tandem em genoma ou proteoma protozoários.

4.1 OBJETIVOS ESPECÍFICOS

4.1.1. Criar um algoritmo de identificação de regiões repetitivas perfeitas e imperfeitas, tanto em dados de genoma quanto em proteomas;

4.1.2. Desenvolver uma ferramenta *web*, com o algoritmo de identificação de regiões repetitivas, de fácil configuração, rápida e que apresente um layout *user-friendly* (autoexplicativo, fácil uso e amigável);

4.1.3. Comparar a ferramenta *web* de identificação de regiões repetitivas aqui desenvolvida com algumas ferramentas já disponíveis;

4.1.4. Criar uma base de dados local de epítopos de células de B de protozoários identificados experimentalmente e disponível na literatura, bem como em banco de dados de epítopos, como o IEDB;

4.1.5. Reprogramar alguns métodos de predição de epítopos de células B baseados em técnicas de aprendizado de máquina treinando-os, especificamente, com dados de protozoários;

4.1.6. Comparar as predições realizadas em um conjunto de dados de protozoários usando ferramentas treinadas e não treinadas com dados específicos de protozoários;

5 FERRAMENTA DE IDENTIFICAÇÃO DE REGIÕES REPETITIVAS

A ferramenta aqui implementada para identificação de repetições em tandem foi intitulada de ProGeRF (*Proteome and Genome Repeat Finder*). Esta ferramenta foi desenvolvida com objetivo de ser rápida, mesmo com grande quantidade de dados, sensível e flexível. O usuário pode definir a natureza da repetição (perfeita ou degenerada), o grau de degeneração assim com outros parâmetros de forma clara e aplicável tanto a dados genômicos quanto proteômicos. Esta parte do trabalho foi publicada na revista *BioMed Research International* (Lopes et al. 2015)(anexo 1).

5.1 MÉTODO

5.1.1 DEFINIÇÕES

Para a compreensão do trabalho algumas definições serão descritas a seguir.

5.1.1.1 Lista circular duplamente encadeada (LCDE)

Lista encadeada é uma estrutura de dados abstrata onde os dados são organizados em ordem linear, que simula uma representação de uma sequência de objetos na memória do computador. No entanto, os nós que armazenam os elementos consecutivos da sequência não ficam necessariamente em posições consecutivas da memória, fazendo-se necessário com que cada nó contenha um objeto de determinado tipo (informação que se deseja armazenar) e contenha o endereço para a célula seguinte.

É uma estrutura de dado flexível, uma vez que se pode alocar e desalocar memória de forma dinâmica, útil em situações que não se sabe *a priori* a quantidade de elementos que serão armazenados. Desta forma, a quantidade de memória utilizada é proporcional ao número de elementos armazenados. Diferente das listas simples, onde é necessário saber de antemão a quantidade de elemento a ser armazenado, pois os dados são armazenados em posições consecutivas de memória (GOODRICH & TAMASSIA, LEISERSON et al. 2002).

A Figura 1 apresenta um esquema das principais formas de listas encadeadas.

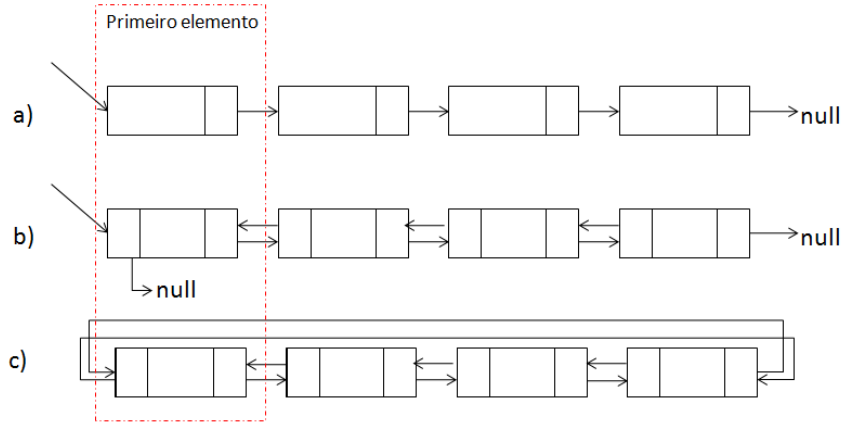


Figura 1 - Alguns tipos de listas encadeadas possíveis: a) Lista encadeada simples; b) Lista duplamente encadeada; e c) LCDE.

Em uma lista encadeada simples (Figura 1a), cada nó contém o endereço do nó seguinte e o último nó armazena no campo endereço o valor *null*. Desta forma para localizar um elemento seja para inserir ou excluir é necessário percorrer todos os nós no sentido do encadeamento, não sendo possível fazer o sentido inverso.

Listas duplamente encadeadas (Figura 1b), cada nó contém o endereço do nó seguinte e do anterior. Sendo que o primeiro e o último nó armazenam o endereço do nó anterior e do nó seguinte, respectivamente. Assim, dado que se deseja encontrar um elemento, é possível realizar uma busca nos dois sentidos. Ainda assim, para inserir um novo elemento em uma lista duplamente encadeada é necessário percorrer toda a lista, para chegar ao último elemento e adicionar o novo elemento após ele. Este processo pode se tornar muito demorado se a lista duplamente encadeada for grande.

Na lista circular duplamente encadeada (Figura 1c), o último elemento aponta para o primeiro e o primeiro aponta para o último. Desta forma, para encontrar o último elemento é necessário realizar apenas uma verificação no primeiro elemento, o que torna o processo de inclusão e exclusão de elementos mais rápido.

5.1.1.2 Janela deslizante

Para identificar repetições em uma sequência de DNA ou proteína é empregado a abordagem de janela deslizante. Esta abordagem consiste em percorrer a sequência selecionada, de maneira que a cada deslocamento da janela é selecionado Q elementos e dê origem a um motivo de tamanho j, onde $|Q| = j$.

A Figura 2 apresenta a sequência de nucleotídeos "ACTGCACTGCACTGC" percorrida por uma janela deslizante de tamanho $j = 5$. Em cada deslizamento uma subsequência (motivo) de tamanho j é extraído. Por exemplo, no primeiro deslizamento, tem-se a subsequência ACTGC.

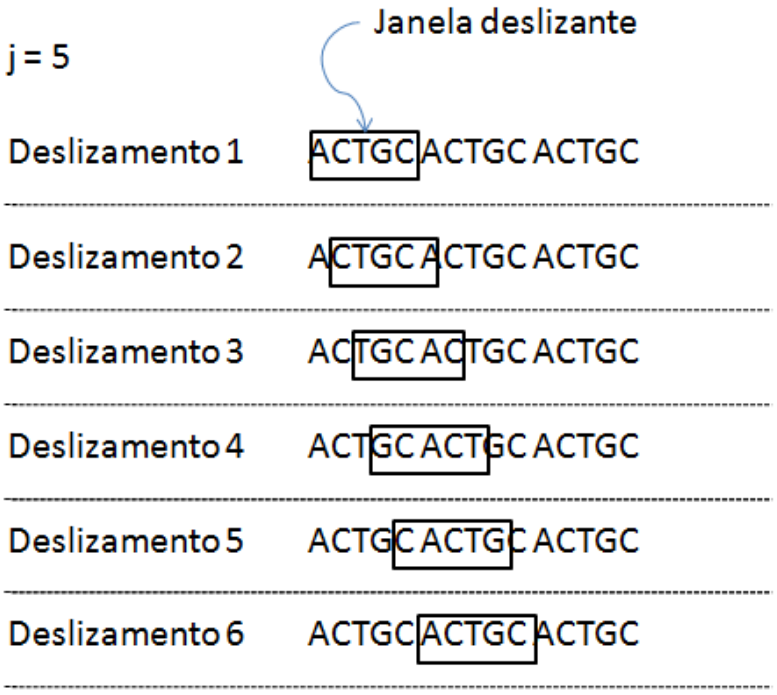


Figura 2 - Exemplo de janela deslizante de tamanho j=5

5.1.1.3 Tabela *hash*

Consiste de uma estrutura de dados do tipo vetor onde os dados são armazenados em uma sequência linear S com n elementos. Podemos nos referir a cada elemento individual k de S por meio de um índice chamado *chave*, pertencente ao intervalo $[0, n-1]$. No entanto, em tabelas hash a *chave* está associada ao elemento armazenado, de forma a tornar o processo de localizar, inserir ou excluir o elemento seja mais rápido. Em nosso caso, cada índice do vetor armazena informações sobre um motivo, sendo que cada motivo é associado a uma chave por meio de uma função *hash*, explicada mais adiante.

A tabela *hash* é alocada dinamicamente, contendo r^j posições, onde r é o número de caracteres possíveis, sendo quatro para DNA e vinte para aminoácidos. Desta forma, a tabela *hash* tem uma posição para cada combinação de nucleotídeos ou aminoácidos de tamanho $|Q|$.

Por exemplo, para uma janela deslizante de tamanho $j = 5$ em dados de DNA, será criado uma tabela *hash* com $4^5 = 1024$ posições.

5.1.1.4 Função hash

Funções *hash* transformam elementos de um universo arbitrário em inteiros positivos que indexem uma tabela - a tabela de *hash* (LEISERSON et al. 2002). Neste contexto, a função hash mapeia cada sequência de nucleotídeos ou aminoácidos para um valor numérico que será utilizado como *chave* para a tabela *hash*.

O mapeamento é baseado na conversão utilizada por Reneker e Shyu (2005), onde cada nucleotídeo em DNA ou aminoácidos em proteína é associado a um valor numérico, como segue:

- Nucleotídeos: m(A) = 0, m(C) = 1, m(G) = 2 e m(T) = 3;
- Aminoácidos: m(G) = 0, m(P) = 1, m(A) = 2, m(V) = 3, m(L) = 4, m(I) = 5, m(M) = 6, m(C) = 7, m(F) = 8, m(Y) = 9, m(W) = 10, m(H) = 11, m(K) = 12, m(R) = 13, m(Q) = 14, m(N) = 15, m(E) = 16, m(D) = 17, m(S) = 18 e m(T) = 19;

Em seguida, uma palavra de DNA ou aminoácido (janela deslizante) é mapeado em um valor numérico aplicando a função $h()$ de conversão, que toma como referência para o cálculo a posição de cada caractere na janela deslizante. Isto é, dado um motivo Q de tamanho $j = |Q|$ e $Q_p = \{q_0 q_1 q_3 \dots q_{|Q|-1}\}$, a função *h*() é definido pela Equação 1.

$$h(Q_p) = \sum_{i=0}^{|Q|-1} m(q_i) r^{(|Q|-1)-i} \qquad \text{Equação 1}$$

Onde Q é o motivo de DNA ou aminoácido, m é a função de associação de cada caractere a um valor numérico, q é um caractere no motivo na

posição i, p é a posição inicial do motivo dentro da sequência de DNA ou aminoácidos, r é o número de símbolos possíveis (quatro para DNA e vinte para aminoácidos) e $|Q|$ é o tamanho do motivo (neste caso o tamanho da janela deslizante). Por exemplo, o motivo ACTGC aplicado na função $h()$, tem como resultado o valor 121, uma vez que $h(ACTGC) = (0 * 4^4) + (1 * 4^3) + (3 * 4^2) + (2 * 4^1) + (1 * 4^0) = 121$.

5.1.1.5 Bucket simples

Consiste de 5-tupla que armazena informações de cada padrão repetitivo identificado. Ele é formado por <sp, fp, mt, g, lt>, onde sp e fp refere-se a posição inicial e posição final do motivo repetitivo dentro da sequência de DNA/aminoácido, mt é o motivo repetitivo, g é número de *gaps* na sequência repetitiva e lt é número de repetições.

Posição inicial	
Posição final	
Motivo	
gap	
Nº_repetições	

Figura 3 – Bucket simples que armazena informações de cada repetitivo. São armazenados a posição inicial e final do motivo repetitivo dentro da sequência de DNA/aminoácido, o próprio motivo repetitivo, o número de *gaps* na sequência repetitiva e o número de repetições.

Cada índice k da tabela hash de elementos repetitivos contém uma LCDE, sendo cada nó um Bucket simples de sequência repetitivas do motivo mapeado na posição k.

5.2 ARQUITETURA

ProGeRF está disponível em dois modos: como um programa *stand-alone*, isto é, de uma forma que o usuário possa fazer o *download* do programa e executar na máquina local e na forma de uma ferramenta web, como pode ser visto na Figura 4. A ferramenta *web* está disponível no endereço web http://64.79.105.19/ligp/. Neste endereço, também é possível realizar *download* da versão *stand-alone*.

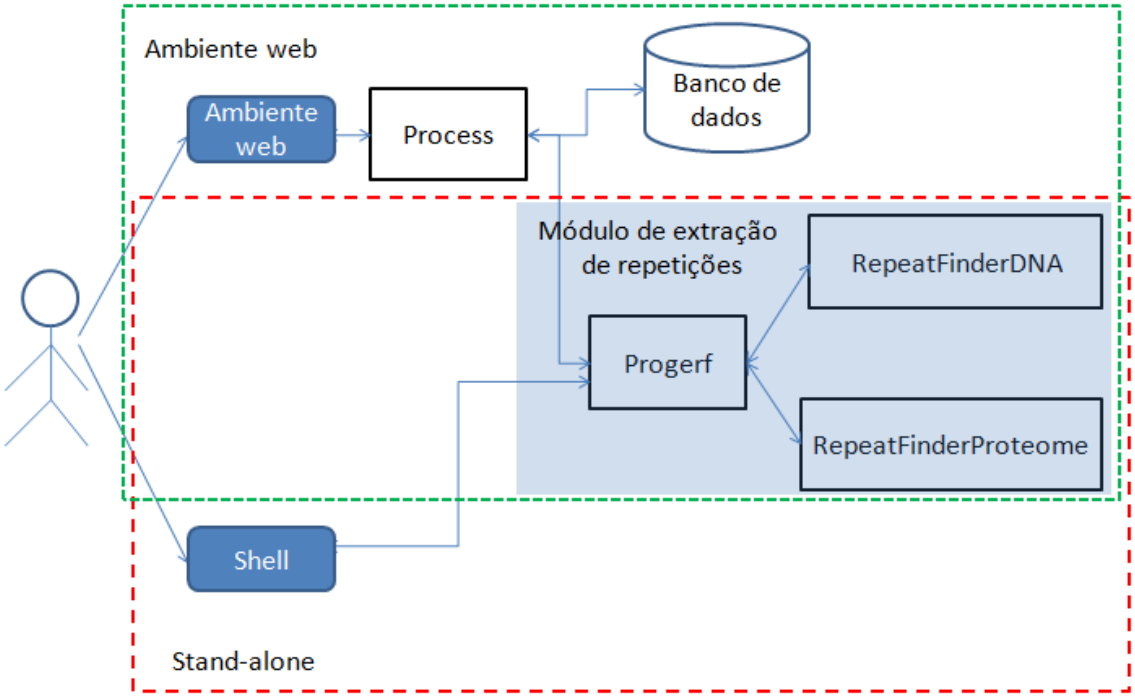


Figura 4 - Arquitetura do ProGeRF. Estrutura da ferramenta tanto para ambiente *web* quanto para o modo *stand-alone* Os retângulos azuis escuros com cantos arredondados representam a forma de interface com o sistema. Já os retângulos transparentes com *background* azul representam algoritmo implementado em C ou Perl.

O modo *web* apresenta uma interface amigável/agradável. Fernandes (2009) define interface amigável como interface capaz de disponibilizar estímulos visuais, como cores, formas, fontes, textos, texturas e outros elementos, de forma equilibrada e harmônica, visando não saturar a visão nem sobrecarregar a capacidade de assimilação dos sujeitos diante do crescente fluxo informacional. Este ainda diz que o design adequado da interface torna o sistema de computação menos cansativo e mais eficaz.

Para proporcionar um ambiente *web* amigável, a interface *web* foi desenvolvida utilizando os pacotes *bootstrap* (Sears et al. 2015) de formatação de *layout*, os *plug-ins* JBrowse (Skinner et al. 2009) e jqGrid (Tomov 2008).

Tanto o modo web como *stand-alone* utilizam os módulos de extração de repetição. Este módulo é composto de três algoritmos, sendo um desenvolvido em Perl e dois em linguagem C. Os algoritmos em linguagem C identificam repetições perfeitas e imperfeitas em arquivos no formato (multi)fasta. Um algoritmo chamado `repeatFinderDNA` identifica repetições em arquivo fasta com DNA e o

outro chamado `repeatFinderProteome` identifica repetições em arquivos fasta com aminoácidos.

Após a identificação realizada pelos algoritmos em C, o script em Perl, chamado `Progerf`, trata sobreposições, calcula estatísticas e gera o arquivo de saída.

O script *`process`* recebe os dados do ambiente *web*, trata, salva em um banco de dados *MySQL* e chama o módulo de identificação de repetições.

5.3 IDENTIFICAÇÃO DE REPETIÇÕES

Os algoritmos desenvolvidos em linguagem C recebem como parâmetros de entrada os seguintes dados:

-q <arquivo_fasta>

-o <nome_do_arquivo_de_saída>

-i <tamanho_da_subsequência>

-t <tamanho_mínimo_do_bloco_repetitivo>

-g <quantidade_máxima_de_gaps_entre_motivos>

-d <grau_de_degeneração>

Os quais podem ser chamados com seguinte comando:

```
./repeatFinderDNA –q sequence.fasta –o saida.ssr –i 5 –t 4 –g –d 0
```

para DNA ou

```
./repeatFinderProteome –q sequence.fasta –o saida.ssr –i 5 –t 4 –g –d 0
```

para arquivos com aminoácidos.

Os comandos acima realizam uma busca por repetições em tandem de tamanho 5, com no mínimo 4 repetições, no máximo 1 *gap* a cada motivo repetitivo no arquivo `sequence.fasta` e salva o resultado no arquivo `saida.srr.`

Os resultados são salvos no arquivo de saída em um formato tabular, onde a primeira coluna contém o identificador da sequência, a segunda o tamanho da sequência do DNA/proteína, a seguinte o mínimo de repetições permitidos, a

próxima a quantidade de repetições encontrada, a quinta e a sexta colunas representam as posições iniciais e finais do microssatélite, respectivamente. A penúltima coluna contém a quantidade *gaps* e a última, o motivo repetitivo. A Figura 5 apresenta um exemplo de resultado.

```
gi|11345450|ref|      141    4    5    122    141    0    LRPR
sp|014958|CASQ2_HUMAN 419    4    6    362    385    4    IEDV
sp|014958|CASQ2_HUMAN 419    4    6    361    384    4    WIED
```

Figura 5 - Saída do algoritmo repeatFinderProteome. A primeira coluna contém o identificador da sequência, a segunda o tamanho da sequência da proteína, a seguinte o mínimo de repetições permitidos, a próxima a quantidade de repetições encontrada, a quinta e a sexta colunas representam as posições iniciais e finais do microssatélite, respectivamente. A penúltima coluna contém a quantidade *gaps* e a última o motivo repetitivo

5.3.1 ALGORITMOS

Os algoritmos de identificação de repetições em tandem feitos em linguagem C são baseados em conjuntos dinâmicos do tipo lista circular duplamente encadeada, tabelas *hash* e funções *hash*.

5.3.1.1 Algoritmos de identificação de repetições em tandem perfeitas

O algoritmo 1 apresenta as etapas utilizadas para identificar repetições perfeitas. Para localizar as repetições em tandem, o algoritmo 1 recebe como dados de entrada o arquivo fasta, o tamanho da janela deslizante que representa o tamanho do motivo, o número máximo de *gap* permitido e o número mínimo de repetições.

O algoritmo 1 lê o arquivo fasta (linha 1) e em seguida aloca uma tabela hash de r^{j-1} posições (linha 2), onde r é quantidade caracteres, sendo r=4 para DNA ou r=20 para proteínas. Das linhas 3 a 24 o algoritmo identifica regiões repetitivas para cada uma das sequências no arquivo.

Algoritmo 1: Identifica Repetições Perfeitas

Entrada: Q : Arquivo (multi) fasta, O : nome do arquivo de saída, J: tamanho da janela deslizante, G: máximo de gaps permitido e R: número mínimo de repetições

Saída: O: Arquivo com repetições

```
1  ♯file = lerArquivo (Q);
2  ♯tabelaHash = criarTabelaHash ();
3  para cada sequencia ∈ ♯file faça
4      para ♯cont=1 até tamanho (sequencia)-J faça
5          ♯motivo = extrairJanela (♯cont,♯cont+J,sequencia);
6          ♯hashValue = calculaHash (♯motivo);
7          se ♯tabelaHash[♯hashValue] == NULL então
8              ♯tabelaHash[♯hashValue] = 1;
9              criarNoLCDE (♯tabelaHash[♯hashValue],♯cont,♯cont+J,0,♯ motivo);
10         senão
11             incremente (♯tabelaHash[♯hashValue]);
12             ♯posFinal = ♯tabelaHash[♯hashValue]→ultimoNo.PosicaoFinal;
13             se ♯posFinal >= ♯cont AND ♯posFinal + G <= ♯cont então
14                 ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoFinal = ♯cont + J;
15                 ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao =+1;
16                 ♯tabelaHash[♯hashValue]→ultimoBucket.Gaps =+ ♯cont - ♯posFinal;
17             senão
18                 se ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao >= R então
19                     criarNoLCDE (♯tabelaHash[♯hashValue],♯cont,♯cont+J,0,♯motivo);
20                 senão
21                     ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoInicial =
                       ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoFinal = ♯cont + J;
22                     ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao = 0 ;
23                     ♯tabelaHash[♯hashValue]→ultimoBucket.gap = 0 ;
24     salvar (♯tabelaHash,O);
25     limpa (♯tabelaHash);
```

O algoritmo percorre toda a sequência de DNA/proteína (linhas 4 a 23 do algoritmo 1), deslizando uma janela de tamanho *j* e extraindo um motivo de mesmo tamanho (linha 5 do algoritmo 1), calcula o valor hash e armazena em *hashValue* (linha 6 do algoritmo 1). Se for a primeira vez que a posição *hashValue* na tabela hash é acessada (linha 7 do algoritmo 1) então a tabela nesta posição recebe valor 1 (linha 8 do algoritmo 1) e é alocado um Bucket simples (linha 9 do algoritmo 1). Se não for a primeira vez (linha 10 do algoritmo 1), a tabela hash na posição *hashValue* é incrementada (linha 11 do algoritmo 1).

Desta forma, toda vez que é encontrado o mesmo motivo, este é contabilizado mais um em sua posição na tabela *hash*. Por exemplo, o

deslizamento 1 e 6 da Figura 6 extraem o mesmo motivo, com isto temos contabilizado 2 motivos na posição 121 da tabela *hash*.

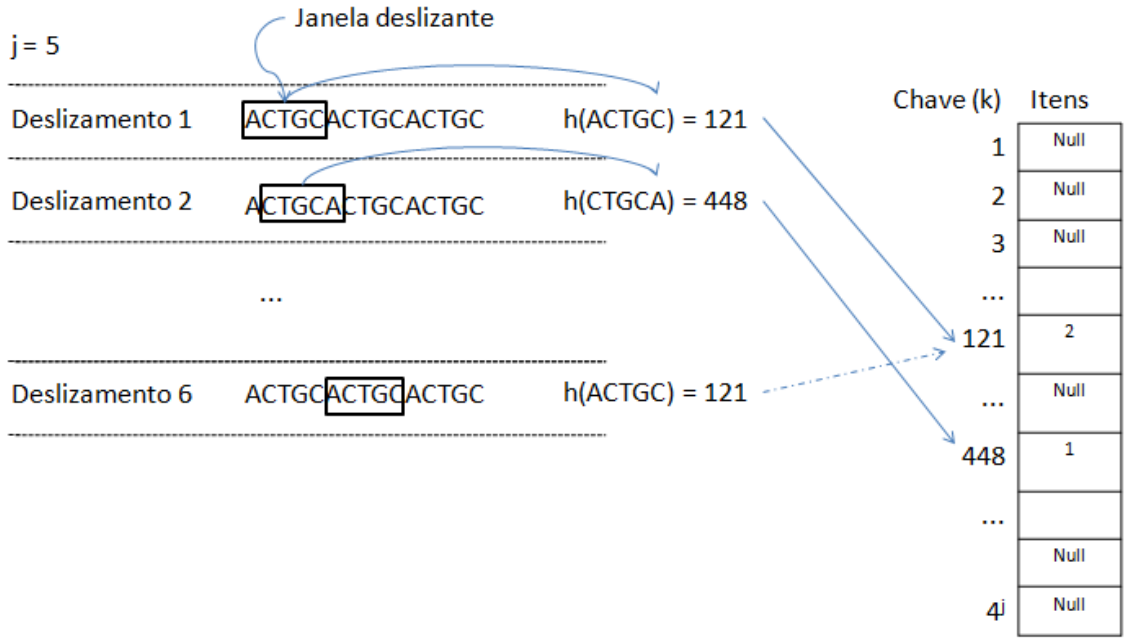


Figura 6 - Aplicação de função de *hash* em uma janela deslizante de tamanho 5. No deslizamento 1 e 6 o motivo extraído é ACTGC, o que resultada no mesmo valor hash, apontando para a posição 121 da tabela hash.

A linha 13 do algoritmo 1 verifica se a posição final do último Bucket da LCDE é maior que a posição inicial do motivo atual (evitar sobreposição) e se a posição inicial do motivo atual é menor que a posição final do LCDE mais o gap permitido. Satisfeita estas condições, atribui na posição final do último Bucket a posição final do motivo atual (linha 14 do algoritmo 1), incrementa o número de repetições do último Bucket (linha 15 do algoritmo 1) e em *gap* é somada a distância entre a janela do atual motivo e da posição final do último Bucket (linha 16 do algoritmo 1).

Se não é satisfeita à condição da linha 13, verifica se o número de repetições no último Bucket da posição *hashValue* na tabela hash está dentro limite mínimo (linha 18). Se sim, cria um novo elemento e defina-o como último elemento (linha 19), caso contrário, apaga os dados do último Bucket e atribui os valores do motivo atual a ele (linhas 21 a 23).

Por fim, a tabela *hash* com LCDE conterá as repetições em tandem perfeitas de motivos de tamanho *j*, então salva as repetições da tabela *hash* no arquivo saída (linha 24) e limpa tabela *hash* (linha 25) para ser utilizada pela próxima sequência.

5.3.1.2 Algoritmos de identificação de repetições em tandem imperfeitas

Para identificar repetições imperfeitas, primeiramente, é executado o algoritmo 2, que cria uma tabela de *hash* de degenerações e recebe como parâmetros o arquivo (multi)fasta, o tamanho da janela deslizante e a porcentagem de degenerações. Na tabela *hash* de degenerações criada, cada posição dela é alocada um LCDE auxiliar de degenerações (linha 1 do algoritmo 2), onde é armazenado apenas as degenerações do motivo referente àquela posição.

Algoritmo 2: Cria_tabela_degeneracoes

Entrada: S: sequencia fasta(nucleotídeos ou aminoácidos), J: tamanho da janela deslizante e D: porcentagem de degenerações
Saída: ♯tabelaHashDegeneracao

```
1  ♯tabelaHashDegeneracao = criarTabelaHash ();
2  para ♯cont=1 até tamanho (S)-J faça
3      ♯motivo = extrairJanela (♯cont,♯cont+J);
4      ♯hashValue = calculaHash (♯motivo);
5      se ♯tabelaHashDegeneracao[♯hashValue] == NULL então
6          ♯tabelaHashDegeneracao[♯hashValue] = 0;
7          ♯tabelaHashDegeneracao[♯hashValue]→seq = ♯motivo;
8  para cada ♯x ∈ ♯tabelaHashDegeneracao AND ♯x ≠ NULL faça
9      ♯seq = ♯tabelaHashDegeneracao[♯x]→seq;
10     ♯degeneracoes = NULL;
11     ♯degeneracoes = gerarTodasDegeneracao (D,♯seq);
12     para cada ♯y ∈ ♯degeneracoes faça
13         se ♯tabelaHashDegeneracao[calculaHash (♯y)] == 0 então
14             ♯tabelaHashDegeneracao[calculaHash (♯y)]→Degeneracoes =
               ♯tabelaHashDegeneracao[calculaHash (♯y)]→Degeneracoes ∪ ♯y ;
15 Retorne (♯tabelaHashDegeneracao);
```

Da linha 2 a linha 7 do algoritmo 2, o algoritmo percorre a sequência com uma janela deslizante de tamanho j, extrai o motivo (linha 3 do algoritmo 2), calcula o *hashValue* e marca com valor 1 a posição *hashValue* da tabela *hash* de degenerações, caso seja *NULL* (linha 5 do algoritmo 2).

Da linha 8 a linha 14 do algoritmo 2, este percorre a tabela *hash* de degenerações e para cada posição x marcada com valor zero é gerado todas as possíveis degenerações (linhas 10 e 11 do algoritmo 2) para o motivo da posição x e armazenado no *buffer* de degenerações. Em seguida, para cada elemento do buffer de *degenerações* (degeneração gerada) é calculado o *valor hash* e verificado

se a tabela *hash* de degeneração na posição do valor calculado está definida com valor maior ou igual a zero (linha 13 do algoritmo 2), que significa que o motivo gerado desta posição existe na sequência analisada. Assim, o motivo degenerado gerado é adicionado a LCDE da tabela *hash* de degenerações da posição x (linha 14 do algoritmo 2) (Figura 7). Ao final do algoritmo 2 (linha 15) é retornada a tabela de degenerações.

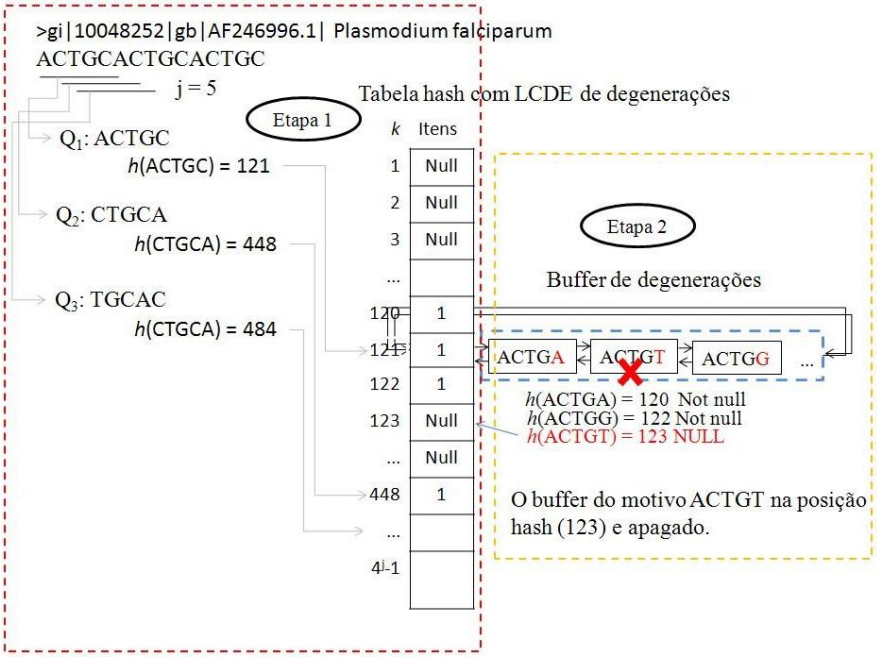


Figura 7 –Criação da tabela hash de degenerações. Na etapa 1, a sequência ACTGCACTGCACTGC é deslizado por uma janela de tamanho cinco, sendo que para cada deslizamento é calculado o valor hash do motivo extraído e na posição do valor calculado é definido valor 1. Em seguida na etapa 2, com o motivo extraído é gerado todas possíveis degenerações até certo limite de variação e para cada degeneração gerada é calculada o valor hash dela e sendo este valor utilizado para verificar se a tabela hash de degeneração nesta posição é diferente de NULL. Caso seja, o motivo degenerado gerado é adicionado a LCDE que contém as degenerações (buffer de degenerações).

Para identificar repetições imperfeitas foi inserida uma chamada ao algoritmo 2 no algoritmo 1, sendo gerado uma tabela de degenerações para cada sequência fasta, representado pela linha 4 no algoritmo 3. Além disso, foi adicionado um trecho de código que verifica se cada motivo extraído da janela deslizante é uma degeneração de algum motivo repetitivo já identificado, linha 26 a linha 32 do algoritmo 3.

Entre as linhas 26 e 32 do algoritmo 3, este verifica se o motivo da atual janela deslizante é uma degeneração de algum motivo já armazenado na tabela *hash*. Para isto, o buffer de degenerações na posição k da tabela *hash* de degenerações é percorrido, sendo que para cada degeneração na posição k da tabela *hash* de degenerações é aplicado na função *h()*. O motivo degenerado é

convertido em um valor inteiro e mapeado na tabela *hash* de elemento repetitivo. Assim, testa se a posição final do último Bucket da LCDE, na posição mapeada da tabela *hash,* é maior que a posição inicial do motivo degenerado (evitar sobreposição) e se a posição inicial do motivo degenerado é menor que a posição final do LCDE mais o *gap* permitido. Satisfeita estas condições, atribui na posição

Algoritmo 3: Identifica_Repetições_PerImPer

Entrada: Q : Arquivo (multi) fasta, J: tamanho da janela deslizante, O : nome do arquivo de saída, G: máximo de gaps permitido, D: porcentagem de degenerações e R: número mínimo de repetições

Saída: O: Arquivo com repetições

```
1  ♯file = lerArquivo (Q);
2  ♯tabelaHash = criarTabelaHash ();
3  para cada sequencia ∈ ♯file faça
4      ♯tabelaHashDegeneracao = Cria_tabela_degeneracoes(S,J,D);
5      para ♯cont=1 até tamanho (sequencia)-J faça
6          ♯motivo = extrairJanela (♯cont,♯cont+J,sequencia);
7          ♯hashValue = calculaHash (♯motivo);
8          se ♯tabelaHash[♯hashValue] == NULL então
9              ♯tabelaHash[♯ hashValue] = 1;
10             criarNoLCDE (♯tabelaHash[♯hashValue],♯cont,♯cont+J,0,♯motivo);
11         senão
12             incremente (♯tabelaHash[♯hashValue]);
13             ♯posFinal = ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoFinal;
14             se ♯posFinal >= ♯cont AND ♯posFinal + G <= ♯cont então
15                 ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoFinal = ♯cont + J;
16                 ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao =+1;
17                 ♯tabelaHash[♯hashValue]→ultimoBucket.gap =+ ♯cont - ♯posFinal;
18             senão
19                 se ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao >= R então
20                     criarNoLCDE (♯tabelaHash[♯hashValue],♯cont,♯cont+J,0,♯motivo);
21                 senão
22                     ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoInicial = ♯cont;
23                     ♯tabelaHash[♯hashValue]→ultimoBucket.PosicaoFinal = ♯cont + J;
24                     ♯tabelaHash[♯hashValue]→ultimoBucket.Nrepeticao = 0 ;
25                     ♯tabelaHash[♯hashValue]→ultimoBucket.gap = 0 ;
26         para cada ♯degeneracao ∈ ♯tabelaHashDegeneracao[♯hashValue] faça
27             ♯hashValueDeg = calculaHash (♯degeneracao);
28             se ♯tabelaHash[♯hashValueDeg] ≠ NULL então
29                 ♯posFinal = ♯tabelaHash[♯hashValueDeg]→ultimoBucket.PosicaoFinal;
30                 se ♯posFinal >= ♯cont AND ♯posFinal + G <= ♯cont então
31                     ♯tabelaHash[♯hashValueDeg]→ultimoBucket.PosicaoFinal = ♯cont + J;
32                     ♯tabelaHash[♯hashValueDeg]→ultimoBucket.Nrepeticao =+ ♯cont -
                       ♯posFinal;
33     salvar (♯tabelaHash,O);
34     limpa (♯tabelaHash);
```

final do último Bucket a posição final do motivo degenerado, incrementa o número de repetições do último Bucket e em *gap* é somada a distância entre a janela do atual motivo degenerado e da posição final do último Bucket. Desta forma é possível identificar repetições em tandem perfeitas e imperfeitas.

Para cada tamanho de motivo desejado é necessário realizar uma chamada do algoritmo 3, isto é, se deseja buscar repetições com motivos de tamanho mínimo 2 e máximo 6 é necessário executar o algoritmo 3 para janela *j* de tamanho 2, 3, 4, 5 e 6. O algoritmo 4 realiza este processo bem como trata as sobreposições de repetições.

Algoritmo 4: ProGeRF

Entrada: Q : Arquivo (multi) fasta, O : nome do arquivo de saída, I : tamanho mínimo de motivo, Y : tamanho máximo de motivo G: máximo de gaps permitido, D: porcentagem de degenerações e R: número mínimo de repetições

Saída: O: Arquivo com repetições

```
1  para ♯janela = I até Y faça em paralelo
2      Identifica_Repetições_PerImPer(Q,arquivo_♯janela,♯janela,G,D,R);
3  para ♯janela = I até Y faça
4      agrupeArquivo (arquivo_♯janela,arquivoFinal);
5  ordenarByPosicaoInicial (arquivoFinal);
6  ordenarBySequenceID (arquivoFinal);
7  para cada ♯sequenceID ∈ arquivoFinal AND unico (♯sequenceID) faça
8      ♯sequencias = retornaTodasSequenciaPorID (♯sequenceID);
9      ♯sequenciaAnterior = ♯sequencias[1];
10     para ♯atual = 2 até tamanho (♯sequencia) faça
11         se ♯sequencia[♯atual].posicaoInicial < ♯sequenciaAnterior.posicaoFinal então
12             ♯sobreposicao = calculeSobreposicao (♯sequencia[♯atual],♯sequenciaAnterior);
13             se ♯sobreposicao > D então
14                 ♯sequenciaAnterior = excluirMenorRep
                   (♯sequencia[♯atual],♯sequenciaAnterior,♯sequencias);
15     salvar (♯sequencias,O);
```

Na linha 1 do algoritmo 4 é realizada uma chamada paralela ao algoritmo 3 para cada um dos tamanhos da janela deslizante. Os resultados de cada chamada são salvos em arquivos separados. Em seguida, os arquivos são agrupados em um único arquivo (linha 3 e 4 do algoritmo 4), as são sequências ordenadas pela posição inicial do motivo repetitivo e pelo o id da sequência (linhas 5 e 6 do algoritmo 4).

O algoritmo 4 entre as linhas 7 e 15 percorre todas as repetições de cada id e verifica se a repetição atual tem a posição inicial menor que posição final da repetição anterior. Sendo satisfeito a condição, então calcule o grau de sobreposição e se o grau de sobreposição estiver dentro permitido, passe para próxima repetição. Se o grau de sobreposição não estiver dentro do permitido, exclua a menor repetição entre a atual e a anterior, definido como atual a maior e passe para próxima repetição. Por fim, salve as repetições restantes no arquivo de saída.

Ao final do algoritmo 4 restará somente repetições que estão dentro do limite de sobreposições permitido.

5.3.1.3 Análise de complexidade

O algoritmo 1 utilizado para identificar repetições em tandem, percorre apenas uma vez a sequência de entrada, e a inserção de motivos na LCDE necessita de apenas uma instrução, devido ser uma LCDE, com isto o tempo de execução deste algoritmo é O(n), onde n é o tamanho da sequência fasta.

O processo de criar a tabela de degenerações, algoritmo 2, percorre uma vez a sequência de entrada para identificar os motivos existentes, marcando com 1 a posição na tabela *hash* de degenerações do motivo encontrado. Este processo requer um tempo de O(n). Logo após percorrer a tabela *hash* de degenerações, para posições marcadas com 1, o algoritmo gera as possíveis degenerações para o respectivo motivo por meio da função gerarTodasDegeneracao().

Como o algoritmo só aceita motivos de no máximo 7 nucleotídeos e 35% de degeneração, só existirá imperfeições de no máximo 2 nucleotídeos. Desta forma, no pior caso, o algoritmo terá que gerar todas as combinações de degenerações de 2 nucleotídeos em um motivo de tamanho sete, que resulta num total de 378 combinações.

A quantidade de combinações para uma degeneração de duas posições para um motivo de tamanho j é dado pela Equação 2.

$$P = s^{v} \prod_{i=0}^{v-1} (j - i)$$ **Equação 2**

Onde s = |N^n|-1 para o conjunto de nucleotídeos ou s = |N^a|-1 para o conjunto de aminoácidos, isto é, N^n = {A,C,T,G} ou N^a ={G,P, A, V, L, I, M, C, F, Y, W, H, K, R, Q, N, E, D, S, T}, respectivamente. Sendo |N| a quantidade de símbolos do conjunto, j o tamanho da janela deslizante e v o número de degenerações.

O produtório da Equação 2, nada mais é que um arranjo simples de j elementos tomados v a v. Desta forma, temos a Equação 3.

$$P = s^{v} \frac{j!}{(j - v)!}$$ **Equação 3**

Para uma janela deslizante de tamanho $j = 7$ em dados genômicos têm-se:

$$s = 4 - 1$$

Que aplicado na Equação 3, resulta em

$$P = 3^2 \frac{7!}{(7 - 2)!}$$

$$P = 9 \frac{7.6.5!}{5!}$$

$$P = 9.7.6 = 378$$

Com isto, cada motivo existente na tabela *hash* poderá ter até 378 degenerações. Como existe no máximo $n - j$ motivos, onde n é tamanho da sequência e j tamanho da janela deslizante, que no pior caso n=7, têm-se a complexidade de O($378(n - 7)$), a qual é da ordem de O(n).

O algoritmo 3 faz com que para cada motivo percorrido pela janela deslizante no algoritmo 1 teste todas as possíveis degenerações daquele motivo na tabela *hash* de degenerações. No pior caso, tanto para nucleotídeo e

aminoácidos, o valor é dado pela Equação 2, podendo dizer que a complexidade é O(Pn). Como P para uma dada janela não muda em função de n, pode-se considerá-lo como constante, logo é desconsiderado assintoticamente, tornando a complexidade O(n).

Vale ressaltar que o tempo de execução da função gerarTodasDegeneracao() está em função do tamanho da janela deslizante e da porcentagem de degenerações. Então o tempo de execução para este procedimento não será linear se a porcentagem de degenerações crescerem muito. No entanto, como estamos trabalhando com pequenos valores de degeneração e janela deslizante este crescimento não será considerado assintoticamente. Desta forma, a complexidade de tempo de execução dos algoritmos de identificação de repetições em tandem é da ordem de O(n).

5.4 INTERFACE

Dado que a ferramenta de identificação de repetições em tandem está disponível nos modos, *stand-alone* e *web*, o usuário pode interagir com ela de duas formas: via linha de comando no *shell* ou via página *web*.

5.4.1 LINHA DE COMANDO

A interação por meio da linha de comando pode ser realizada indicando:

- O endereço do arquivo (multi)fasta contendo as sequências de DNA ou aminoácidos;
- Os tamanhos mínimos e máximos dos motivos;
- O número mínimo de blocos repetitivos para todos os motivos ou para cada um em específico;
- O máximo de *gap* permitido entre motivos de uma repetição;
- A porcentagem de degeneração permitida em um motivo;
- A porcentagem de sobreposições permitida entre repetições;
- O modo de execução, onde é definido se será executado para sequência de DNA ou de aminoácidos; e
- O nome do arquivo de saída;

O comando para executar o *script* definindo tais informações é como segue:

```
perl progerf.pl -q [FILE] -o [STRING] -i [INTERGER] -y [INTERGER] -r [INTERGER]
-g [INTERGER] -v [INTERGER] -d [INTERGER]
```

Onde:

```
-q [FILE]

Arquivo de entrada como sequências DNA/aminoácidos no formato (multi)fasta.

-o [STRING]

Nome do arquivo de saída. Nome padrão results;

-i [INTERGER]

Tamanho mínimo do motivo. Valor padrão 2;

-y [INTEGER]

Tamanho máximo do motivo. Valor padrão 5;

-r [INTERGER] ou -rl [INTERGER-...-INTERGER]

Valor mínimo de blocos repetitivos do motivo. Valor padrão -r 5;

   -g [INTERGER]

Gap máximo permitido entre motivos de uma repetição em tandem. Valor padrão 0;

-v [INTERGER]

Máximo da sobreposição permitido.  Valor padrão 0;

-d [INTERGER]

Máximo de degeneração permitida. Valor padrão 0;

-m [STRING]

Modo de execução. n para nucleotídeos ou p para proteínas. Valor padrão n.
```

Por exemplo, a sequência de comando:

```
#perl progerf.pl -q Linfantum_JPCM5.fasta -o output -i 2 -y 6 -r 5 -g 3 -v 0 -d
20 -m n
```

Realiza uma busca por repetições em tandem no arquivo `Linfantum_JPCM4.fasta`, onde os tamanhos dos motivos variam de 2 a 6, o número mínimo de blocos repetitivos para todos os tamanhos de motivos é 5, o máximo de *gaps* aceito entre motivos é 3, aceita motivos com até 20% de degeneração, não aceita sobreposições e o resultado será salvo no arquivo `output`.

5.4.2 INTERFACE WEB

A interface *web* (Figura 8) apresenta a mesma flexibilidade do modo em linha de comando. Todavia, diferentemente do modo *stand-alone*, o modo web é independente de plataforma, pode ser executado em qualquer *browser*, a definição do parâmetro é realizada por meio de formulários, com botões, caixas de texto e *combo box*.

A interface *web* disponibiliza três formas de envio do arquivo fasta, sendo elas:

1. Upload de arquivo: o usuário pode enviar um arquivo fasta de seu próprio computador;
2. Colar a sequência: o usuário copia uma sequência de interesse e cola na caixa de texto;
3. *Download* automático da base de dados do Centro Nacional de Biotecnologia (NCBI do inglês *National Center for Biotechnology*): o usuário digita um ou uma lista de *GI numbers*[1] separado por vírgulas, e a ferramenta irá busca na base do NCBI as sequências, realizar o *download* e executar o algoritmo de extração de repetições.

[1] O *GI number* (*genInfo identifier*) é um número único no qual identifica uma sequência de DNA ou proteína em particular no NCBI

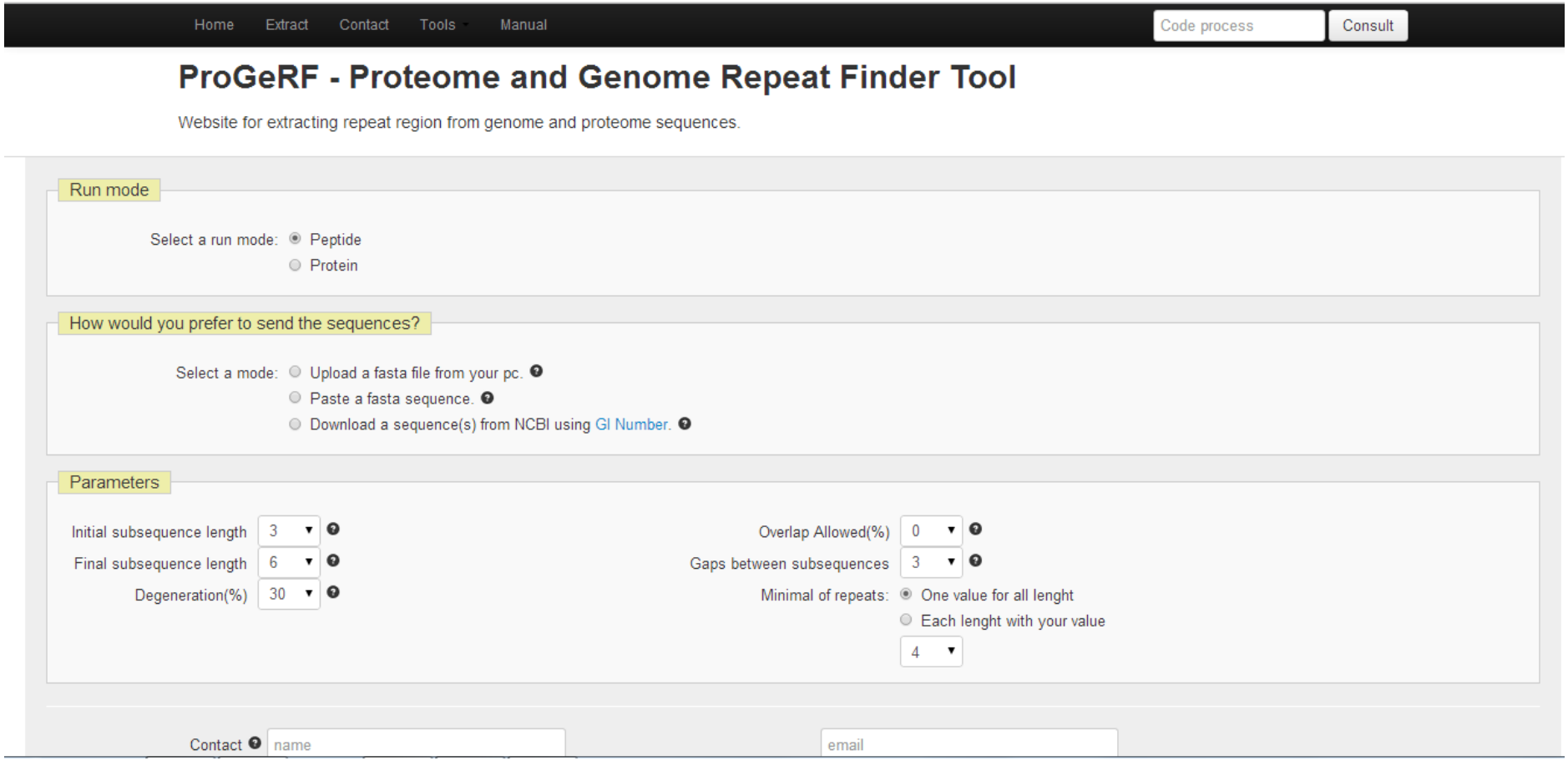


Figura 8 - Screenshot da tela inicial da ferramenta web. O usuário tem que definir o modo de execução da ferramenta (nucleotídeo ou proteína), selecionar a forma de envio das sequências (multi)fasta, que pode ser via *upload* do arquivo fasta, copiar e colar sequência em uma caixa de texto e/ou digitando o *GI Number* das sequências desejadas separadas por vírgula. Por fim, deve ser definido os demais parâmetros da ferramenta. Além disto, o usuário pode deixar o nome e e-mail para receber o código do processo quando ele for finalizado.

.

A cada execução da ferramenta *web* é gerado um código de identificação, exclusivo do processo, o qual é apresentado na tela quando o usuário clica em "*send*".

Este código serve para que, em outro momento, o usuário possa rever os resultados daquele processo, sem necessidade de reexecutar a ferramenta. Se o usuário preencher o campo de contato, ao final da execução do algoritmo de identificação é enviado um e-mail com o código do processo e um link para acessar o resultado.

No campo, *code process*, no canto superior direito da página *web* (Figura 8), o usuário pode digitar o código de identificação do processo que lhe será apresentado uma página (Figura 9), contendo:

- Os parâmetros utilizados na execução da ferramenta;
- A data e o horário de execução;
- Um botão para realizar o *download*;
- O status do processo;
- Botões para realizar o *download* do arquivo fasta utilizado bem como do arquivo de resultado e um botão para visualizar o resultado na página *web*;

A visualização dos resultados na página *web* pode ser feita de duas formas, por meio de uma tabela utilizando o script jqGrid e na forma gráfica, por meio do *plug-in* JBrowse (Skinner et al. 2009).

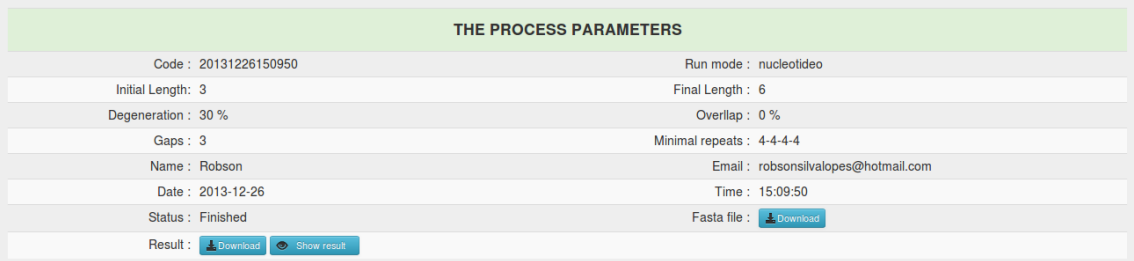


Figura 9 - *Screenshot* da página de resultados, onde pode ser visualizado os parâmetros do processo, baixar o arquivo com a sequência fasta submetida e baixar o arquivos com a repetições identificada.

jqGrid é um script baseado em jQuery e Ajax para representar e manipular dados tabulares na *web* de forma dinâmica (Tomov 2008). Utilizando o jqGrid o usuário pode fazer consultas por algum padrão específico de motivo, definindo diversos filtros de consulta e ordenar os resultados, por qualquer uma das colunas. A Figura 10, mostra como os resultados são apresentados na página *web* utilizando o jqGrid.

Outra forma de visualizar os resultados é por meio do JBrowse. O JBrowse é um browser para visualização de genomas desenvolvido em JavaScript, onde o usuário pode navegar pelas anotações do genoma na *web*. Com ele, pode realizar zoom, navegar e selecionar faixa de subsequências dentro de um genoma (Skinner et al. 2009).

A Figura 11 traz a visualização dos resultados utilizando o JBrowse, com o mapeamento das repetições ao longo da sequência proteômica. Na coluna maior do lado direito da parte superior ficam os botões de navegação como: aumentar e diminuir zoom e ir para o lado direito ou esquerdo da sequência. Logo mais abaixo destes botões, está o proteoma ou genoma.

No exemplo da Figura 11 foi identificado a repetição de aminoácidos VDETVDETVDETVDETVDET na proteína *merozoite surface protein* 9 (GI 525346946) do organismo *Plasmodium vivax*. Este elemento repetitivo contém o motivo VDET repetido cinco vezes.

Na parte central, representado por linhas verdes, estão as repetições localizadas no genoma/proteoma. Clicando na linha verde é apresentada uma tela com informações mais detalhadas da repetição (Figura 12), tais como: motivo; posição no arquivo fasta enviado; tamanho em pares de

base; quantidade de repetições; *gaps;* e estatística da quantidade de aminoácidos repetitivos no motivo VDETVDETVDETVDETVDET.

Um manual da ferramenta está disponível no link http://64.79.105.19/ligp/?pg=manual.

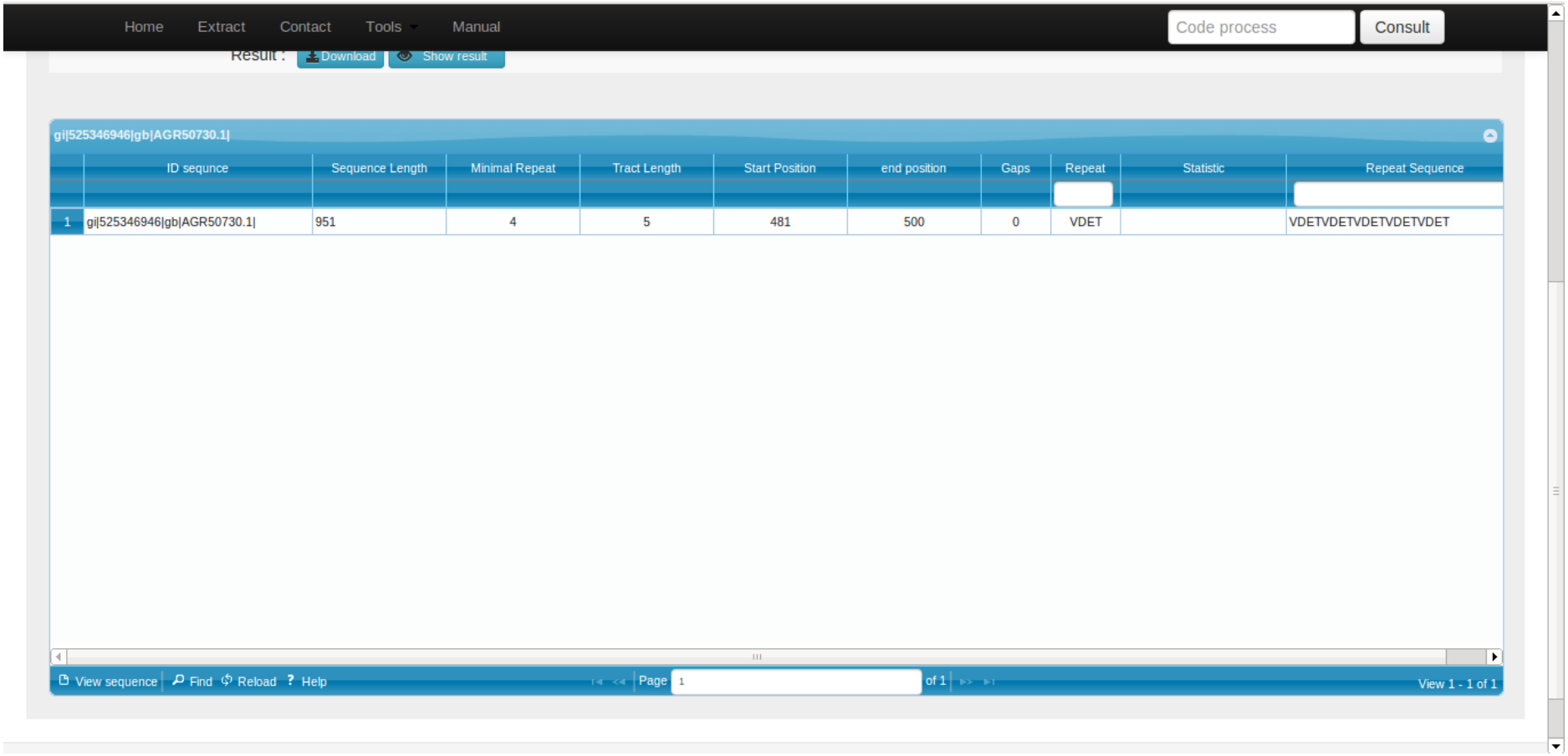


Figura 10–Screenshot do *Datagrid* de resultados utilizando o *script* jqGrid. A primeira coluna contém o identificador da sequência, a segunda o tamanho da sequência da proteína, a seguinte o mínimo de repetições permitidos, a próxima a quantidade de repetições encontrada, a quinta e a sexta colunas representam as posições iniciais e finais do microssatélite, respectivamente. A penúltima coluna contém a quantidade *gaps* e a última o motivo repetitivo

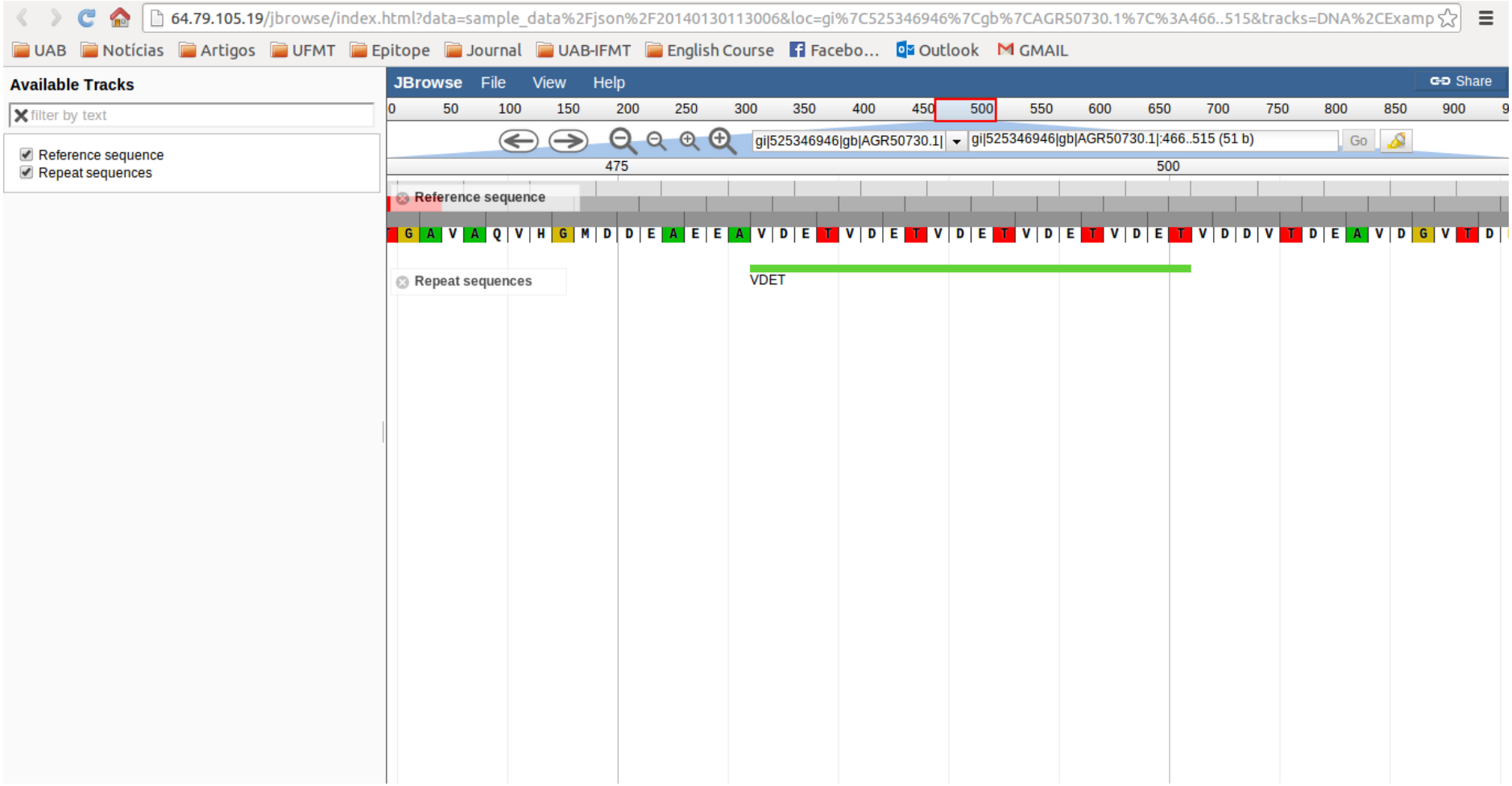


Figura 11 - *Screenshot* do JBrowse. A linha em verde representa a posição da repetição em tandem dentro do genoma/proteoma. As letras logo abaixo da linha verde especificam o motivo repetitivo. Para visualizar mais informações, basta clicar na linha verde.

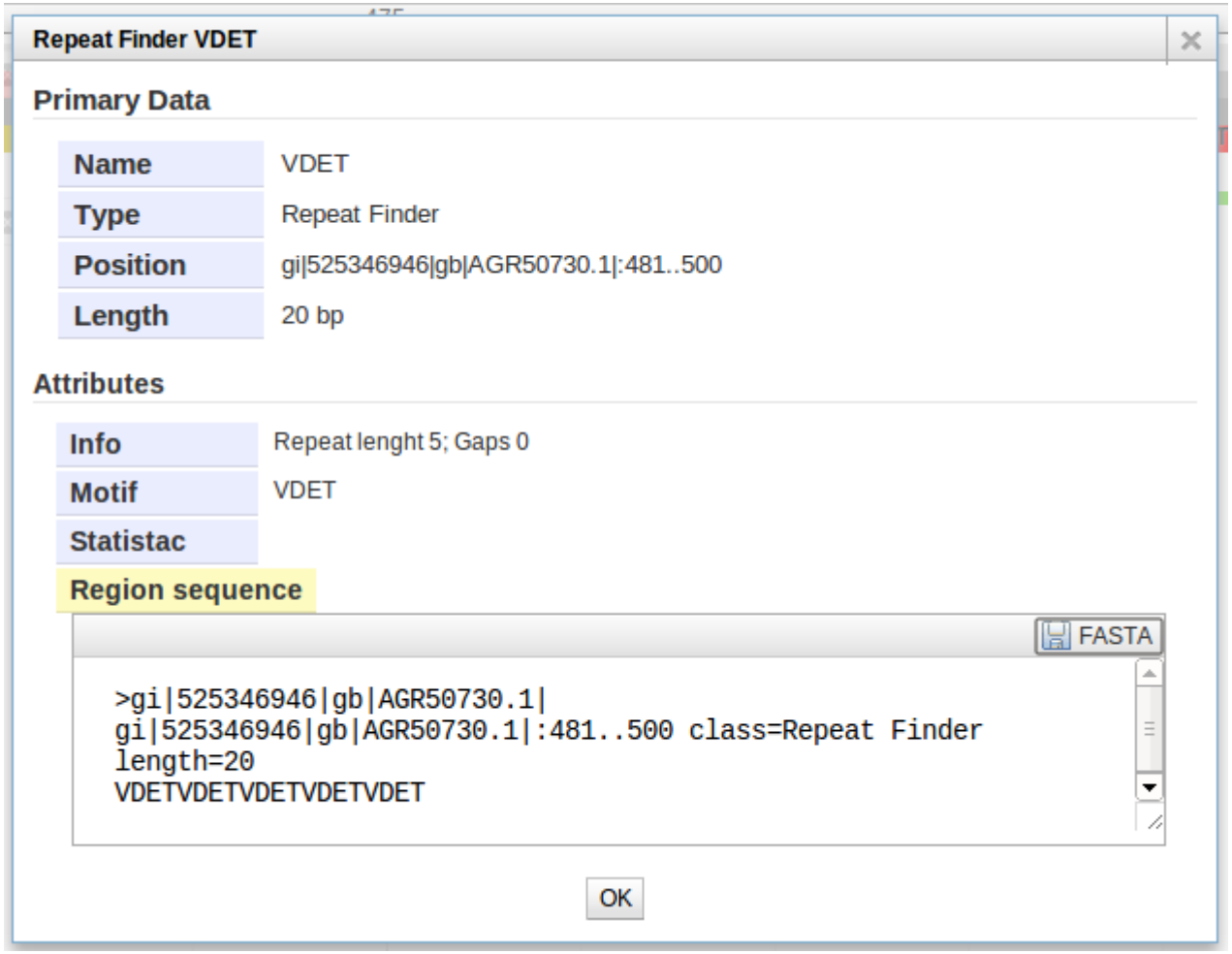


Figura 12 - Tela com informações da repetição.

5.5 COMPARAÇÃO COM OUTRAS FERRAMENTAS

Apresentamos dois experimentos para validação da ferramenta de identificação de repetições desenvolvida neste trabalho. No primeiro experimento é demonstrada a eficiência do ProGeRF comparado com outras ferramentas de identificação de microssatélites, e o segundo experimento demonstra o uso do algoritmo em arquivos FASTA de proteínas na busca de elementos repetitivos.

Os testes foram realizados em computador DELL Inspiron, processador Intel core 2 duo 2.2 GHz, com 2 MB de cache, 3 GB RAM, 320 GB de HD e sistema operacional Ubuntu 14.04 LTS 32 bits.

Para o primeiro experimento foram utilizadas as ferramentas MISA (Thiel et al. 2003), Mreps (Kolpakov et al. 2003), GMATo (Wang et al. 2013), SciRoKo (Kofler et al. 2007), Sputnik (Rota et al. 2005), TRF (Benbson 1999). Cada ferramenta foi executada em cada uma das seguintes sequências genômicas: *Plasmodium falciparum* chromosome IV (NC 004318.1), *Saccharomyces cerevisiae* chromosome IV (NC 001136.8), *Mycobacterium tuberculosis* H37Rv genome (NC 000962.2) utilizados no trabalho de Mudunuri e Nagarajaram (2007) baixado de ftp://ftp.ncbi.nih.gov/genomes, e o genoma completo da *Setaria italica* utilizado no trabalho de Wang et al. (2013), baixado de phytozome http://www.phytozome.net/.

Para as ferramentas que permitem configuração dos parâmetros tamanho máximo e mínimo do motivo, número mínimo de repetições, os valores definidos foram 1, 6 e 5 respectivamente. Para os demais parâmetros, os seguintes valores foram utilizados de acordo com a ferramenta: a) MISA: diferença máxima entre 2 SSR de 0; b) Mreps: a resolução de 5; c) SciRoKo: modo mismatched fixed penalty, demais parâmetros com valores default; d) Sputnik: tamanho máximo de 5 (máximo permitido pela ferramenta), score mínimo de 5, recursão máxima de 0; tamanho mínimo de SSR reportada de 10 e pontos por mismatch e match de 1; e) TRF: peso do match: 2, mismatch penalty: 7, indel penalty: 7, probabilidade de match de 80, indel probability de 10, score mínimo de 2, período máximo15, e f) ProGeRF: máximo de gap permitido de 1, zero de sobreposição, uma degeneração de 20%, modo nucleotídeo.

5.5.1.1 Teste de desempenho de tempo e número de repetições em arquivos fasta de DNA

A ferramenta IMEX apresentou erro durante a execução das versões 1.0 e 2.0 no sistema operacional Ubuntu 14.04 de 32 bits. Desta forma, não foi possível comparar os resultados desta ferramenta. Nas três primeiras sequências (Tabela 2), ProGeRF foi mais lento que SciRoKo, Sputnik, MISA e Mreps. No entanto, o tempo de ProGeRF pode ser considerado bom se considerarmos que ela encontrou um número maior de repetições que as outras ferramentas.

Tabela 2 - Comparação da quantidade de detecção e tempo de execução (em segundos) do Mreps, Sputnik, GMATo, SciRoKo, TRF e ProGeRF.

Sequência	Mreps R(T)	MISA R(T)	Sputnik R(T)	GMATo R(T)	SciRoKo R(T)	TRF R(T)	ProGeRF R(T)
NC_004318.1 (1204 Kb)	9608 (2.8)	22867 (3.2)	7420 (0.7)	23539 (10.3)	3763 (1.1)	30244 (72.4)	26164 (3.9)
NC_001136.8 (1531 Kb)	935 (1.4)	10640 (3.3)	1427 (0.9)	10721 (7.7)	185 (0.7)	8101 (4.5)	11552 (2.4)
NC_000962.2 (4411 Kb)	1412 (3.9)	6832 (8.9)	3140 (1.46)	6846 (12.1)	72 (1.5)	19496 (24.5)	11422 (4.0)
*Setaria** *(*515 Mb)	-(-)	2054241 (868.3)	480644 (105.7)	2073643 (9859.1)	47770 (129.0)	2438036 (1481.5)	2319812 (1352.0)

R significa repetições e T tempo em segundos.

5.5.1.2 Comparação da localização física no genoma dos resultados das ferramentas

Avaliamos também se as detecções retornadas pelas ferramentas nas sequências NC_004318.1, NC_001136.8 e NC_000962.2 (Tabela 1) ocorreram na mesma localização física no genoma. Percebeu-se que mais de 75% das regiões contendo microssatélites identificados pelas ferramentas SciRoKo, Sputnik e Mreps também foram identificadas pela ferramenta ProGeRF nas três sequências. Sendo que para as ferramentas GMATo e MISA, a ferramenta ProGeRF identificou 100% dos microssatélites localizados por essas ferramentas (Tabela 3).

Tabela 3 - Loci e nucleotídeos cobertos entre as ferramentas.

Sequência		Ferramenta	Mreps	MISA	B Sputnik	GMATo	SciRoKo	TRF	ProGeRF
Plasmodium Chr4 NC_004318.1	A	Mreps	--	78(45)	53(33)	78(45)	41(36)	98(74)	89(60)
		MISA	47(63)	--	21(4)	100(99)	18(40)	88(79)	89(98)
		Sputnik	91(86)	70(74)	--	70(74)	58(69)	0(0)	83(84)
		GMATo	49(62)	100(99)	21(40)	--	19(40)	89(79)	100(98)
		SciRoKo	96(95)	93(76)	95(71)	92(76)	--	95(96)	98(91)
		TRF	51(41)	54(32)	0(0)	54(32)	16(21)	--	68(46)
		ProGeRF	56(56)	86(67)	21(31)	86(67)	16(32)	87(78)	--
SAC Chr4 NC_001136.8	A	Mreps	--	60(40)	42(26)	68(40)	18(20)	95(74)	86(62)
		MISA	7(12)	--	3(7)	100(99)	1(4)	33(37)	100(99)
		Sputnik	30(35)	29(30)	--	29(30)	12(1)	74(73)	38(39)
		GMATo	7(12)	100(99)	3(7)	--	1(4)	33(37)	100(99)
		SciRoKo	91(89)	77(61)	86(59)	77(61)	--	99(99)	94(72)
		TRF	15(16)	41(26)	13(12)	41(26)	2(5)	--	48(35)
		ProGeRF	8(16)	92(80)	4(7)	92(80)	1(5)	34(39)	--
MTB H37Rv NC_000962.2	A	Mreps	--	9(3)	15(7)	9(3)	3(3)	91(71)	75(58)
		MISA	2(3)	--	1(1)	100(99)	0.5(1)	13(13)	100(99)
		Sputnik	6(7)	2(2)	--	2(2)	1(1)	66(64)	14(14)
		GMATo	2(3)	100(99)	1(1)	--	0.5(1)	13(13)	100(99)
		SciRoKo	73(74)	47(36)	63(41)	47(36)	--	100(100)	79(72)
		TRF	8(7)	4(1)	10(7)	4(1)	0.4(0.5)	--	18(13)
		ProGeRF	9(16)	60(35)	4(4)	60(35)	0.5(1)	29(33)	--

Porcentagem do número total de detecções (repetições perfeitas e imperfeitas) da ferramenta A também detectado (coberto) pela ferramenta B. O valor entre parênteses é a proporção de nucleotídeos detectados por A e coberto por B.

5.5.1.3 Teste de desempenho de tempo e número de repetições em arquivos fasta de proteínas

Uma vez que as principais ferramentas para identificação de microssatélites, listadas acima, identificam elementos repetitivos apenas em dados genômicos, apresentamos no segundo experimento, um teste da ferramenta ProGeRF executando em dados proteômicos. Para este segundo experimento, executamos a ferramenta ProGeRF na versão *web* no modo *protein* nas proteínas: *circumsporozoite protein* (ACO49545.1), *merozoite surface protein* 1 (XP 001352170.1) e *merozoite surface protein* 9 (AAN36363.1).

Tabela 4 - Elementos repetitivos de proteínas encontrados pela ferramenta web ProGeRF.

ID	Lócus	Motivo	Repetição
XP_0013522170.1	62-97	GASAQS	6
ACO49545.1	146-297	PNAN	38
AAN36363.1	693-712	KEKEE	4

Os parâmetros utilizados foram motivos de tamanho entre 2 e 6, repetições de pelo menos 4 motivos, zero para gap, sobreposição e degeneração.

A Tabela 4 apresenta os resultados da execução da versão *web* do ProGeRF, nas proteínas *circumsporozoite protein* (ACO49545.1), *merozoite surface protein* 1 (XP 001352170.1) e *merozoite surface protein* 9 (AAN36363.1), no qual o elemento repetitivo PNAN (PRO-ASN-ALA-ASN) foi identificado na proteína ACO49545.1. Nas outras proteínas, elementos repetitivos foram identificados com uma baixa frequência de repetição. A Figura 13 apresenta os resultados que são disponibilizados para o usuário no ambiente *web*: a) Visualização do resultados através do *plug-in* jqGrid. Clicando sobre o elemento repetitivos é aberta uma nova janela gráfica; b) Nesta nova janela, os elementos repetitivos são mapeados e dispostos graficamente através do JBrowse.

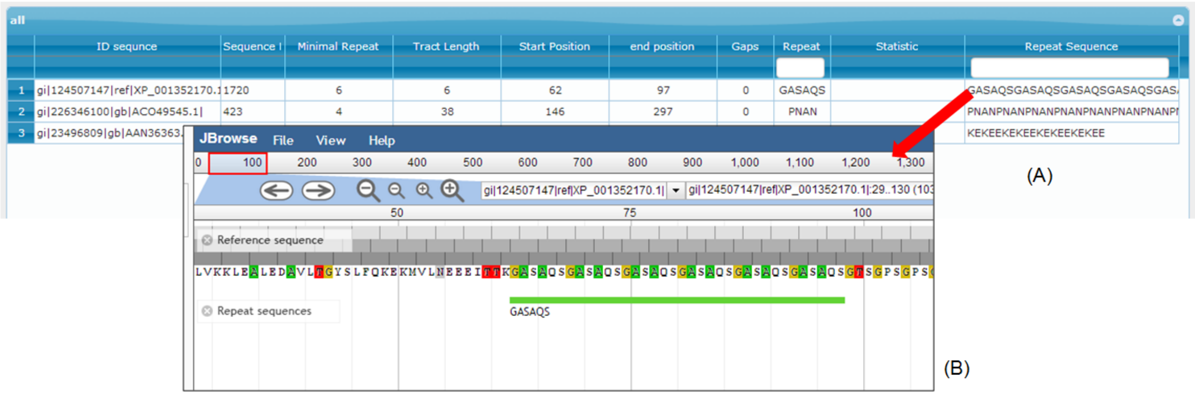


Figura 13 – Screenshot do resultado da ferramenta web ProGeRF nas proteínas *circumsporozoite protein* (ACO49545.1), *merozoite surface protein* 1 (XP 001352170.1) e *merozoite surface protein* 9 (AAN36363.1): a) Visualização dos resultados através do plug-in jqGrid. Clicando sobre o elemento repetitivo é aberta uma janela com visualização gráfica; b) Os elementos repetitivos são mapeados e apresentados graficamente através do JBrowse.

6 PREDIÇÃO *IN SILICO* DE EPÍTOPOS LINEARES DE CÉLULAS B

6.1 MÉTODOS

6.1.1 BASE DE DADOS

6.1.1.1 Treinamento

O treinamento é parte importante quando se utiliza aprendizado de máquina. Os algoritmos de aprendizado de máquina extraem a representação do conhecimento a partir de dados exemplos, para que a partir da representação gerada sejam capazes de produzir saídas corretas para novas entradas não apresentadas previamente (Lorena & Carvalho 2007).

Para realização dos experimentos foram extraídos epítopos positivos e epítopos negativos lineares de células B não redundantes de bactérias, vírus e protozoários da base de dados do IEDB (Zhang et al. 2008, Vita et al. 2010), que foram utilizados no processo de treinamento. O IEDB é uma base de dados de epítopos constantemente atualizada e apresenta uma grande quantidade epítopos validados experimentalmente.

São considerados epítopos positivos motivos que foram validados experimentalmente e apresentaram ensaio positivo em pelo menos um experimento, e os epítopos negativos são motivos testados experimentalmente que não apresentou ensaio positivo em nenhum experimento. Não-epítopos são sequências de aminoácidos que não foram testadas experimentalmente, ou seja, aminoácidos que não pertencem ao conjunto de epítopos positivos e nem a epítopos negativos, mas pertencentes as mesmas proteínas que apresentam epítopos positivos.

A extração dos epítopos positivos e negativos do IEDB foi realizada em 02 de dezembro de 2014, sendo que apenas epítopos de agentes causadores de doenças infecciosas em seres humanos foram selecionados e, também, foram excluídos epítopos de células T (Tabela 5).

Sequências de epítopos que apresentaram similaridade maior ou igual a 80% (Gao et al. 2012), avaliadas pelo BLAST (Camacho et al. 2013), foram agrupadas, sendo que apenas um deles foi selecionado aleatoriamente e mantido como uma sequência de epítopo no conjunto de dados final. Por fim, os arquivos com

epítopos foram submetidos a um script *in house* de controle de qualidade de dados proteômicos que retira as sequências que apresentam letras B, J, O, X, U, Z e símbolos como * e #, que não representam aminoácidos.

Tabela 5 - Filtros de consultas utilizados na extração de epítopos de bactérias, vírus e protozoários no IEDB (Zhang et al. 2008).

Taxo	Filtro
Bactérias	Disease Name is bacterial infection [DOID:104] Host disease state/status is Selected T Cell is excluded MHC Ligand is excluded Host Organism is Homo sapiens (human)
Protozoários	Disease Name is visceral leishmaniasis [DOID:9146] or Disease Name is Chagas' disease [DOID:12140] or Disease Name is Chagas cardiomyopathy [DOID:12569] or Disease Name is Cutaneous leishmaniasis [DOID:9111] or Disease Name is amebiasis [DOID:9181] or Disease Name is giardiasis [DOID:10718] or Disease Name is leishmaniasis [DOID:9065] or Disease Name is mucocutaneous leishmaniasis [DOID:9155] or Disease Name is Plasmodium falciparum malaria [DOID:14067] or Disease Name is Plasmodium vivax malaria [DOID:12978] or Disease Name is toxoplasmosis [DOID:9965] or Disease Name is trichomoniasis [DOID:1947] or Disease Name is trypanosomiasis [DOID:10113] or Disease Name is cryptosporidiosis[DOID:1733] Host disease state/status is Selected T Cell is excluded MHC Ligand is excluded Host Organism is Homo sapiens (human)
Vírus	Disease Name is viral infection [DOID:934] Host disease state/status is Selected T Cell is excluded MHC Ligand is excluded Host Organism is Homo sapiens (human)

Com isto, foram criadas duas bases de dados para cada táxon, uma com epítopos positivos e outra como epítopos negativos, num total de seis arquivos, com 5.548 epítopos positivos e 5882 epítopos negativos (Tabela 6).

Tabela 6 - Quantidade epítopos e não-epítopos de cada táxon extraídos do IEDB.

Táxon	Epítopos Positivos*	Epítopos negativos**
Bactéria	1083(19,52%)	1949(33,13%)
Vírus	3669(66,13%)	3513(59,72%)
Protozoário	796 (14,34%)	420(7,14%)
Total	5548	5882

*Em pelo menos um teste experimental apresentou resultado positivo;
** Não apresentou resultado positivo em qualquer teste experimental realizado.

O treinamento da máquina de aprendizado foi realizado em oito bases de dados diferentes, as quais variavam na composição de epítopos positivos, epítopos negativos e não-epítopos de cada táxon. Sendo elas:

1. **DB_all_prop:** Composta por epítopos positivos e negativos de todos os táxons, sendo 1/3 de resíduos de bactérias (epítopos positivos e negativos), 1/3 de resíduos de vírus (epítopos positivos e negativos) e 1/3 de resíduos de protozoários (epítopos positivos e negativos);
2. **DB_all_propN:** Composta por epítopos positivos e não-epítopos de todos os táxons, sendo 1/3 de resíduos de bactérias (epítopos positivos e não-epítopos), 1/3 de resíduos de vírus (epítopos positivos e não-epítopos) e 1/3 de resíduos de protozoários (epítopos positivos e não-epítopos);
3. **DB_prot:** Composta apenas por resíduos de epítopos positivos e negativos de protozoários;
4. **DB_protN:** Composta apenas por resíduos de epítopos positivos e não-epítopos de protozoários;
5. **DB_bac:** Composta apenas por resíduos de epítopos positivos e negativos de bactérias;
6. **DB_bacN:** Composta apenas por resíduos de epítopos positivos e não-epítopos de bactérias;
7. **DB_vir:** Composta apenas por resíduos de epítopos positivos e negativos de vírus;
8. **DB_virN:** Composta apenas por epítopos positivos e não-epítopos de vírus;

As bases de dados de treinamento sempre foram compostas por 50% de epítopos positivos e 50% de epítopos negativos ou 50% de epítopos positivos e 50% de não-epítopos.

Uma vez que a quantidade de epítopos positivos, epítopos negativos e não-epítopos de cada táxon eram de valores diferentes, foi necessário deixá-las em quantidade proporcionais. Assim, para as duas primeiras bases de dados, foi identificado a base de epítopo positivo e epítopo negativo/não-epítopo por táxon de menor tamanho (neste caso sempre foi a base de dados de epítopos positivos de protozoários) e em seguida convertia as bases maiores que ela para mesmo tamanho da menor, sendo que a conversão foi realizada por meio de seleção aleatória.

Todos os epítopos positivos não redundantes de protozoário do IEDB foram mantidos nas bases DB_all_prop e DB_all_propN. Todavia, para testar a capacidade de generalização da ferramenta, isto é, medir a eficiência na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento, foi criada uma nona base de dados chamada DB_all_propN50 que contém apenas 50% dos epítopos positivos não redundantes de protozoários extraídos do IEDB. Desta forma, DB_all_propN50 é composta por metade dos epítopos positivos e não-epítopos de bactérias, vírus e protozoários da base DB_all_propN.

6.1.1.2 Teste

Na literatura diversas bases de dados de epítopos foram criadas para testar ferramentas de aprendizado, como Sollner dataset (Sollner et al. 2008), AntiJen1 e AntiJen2 dataset (Toseland et al. 2005), PC dataset (Wang et al. 2011) e Pellequer dataset (Larsen et al. 2006). No entanto, a presença de dados de protozoários na grande maioria destas bases não é maior que 9% (Tabela 7).

Tabela 7 - Porcentagem de cada táxon presente nas bases de dados utilizadas em treinamento e teste de ferramentas de predição de epítopos.

Taxo	Sollner dataset (Sollner et al. 2008)	AntiJen 1 dataset (Toseland et al. 2005)	AntiJen 2 dataset (Toseland et al. 2005)	PC dataset (Wang et al. 2011)	Pellequer dataset (Larsen et al. 2006)
Bactéria	49,12%	26,00%	27,00%	18,18%	14,28%
Vírus	42,10%	39,00%	31,00%	45,45%	14,28%
Protozoários*	8,77%	9,00%	2,50%	0,00%	0,00%
Outros	0,00%	26,00%	39,50%	36,36%	71,42%

* Filos Sarcomastigophora (ordem Kinetoplastida) e Apicomplexa

Diante disto, para testar a capacidade de aprendizado dos modelos, foram utilizadas três bases de dados, sendo:

1. Protozoa_epitope: arquivo fasta contendo proteínas de protozoários com epítopos experimentalmente validados, extraídos do IEDB;
2. Sollner dataset: esta base de dados foi criada por Sollner et at. (2008) e contém 57 sequências de proteínas não redundantes de patógenos extraídas da base de dados do IEDB. A base é composta de 2.317 resíduos de epítopos positivos (5.04%) e 43.690 resíduos de não-epítopos (94.96%) (Lin et al. 2013).
3. AntiJen 1: Esta base foi extraída da base de dados AntiJen, proposta por Larsen et al (2006), contém 124 sequências de proteínas sendo 5.529 resíduos de epítopos positivos (8.34%) e 60.800 resíduos de não-epítopos(91.66%) (Lin et al. 2013).

As bases de dados Sollner e AntiJen 1 foram selecionadas por conterem um maior número de proteínas de protozoários.

6.1.2 ESTRATÉGIA DE PREDIÇÃO

A estratégia de predição implementada neste trabalho, chamada de SBP (SVM BeeproPipe), foi inspirada na ferramenta BEEPro (*B-cell epitope prediction by evolutionary information and propensity scales*) (Lin et al. 2013). Esta ferramenta foi escolhida, pois apresentou bons resultados em testes realizados na base de dados Sollner, com área sobre a curva de (AUC) = 0.9987, acurácia = 99.29%, coeficiente de correlação de Matthews (MCC, do inglês *Matthews correlation coefficient*) = 0.9281, sensitividade = 0.9604, especificidade= 0.9946 e valor preditivo positivo (PPV) = 0.9042.

O BEEPro aplica a técnica de aprendizado de máquina máquina de vetores de suporte (SVM do inglês *Support Vector Machine*) em 16 propriedades para predizer tanto epítopos lineares como conformacionais de células B. Fazem parte destas propriedades: a) Matrizes de pontuação de posição específica (PSSM, do inglês *position-specific scoring matrix*), b) uma escala de proporção de aminoácidos (AAR, do inglês *Amino acid ratio*) e d) um conjunto de 14 propriedades físico-químicas obtidos da base de dados do AAindex (Kawashima et al. 2008).

Todavia, a estratégia utilizada pelo BEEPro em sua totalidade acaba sendo inviável de ser aplicada no desenvolvimento de um pipeline de predição de

epítopos, uma vez que o cálculo da PSSM é muito demorado para consultas com proteomas muito grandes. Por esta razão a ferramenta não está disponível para *download* ou como um serviço *web*. Diante disto, neste trabalho foram utilizadas apenas 15 propriedades, não sendo utilizada a propriedade PSSM.

6.1.2.1 Taxa de proporção de aminoácido

A taxa de proporção de aminoácido é utilizada nesta técnica de predição, pois diversos trabalhos (EL-Manzalawy et al. 2008, Wee et al. 2010a, Wang et al. 2011, Yao et al. 2012) relatam que determinados aminoácidos (di ou tri-peptídeos) ocorrem com maior ou menor frequência em epítopos. Análise realizada por Wee et. al. (2010) revelaram que os aminoácidos triptofano (W), prolina (P), glutamina (N) apresentam maior frequência em epítopos positivos, enquanto fenilalanina (F) e leucina (L) são encontrados com menor frequência em epítopos positivos. No mesmo sentido, Yao et al. (2012) ao analisar a frequência de tri-peptídeos relata que os aminoácidos glutamina e prolina podem desempenhar um papel importante no reconhecimento de epítopos por anticorpos de células B, uma vez que estes aminoácidos ocorrem com maior frequência nos tri-peptídeos.

Para determinar a taxa de proporção de aminoácidos, é calculada a frequência de cada um dos 20 aminoácidos na base de dados, de acordo com a Equação.

$$p(\alpha_i) = \frac{f(\alpha_i^+)/\sum_i f(\alpha_i^+)}{f(\alpha_i^-)/\sum_i f(\alpha_i^-)} \qquad \textbf{Equação 4}$$

Onde $f(\alpha_i^+)$ e $f(\alpha_i^-)$ representa a frequência de ocorrência do aminoácido α_i em epítopos positivos e epítopos negativos/não-epítopos na sequência de peptídeos, respectivamente. Para impedir a dominância da frequência de alguns aminoácidos, estes foram normalizados entre [-1,1], segundo a Equação 5

$$\rho(\alpha_i) = 2\left(\frac{p(\alpha_i) - \min\{p(\alpha_i)\}}{\max\{p(\alpha_i)\} - \min\{p(\alpha_i)\}}\right) - 1 \qquad \textbf{Equação 5}$$

Onde $\max\{p(\alpha_i)\}$ e $\min\{p(\alpha_i)\}$ retornam o maior e o menor valor de frequência entre os aminoácidos, respectivamente. Maiores detalhes em Lin et al. (2013 p. 3).

6.1.2.2 Propriedades físico-químicas

Propriedades físico-químicas como escala de superfície de acessibilidade (Emini et al. 1985), estrutura de folhas beta (Chou & Fasman 1978), flexibilidade (Karplus & Schulz 1985), antigenicidade (Kolaskar & Tongaonkar 1990) e hidrofilicidade (Parker et al. 1986) foram os principais dados observados nos aminoácidos com a finalidade de tentar identificar características ou padrões que pudessem caracterizar um epítopo.

No entanto, Blythe e Flower (2005) após testarem 484 propriedades físico-químicas com o objetivo de predizer epítopos de células B relataram que os resultados desta predição utilizando apenas estas propriedades são, marginalmente, superior ao aleatório. Por fim, relatam que melhores resultados podem ser alcançados utilizando propriedades físico-químicas em combinação com técnicas de inteligência artificial ou aprendizado de máquina.

Lin et al. (2013) testaram a combinação de 16 índices de propriedades físico-químicas e bioquímicas, extraídas do AAindex[2] (Kawashima & Kanehisa 2000, Kawashima et al. 2008), com SVM e notaram que 14 apresentaram bons índices de acurácia na predição de epítopos lineares de células B. Sendo elas: hidrofilicidade (PARJ860101)[3], hidrofobicidade (PONP930101), flexibilidade (KARP850102 e BHAR880101), interatividade (BASU050101), composição (GRAR740101), volume (GRAR740103), transferência de carga (CHAM830107), capacidade de doar transferência de carga (CHAM830108), capacidade de doar ligações de hidrogênio (FAUJ880109), frequência da estrutura alfa-hélice (NAGK730101), frequência da estrutura beta (NAGK730102), frequência da estrutura coil (NAGK730103) e antigenicidade (ANTIGEROB). Desta forma, estas propriedades também foram utilizadas na implementação do presente trabalho.

[2] AAindex é uma base de dados de índices numéricos que representam várias propriedades físico-químicas e bioquímicas de aminoácidos e pares de aminoácidos(Kawashima & Kanehisa 2000, Kawashima et al. 2008).

[3] Os valores entre parênteses correspondem aos códigos de cada propriedade físico-química e bioquímica na base AAindex.

6.1.2.3 Máquinas de Vetores Suporte

Máquinas de vetores de suporte (SVM) constituem algoritmos de aprendizado supervisionado para classificação e regressão, que têm sido utilizadas com grande frequência em trabalhos de predição por alcançarem uma precisão notável em seus resultados. SVM geram classificadores binários, isto é, que realiza uma separação automática entre duas classes diferentes por meio de um hiperplano, ou fronteira linear, posicionado entre os exemplos das duas classes, sendo que os exemplos mais próximos do hiperplano são chamados de vetores de suporte (Lorena & Carvalho 2007).

Apesar de SVM ser uma técnica que apresenta bons índices de generalização, o desempenho da predição depende da seleção da função *Kernel* e seus parâmetros. A seleção de parâmetros inadequados pode resultar em perda de acurácia dos resultados. Quando os dados não são linearmente separáveis é utilizado uma função *Kernel* que visa otimizar a atuação do separador de classes.

Neste trabalho foi utilizado a função *Kernel* Radial Basis Kernel (RBF), que apresenta dois parâmetros que podem ser variados em busca de um melhor resultado de aprendizagem, são eles C (custo) e γ (gama).

Basicamente, o parâmetro C é uma espécie de tolerância a erros existentes em uma classificação durante o treinamento. Quando o valor de C é elevado, o SVM penaliza fortemente exemplos classificados erroneamente durante o treinamento. Com isto, o modelo se ajusta para evitar bias, prejudicando a generalização e, talvez, conduzindo a super-ajustamento (*overfitting*). Super-ajustamento ocorre quando o modelo se especializa, excessivamente, nos dados utilizados em seu treinamento, gerando um modelo ruim e uma taxa de acurácia baixa para novos dados. Por outro lado, baixos valores de C penalizam levemente erros de classificação e o resultado pode ser uma classificação errônea e um possível sub-ajuste (*underfitting*). Sub-ajuste ocorre quando o modelo não consegue capturar os padrões existentes nos dados utilizados no seu treinamento, com isto o modelo apresenta baixa taxa de acertos.

O parâmetro gama pertencente à função *Kernel* (para lidar com classificação não linear) geralmente é definido por uma distribuição Gaussiana. Em um valor de gama alto, a decisão tende a ser flexível, de caráter geral, considerando

quase todos os pontos de dados e, por isso, a decisão é muito suave. Com isto, ela tende a considerar classificações erradas enquanto prediz, evitando o perigo de *overfitting*. Por outro lado, para um valor de gama menor, a fronteira de decisão tende a ser rigorosa e estrita, em contraste com a situação anterior, tende a *overfitting*. Assim, os parâmetros C e γ devem ser selecionados de acordo com a base, uma vez que influenciam na capacidade de classificação dos dados e na generalização dos mesmos.

Neste trabalho foi utilizada a ferramenta LIBSVM (do inglês *A Library for SVMs*) (Chang & Lin 2011) para treinar e testar cada base de dados, bem como identificar os melhores parâmetros C e γ. Para isto, foi realizada uma busca em *grid* em cada base de dados, onde $C = 2^a$ e $\gamma = 2^b$, com *a* variando entre -5 e 15 e *b* variando entre-15 e 3. Para cada ponto da dupla *a* e *b*, foi realizada uma validação cruzada (5-*fold*) nos dados de treinamento para avaliar o desempenho dos modelos SVM.

A técnica de validação cruzada divide a base de treinamento em cinco subconjuntos iguais e mutuamente exclusivos, sendo que um subconjunto é utilizado para treino e os demais para teste (validação). Este processo é realizado cinco vezes alternando de forma circular o subconjunto de teste, onde no final do ciclo calcula-se a acurácia sobre os erros encontrados, obtendo assim uma medida mais confiável sobre a capacidade do modelo.

O formato dos dados utilizados para treinamento e teste pelo algoritmo LIBSVM deve ser o seguinte:

$$<label_1><index_1>:<value_1><index_2>:<value_2>...<index_n>:<value_n>$$
$$<label_2><index_1>:<value_1><index_2>:<value_2>...<index_n>:<value_n>$$
$$...$$
$$<label_m><index_1>:<value_1><index_2>:<value_2>...<index_n>:<value_n>$$

Cada linha contém uma instância utilizada no processo de treinamento e teste. Para classificação o <*label*> é um inteiro que indica a classe, neste caso epítopo positivo ou epítopo negativo/não-epítopo, 1 ou -1, respectivamente. O <*index*> é um inteiro que representa um atributo, isto é, uma propriedade físico-química ou taxa de proporção de aminoácido e, por fim <*value*> é um valor real, que representa o valor do atributo.

Os atributos são obtidos da base de epítopos positivos, negativos e não-epítopos por meio de cálculos que tomam como base uma janela deslizante composta de 20 aminoácidos, pois foi este tamanho que apresentou os melhores resultados no trabalho de Lin et al. (2013). O método da janela deslizante realiza uma varredura da sequência de aminoácidos usando uma janela móvel e atribui uma média ao aminoácido central da janela na a propriedade *j*, para *j* = 1, 2,...,15 segundo a Equação 6.

$$avgEscala_j = \frac{\sum_i (1 - f|c - i|)s_i}{w} \qquad \textbf{Equação 6}$$

Onde i é o índice da posição do resíduo na janela deslizante, c é o índice da posição do resíduo central da janela, $|c - i|$ é a distância em número de resíduos entre o resíduo *i* e o resíduo central, f é o fator de peso linear (neste trabalho o melhor valor para esta variável foi 0.08), s_i é o valor da propriedade físico-química ou taxa de proporção de aminoácido do resíduo na posição i.

Desta forma, cada janela deslizante é marcada com +1 se o elemento central da janela pertence a um epítopo positivo ou -1 se o elemento central da janela não é parte de um epítopo positivo (podendo ser epítopo negativo ou não-epítopo). Por exemplo, se o elemento central for epítopo positivo uma instância do arquivo será

+1 1:avgEscala$_1$ 2:avgEscala$_2$ 3:avgEscala$_3$ ... 15:avgEscala$_{15}$

Para impedir que atributos com valores numéricos muito grandes dominem valores bem menores, os valores dos atributos são normalizados entre [-1,1] utilizando um programa da ferramenta LIBSVM.

6.1.3 MEDIDAS DE AVALIAÇÃO

O treinamento do SVM em cada uma das oito bases de treinamento, listadas na seção 6.1.1, gerou oito modelos SVM (utilizamos o mesmo nome de cada base de treinamento, porém referindo como modelo). Desta forma, uma vez criado os modelos SVM para cada base de treinamento, buscou-se compará-las na predição de

epítopos de protozoários, bem como nas bases de dados Sollner e AntiJen1. Para isto, foram utilizadas as medidas comumente empregadas para avaliar a qualidade de ferramentas de predição, tais como sensibilidade, especificidade, acurácia e o MCC, descritas pelas equações 7, 8, 9 e 10.

$$R_{sen} = \frac{TP}{TP + FN}$$ **Equação 7**

$$R_{esp} = \frac{TN}{TN + FP}$$ **Equação 8**

$$R_{acc} = \frac{TP + TN}{TP + FP + TN + FN}$$ **Equação 9**

Onde TP representa o número de verdadeiro-positivos, neste caso, quantidade de elementos centrais de uma janela *w* preditos como epítopos positivo e, realmente, pertence a um epítopo; TN representa o número de verdadeiro-negativos, isto é, quantidade de elementos centrais de uma janela *w* preditos como não epítopos[4] e, realmente, pertence a um não epítopo e; FP o número de falso-positivos, quantidade de elementos centrais de uma janela *w* predito como epítopo, mas que pertence a um não epítopo e FN número de falso-negativos, isto é, quantidade de elementos centrais de uma janela *w* predito como não epítopo, mas que pertence a um epítopo. Ademais, R_{sen} reflete a sensibilidade, isto é, a taxa de epítopos que são corretamente preditas como epítopos; R_{esp} reflete a especificidade, isto é, a taxa de não epítopos preditos corretamente como não epítopos; R_{acc} reflete a acurácia, isto é, a taxa de peptídeos corretamente preditos;

Além disso, o MCC foi utilizado (Equação 10). Este coeficiente proposto por Matthews (1975) é considerado uma medida balanceada que pode ser usada mesmo quando as classes em estudo são de tamanhos muito desiguais. Assume valores entre -1 e +1, onde um valor igual a +1 corresponde à predição perfeita (total

[4]Não epitopo, neste contexto, refere-se a resíduo que não é um epitopo, independente se ele é não-epitopo (com hífen) ou epitopo negativo.

acordo), zero corresponde à predição completamente aleatória e -1, a predição inversa (total desacordo).

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad \textbf{Equação 10}$$

Para comparação dos modelos SVM criados foi utilizado curvas ROC (do inglês, *Receiver Operating Characteristics*), que é um método gráfico útil e poderoso para avaliação de modelos de classificação e predição que utilizam aprendizado de máquinas, sendo uma forma mais eficiente de demonstrar a relação, normalmente, antagônica entre a sensibilidade e a especificidade dos testes. Para comparar classificadores é desejável reduzir a curva ROC a um simples escalar, sendo que um método bastante utilizado é calcular a área sobre a curva ROC, conhecido como AUC (do inglês *Area Under a Curve*).

6.2 RESULTADOS

6.2.1 PARÂMETROS SVM

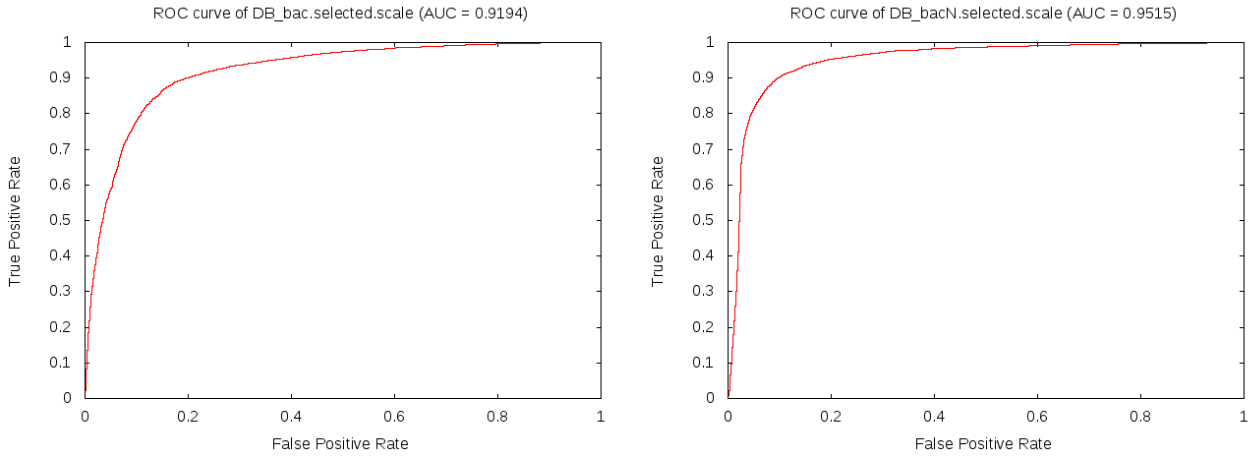
Na Tabela 8 estão apresentados os melhores valores para os parâmetros C e γ para cada um dos modelos, bem como os respectivos valores de acurácia. Observa-se que todos os modelos apresentaram acurácia acima de 85% na validação cruzada 5-*fold*. No entanto, os modelos específicos de bactéria e protozoário obtiveram acurácia acima de 94%.

Tabela 8 - Valores identificados para cada base de dados utilizando o script *grid* da ferramenta LIBSVM

Modelo SVM	Parâmetros		
	C	γ	Acurácia
DB_all_prop	2048	2	85.12
DB_all_propN	2048	2	83.11
DB_all_propN50	32	8	84.04
DB_prot	128	2	95.71
DB_protN	8	8	94.90
DB_bac	512	2	92.01
DB_bacN	8	8	94.94

DB_vir	2048	2	89.09
DB_virN	32	8	88.47

Identificados os melhores parâmetros SVM para cada base de dados, foi calculado a curva ROC, buscando avaliar a habilidade do método implementado em identificar epítopos positivos. Neste sentido, todas as bases de dados utilizadas apresentaram uma curva ROC com acurácia acima de 84%, sendo que as bases de dados com epítopos positivos de protozoários apresentaram os melhores níveis de acurácia (95%) (Figura 14 a Figura 16).



(a) (b)

Figura 14 - Área sob a curva ROC utilizando validação cruzada 5-fold para as bases: a) DB_bac: base de dados composta de epítopos positivos e negativos de bactérias; b) DB_bacN: base de dados composta de epítopos positivos e não-epítopos de bactérias;

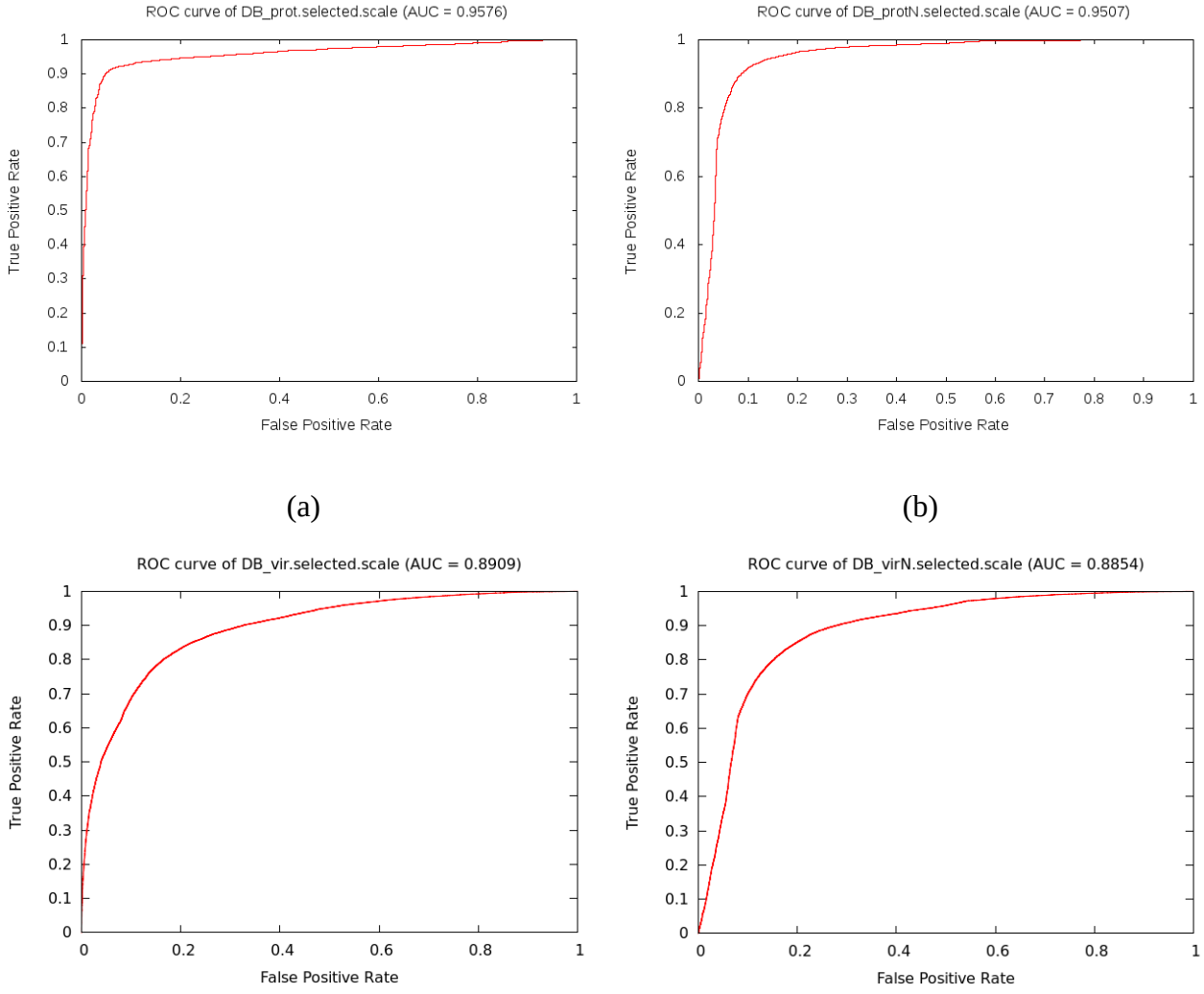
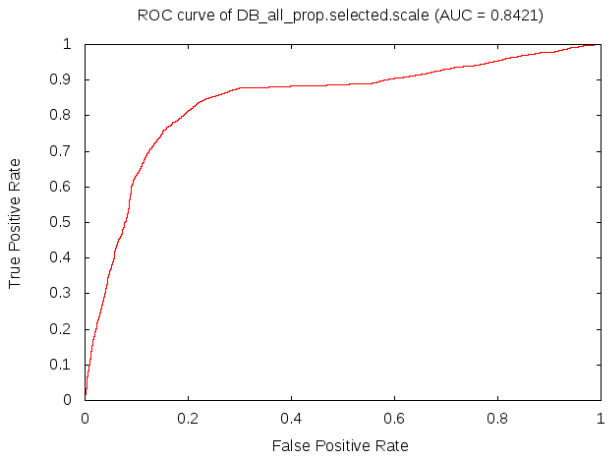
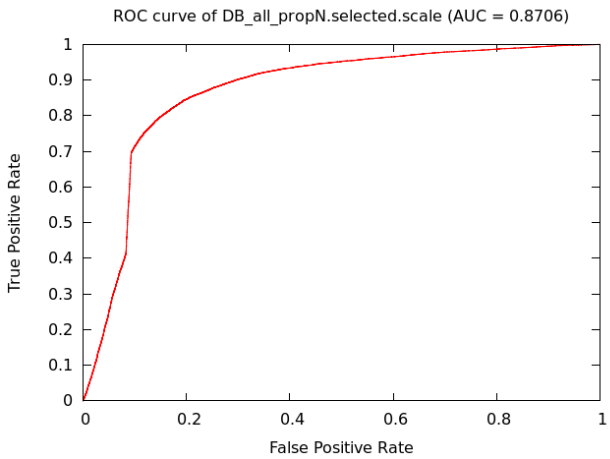


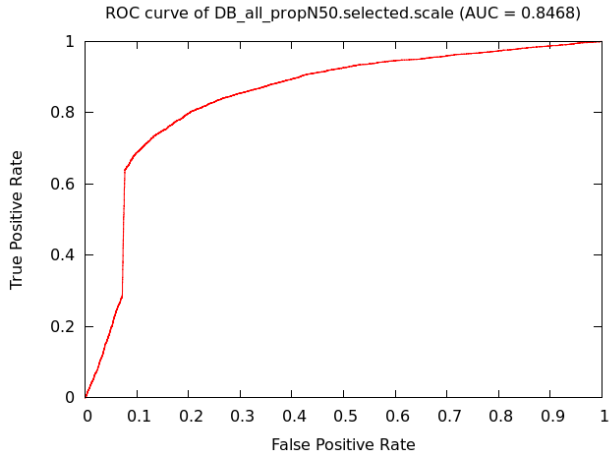
Figura 15 - Área sob a curva ROC utilizando validação cruzada 5-fold para as bases: a) DB_prot: base de dados composta de epítopos positivos e negativos de protozoários; b) DB_protN: base de dados composta de epítopos positivos e não-epítopos de protozoários; c) DB_vir: base de dados composta de epítopos positivos e negativos de vírus; d) DB_virN: base de dados composta de epítopos positivos e não-epítopos de vírus;



(a)



(b)



(c)

Figura 16 - Área sob a curva ROC utilizando validação cruzada 5-fold para as bases: a) DB_all_prop: base de dados composta de epítopos positivos e negativos de todos os táxons; b) DB_all_propN: base de dados composta de epítopos positivos e não-epítopos em quantidade proporcionais de cada taxo; e c) DB_all_propN50: base de dados composta pela metade dos epítopos positivos e não-epítopos existente na base DB_all_propN.

6.2.2 TESTE DE PREDIÇÃO DOS MODELOS

Com os melhores parâmetros SVM identificados, as bases de dados foram utilizadas para treinar e com isto criar os modelos SVM de cada uma. Criado os modelos, estes foram testados nas bases de testes protozoa_epitope, Sollner dataset e AntiJen 1. As bases de dados Sollner e AntiJen 1 foram selecionadas por conterem um maior número de proteínas de protozoários, sendo 8,77% e 9%, respectivamente, do total de proteínas destas bases.

6.2.2.1 Teste dos modelos na base de dados de protozoários

Nas Figura 17a-17d estão apresentadas as taxas de sensibilidade, especificidade, acurácia e MCC por modelos resultantes da aplicação na base de dados de protozoário, respectivamente. Percebe-se que os modelos DB_all_propN, DB_all_propN50 e DB_protN apresentaram altos índices nas quatro medidas e embora os modelos DB_bacN e DB_virN tenham apresentado especificidade e acurácia acima de 65%, o MCC é baixo, o que reflete uma predição em média aleatória.

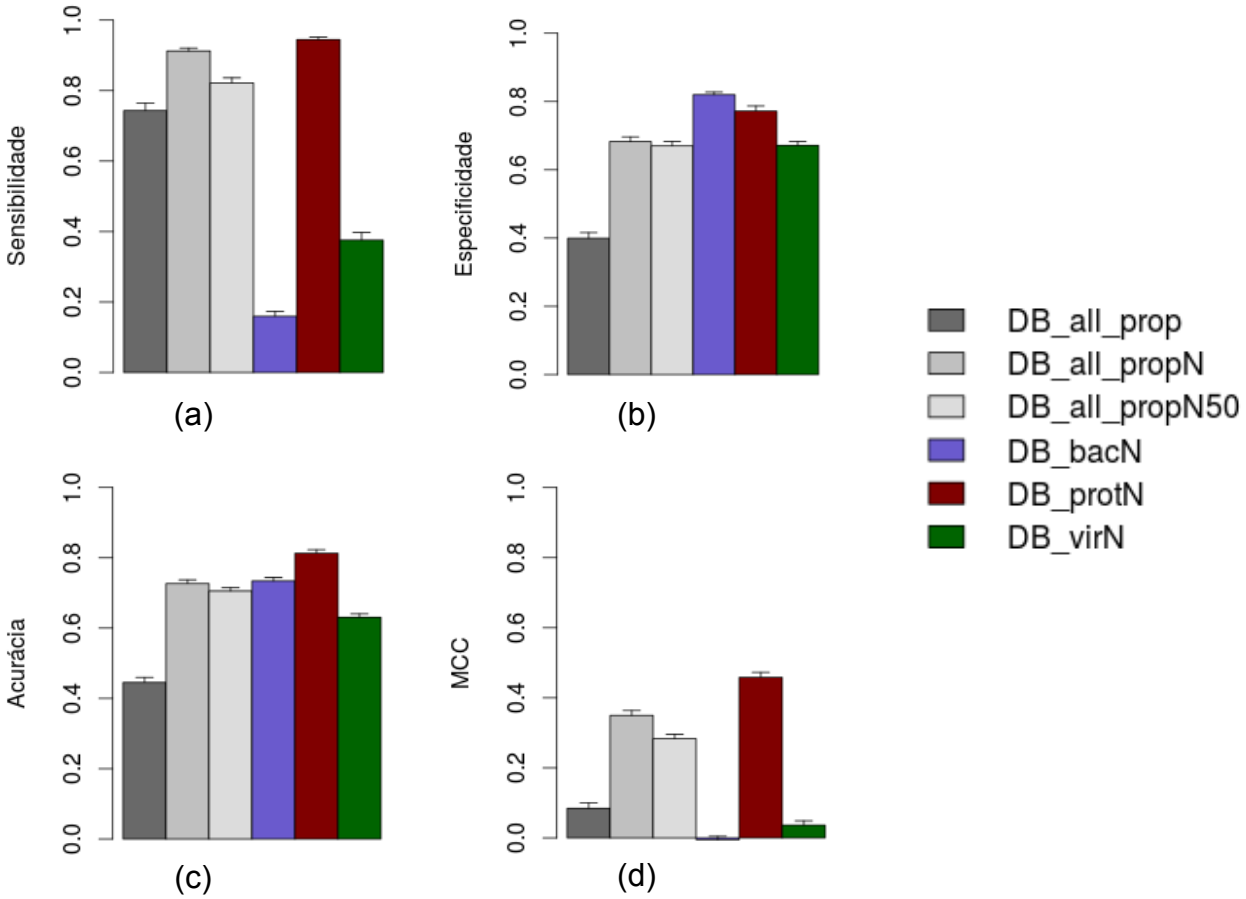


Figura 17 - Resultados (a- Sensibilidade, b- Especificidade, c- Acurácia e d-MCC) das predições na base de teste protozoa_epitope, utilizando os modelos SBP treinados com as bases de dados DB_all_prop, DB_all_propN, DB_all_propN50, DB_bacN, DB_protN e DB_virN.

Os modelos DB_prot, DB_bac e DB_vir, que contém epítopos positivos e negativos de cada táxon apresentaram sensibilidade de 90.65%, 58,04% e 58,84%, especificidade de 27.56%, 43.76% e 39.76%, acurácia de 35.58%, 45.78 e 42.12%, respectivamente.

6.2.2.2 Área sob a curva

Os modelos DB_all_propN, DB_all_propN50 e DB_protN apresentaram AUC de 0.794927, 0.739041 e 0.958896, respectivamente. Já os modelos DB_all_prop, DB_bacN e DB_virN apresentaram menores valores com AUC de 0.578137, 0.472818 e 0.513026, respectivamente.

6.2.2.3 Verificação se os valores das métricas resultantes por modelo provêm da mesma distribuição

Para verificar se os resultados de todos os modelos aplicados na base de dados de protozoários possuem funções de distribuições iguais foi utilizado o teste não paramétrico de Kruskal-Wallis (KW) e o teste de múltipla comparação, fixando o nível significância em $\alpha = 0,05$.

Na Tabela 9 estão os modelos comparados por métrica que não apresentaram diferenças significativas na predição na base de dados de protozoários.

Tabela 9 – Análise estatística (Kruskal-Wallis) dos resultados apresentados pelos diversos modelos aplicados na base de dados de protozoários. Todos os modelos apresentaram p-value < 2.2e-16 e grau de liberdade = 6. São apresentados apenas modelos que não apresentaram diferença significativa.

Métrica	KW qui-quadrado	Modelos comparados	Diferença observada	Diferença crítica
Sensibilidade	527.3673	DB_all_prop – DB_all_propN50	39.48077	99.04464
		DB_all_propN – DB_protN	55.54615	99.04464
Especificidade	383.2284	DB_all_propN – DB_all_propN50	28.12308	99.04464
		DB_all_propN – DB_virN	45.13077	99.04464
		DB_all_propN50 – DB_virN	17.00769	99.04464
		DB_bacN – DB_protN	61.59231	99.04464
Acurácia	412.1849	DB_all_propN – DB_all_propN50	44.44615	99.04464
		DB_all_propN – DB_bacN	17.16538	99.04464
		DB_all_propN50 –DB_bacN	61.61154	99.04464
MCC	514.0708	DB_all_prop – DB_virN	64.16538	99.04464
		DB_all_propN – DB_all_propN50	66.88462	99.04464
		DB_all_propN – DB_protN	93.14615	99.04464
		DB_bacN – DB_virN	59.97308	99.04464

6.2.2.4 Predição de cada modelo por espécie de protozoário

Uma vez que observamos diferenças entre os modelos de predição na base de dados de protozoários, principalmente os modelos DB_all_propN, DB_all_propN50 e DB_protN, realizamos então a avaliação do desempenho de todos os modelos, desta vez por espécie.

Na Tabela 10 está apresentado o desempenho de predição do SBP utilizando os diversos modelos por espécie. Os modelos DB_all_propN, DB_all_propN50 e DB_protN apresentaram altas taxas de sensibilidade, especificidade, acurácia e MCC para quase todas as espécies, exceto *Leishmania panamensis* onde a especificidade e MCC apresentaram valores muito baixos.

Tabela 10–Desempenho do SBP de predição por espécie e por modelo/ferramentas na base de dados de Proteína de protozoários.

Espécie	Nº*	Ferramenta	SEN	ESP	ACC	MCC
Cryptosporidium parvum	1	DB_all_prop	0,30	0,53	0,52	-0,09
		DB_all_propN	1,00	0,56	0,59	0,29
		DB_all_propN50	1,00	0,59	0,62	0,31
		DB_bacN	0,35	0,76	0,73	0,06
		DB_protN	0,90	0,77	0,78	0,38
		DB_virN	0,30	0,52	0,51	-0,09
Leishmania braziliensis	4	DB_all_prop	0,68	0,76	0,74	0,38
		DB_all_propN	0,91	0,70	0,72	0,40
		DB_all_propN50	0,83	0,72	0,74	0,37
		DB_bacN	0,05	0,82	0,73	-0,03
		DB_protN	0,99	0,87	0,88	0,64
		DB_virN	0,32	0,72	0,67	0,03
Leishmania chagasi	1	DB_all_prop	0,59	0,61	0,61	0,09
		DB_all_propN	1,00	0,73	0,74	0,35
		DB_all_propN50	0,53	0,81	0,80	0,19
		DB_bacN	0,00	0,86	0,81	-0,09
		DB_protN	1,00	0,80	0,81	0,42
		DB_virN	0,65	0,67	0,66	0,15
Leishmania donovani	1	DB_all_prop	0,88	0,00	0,62	-0,20
		DB_all_propN	0,98	0,30	0,78	0,43
		DB_all_propN50	0,91	0,52	0,79	0,47
		DB_bacN	0,09	0,96	0,35	0,09
		DB_protN	1,00	0,19	0,76	0,37

		DB_virN	0,20	1,00	0,43	0,26
Leishmania infantum	5	DB_all_prop	0,26	0,62	0,59	0,03
		DB_all_propN	0,98	0,60	0,74	0,29
		DB_all_propN50	0,95	0,61	0,72	0,23
		DB_bacN	0,36	0,77	0,64	0,04
		DB_protN	0,94	0,67	0,86	0,31
		DB_virN	0,63	0,59	0,58	0,03
Leishmania panamensis	1	DB_all_prop	0,74	0,43	0,55	0,17
		DB_all_propN	0,89	0,35	0,58	0,28
		DB_all_propN50	0,95	0,04	0,41	-0,04
		DB_bacN	0,03	0,85	0,51	-0,20
		DB_protN	0,87	0,33	0,55	0,23
		DB_virN	0,03	0,69	0,41	-0,36
Plasmodium falciparum	64	DB_all_prop	0,77	0,38	0,42	0,07
		DB_all_propN	0,89	0,74	0,77	0,35
		DB_all_propN50	0,84	0,73	0,75	0,31
		DB_bacN	0,13	0,85	0,77	0,00
		DB_protN	0,95	0,82	0,85	0,48
		DB_virN	0,32	0,68	0,64	0,02
Plasmodium vivax	19	DB_all_prop	0,78	0,23	0,28	-0,01
		DB_all_propN	0,89	0,65	0,69	0,31
		DB_all_propN50	0,73	0,66	0,68	0,24
		DB_bacN	0,20	0,80	0,73	0,01
		DB_protN	0,92	0,74	0,76	0,40
		DB_virN	0,27	0,68	0,66	0,00
Toxoplasma gondii	33	DB_all_prop	0,67	0,43	0,46	0,08
		DB_all_propN	0,92	0,62	0,67	0,34
		DB_all_propN50	0,82	0,62	0,66	0,27
		DB_bacN	0,20	0,77	0,70	-0,01
		DB_protN	0,92	0,77	0,80	0,47
		DB_virN	0,46	0,66	0,63	0,06
Trypanosoma cruzi	24	DB_all_prop	0,85	0,48	0,54	0,20
		DB_all_propN	0,95	0,68	0,74	0,41
		DB_all_propN50	0,83	0,65	0,70	0,29
		DB_bacN	0,12	0,83	0,74	-0,02
		DB_protN	0,97	0,74	0,79	0,47
		DB_virN	0,49	0,66	0,62	0,10

*número de proteínas de cada espécie.

6.2.3 COMPARAÇÃO DOS MODELOS DO SBP COM OUTRAS FERRAMENTAS DE PREDIÇÃO DE EPÍTOPOS JUNTO A BASE DE DADOS DE TREINAMENTO PROTOZOA_EPITOPE.

Identificados os modelos SBP com melhores taxas de especificidade, sensibilidade, acurácia e MCC, buscamos compará-los com algumas das principais ferramentas de predição de epítopos disponíveis para *download* ou como ferramenta online. Desta forma, foram utilizadas as ferramentas: 1) BepiPred: por ser possível executá-la localmente. Nela foram utilizados *threshold* de 0.6 e 0.8; 2) ABCpred: por disponibilizar uma ferramenta *online* sem limite de consultas e sem impor *captcha*[5] (teste para verificar se o usuário é humano), sendo que foram utilizados *threshold* de 0.6 e 0.8; 3) LEPS: também por disponibilizar uma ferramenta *online* sem limite de consultas e sem impor captcha.

[5] Captcha é um tipo de medida de segurança conhecido como autenticação por desafio e resposta, que oferece proteção contra entradas digitais remotas garantindo que somente um ser humano que possua a senha correta ou saiba reconhecer alguma informação (letras, números, símbolos) possa acessar o sistema.

Na Tabela 11 estão apresentados os resultados da sensibilidade, especificidade, acurácia e MCC de cada uma das ferramentas e modelos do SBP. Nota-se que a ferramenta ABCpred apresenta alta taxa de sensibilidade (100% e 72% para ABCpred-0.6 e ABCpred-0.8, respectivamente), porém baixa especificidade (5% e 39% para ABCpred-0.6 e ABCpred-0.8, respectivamente), o que fez com que seus valores de MCC fossem baixos. A ferramenta LEPS apresentou baixa sensibilidade (15%), alta especificidade (89%), alta acurácia (81%) e um MCC pior do que o apresentado pelo ABCpred. O BepiPred apresentou sensibilidade, especificidade e acurácia em torno de 50% e um MCC próximo do MCC observado para o ABCpred.

Os modelos DB_all_propN, DB_all_propN50 e DB_protN apresentaram sensibilidade acima de 85%, especificidade e acurácia maiores que 70%. Desta forma, seus MCC foram melhores do que os valores apresentados pelas demais ferramentas.

Tabela 11 - Comparação dos modelos SBP com as ferramentas ABCpred, BepiPred e LEPS aplicados a base de teste protozoa_epitope. Nas ferramentas ABCpred e BepiPred foram utilizados threshold de 0.6 e 0.8 em cada uma.

Dataset	Táxon	Ferramenta	SEN	ESP	ACC	MCC
Protozoa_epitope	Protozoário	ABCpred-0.6	1,00	0,05	0,14	0,05
		ABCpred-0.8	0,72	0,39	0,42	0,06
		BepiPred l-0.6	0,54	0,56	0,55	0,03
		BepiPred l-0.8	0,46	0,65	0,62	0,05
		DB_all_propN	0,93	0,71	0,74	0,33
		DB_all_propN50	0,85	0,70	0,72	0,28
		DB_protN	0,97	0,82	0,84	0,45
		LEPS	0,15	0,89	0,81	0,01

6.2.3.1 Teste dos modelos na base de dados Sollner e Antijen1

Na Tabela 12 estão apresentados os resultados da comparação de desempenho das ferramentas ABCpred, BepiPred, LEPS e todos os modelos SVM do SBP aplicados nas bases de dados Sollner e Antijen1.

Tabela 12 - Comparação do desempenho das ferramentas ABCpred, BepiPred, LEPS e todos os modelos SVM do SBP nas bases de dados Sollner e Antijen1.

Dataset	Taxo	Ferramenta	SEN	ESP	ACC	MCC
Sollner	Bactéria	ABCpred-0.6	1,00	0,05	0,11	0,04
		ABCpred-0.8	0,57	0,37	0,38	-0,01
		BepiPred-0.6	0,29	0,71	0,68	0,00
		BepiPred-0.8	0,18	0,80	0,76	-0,01
		DB_all_prop	0,55	0,55	0,56	0,05
		DB_all_propN	0,46	0,63	0,61	0,04
		DB_all_propN50	0,35	0,65	0,62	-0,01
		DB_bacN	0,20	0,77	0,73	-0,02
		DB_protN	0,29	0,77	0,74	0,04
		DB_virN	0,40	0,63	0,61	0,02
		LEPS	0,05	0,95	0,90	0,02
Sollner	Protozoário	ABCpred-0.6	1,00	0,06	0,19	0,05
		ABCpred-0.8	0,82	0,45	0,46	0,11
		BepiPred-0.6	0,68	0,58	0,56	0,08
		BepiPred-0.8	0,65	0,66	0,60	0,10
		DB_all_prop	0,84	0,42	0,48	0,13
		DB_all_propN	0,82	0,61	0,65	0,17
		DB_all_propN50	0,78	0,65	0,67	0,18
		DB_bacN	0,18	0,82	0,74	0,01
		DB_protN	0,75	0,75	0,76	0,23
		DB_virN	0,33	0,68	0,64	-0,03
		LEPS	0,13	0,93	0,77	0,07
Sollner	Vírus	ABCpred-0.6	0,99	0,04	0,08	0,02
		ABCpred-0.8	0,77	0,35	0,37	0,04
		BepiPred-0.6	0,30	0,76	0,74	0,05
		BepiPred-0.8	0,23	0,83	0,80	0,05
		DB_all_prop	0,50	0,55	0,54	0,03
		DB_all_propN	0,46	0,60	0,59	0,01
		DB_all_propN50	0,48	0,62	0,62	0,04
		DB_bacN	0,15	0,78	0,75	-0,02
		DB_protN	0,19	0,78	0,75	-0,01
		DB_virN	0,53	0,64	0,64	0,06
		LEPS	0,07	0,95	0,90	0,03
Antijen1	Bactéria	ABCpred-0,6	0,99	0,06	0,16	0,05
		ABCpred-0,8	0,59	0,37	0,41	0,00
		BepiPred-0,6	0,45	0,71	0,69	0,09
		BepiPred-0,8	0,31	0,80	0,75	0,07
		DB_all_prop	0,42	0,59	0,57	0,00
		DB_all_propN	0,48	0,60	0,58	0,02
		DB_all_propN50	0,47	0,61	0,59	0,04
		DB_bacN	0,25	0,77	0,72	0,01

		DB_protN	0,28	0,73	0,69	0,02
		DB_virN	0,36	0,63	0,60	-3,00
		LEPS	0,20	0,95	0,87	0,11
Antijen1	Protozoário	ABCpred-0,6	0,99	0,09	0,20	0,07
		ABCpred-0,8	0,80	0,43	0,45	0,12
		BepiPred-0,6	0,60	0,65	0,63	0,09
		BepiPred-0,8	0,45	0,70	0,65	0,07
		DB_all_prop	0,62	0,39	0,41	0,02
		DB_all_propN	0,61	0,64	0,61	0,09
		DB_all_propN50	0,50	0,66	0,63	0,05
		DB_bacN	0,11	0,84	0,76	-0,02
		DB_protN	0,60	0,82	0,76	0,16
		DB_virN	0,20	0,63	0,60	-0,09
		LEPS	0,26	0,90	0,82	0,13
Antijen1	Vírus	ABCpred-0,6	0,99	0,04	0,11	0,03
		ABCpred-0,8	0,77	0,33	0,37	0,05
		BepiPred-0,6	0,36	0,68	0,66	0,03
		BepiPred-0,8	0,27	0,76	0,72	0,03
		DB_all_prop	0,57	0,49	0,49	0,02
		DB_all_propN	0,45	0,60	0,58	0,03
		DB_all_propN50	0,38	0,62	0,60	0,02
		DB_bacN	0,20	0,77	0,73	-0,01
		DB_protN	0,15	0,80	0,75	-0,02
		DB_virN	0,47	0,60	0,60	0,05
		LEPS	0,21	0,91	0,85	0,09

7 DISCUSSÃO

Identificar novos marcadores imunológicos e moleculares, como epítopos e microssatélites pode permitir uma rápida seleção de potenciais alvos para uso em diagnósticos, protocolos vacinais e como imunoterapêuticos. Na era pós-genômica, a integração de conhecimentos das áreas biológicas, de computação, de estatística e de bioinformática tem favorecido a identificação em larga escala destes marcadores.

Métodos experimentais de identificação de marcadores imunológicos e moleculares apresentam elevados custos e requerem longos períodos de experimentação. Uma alternativa empregada é a utilização de métodos *in silico* de identificação destes marcadores, haja visto a grande quantidade de sequências genômicas e proteômicas disponíveis em bancos de dados públicos.

Para identificação de marcadores moleculares do tipo repetições em tandem existem várias ferramentas disponíveis, tais como: IMEx, MISA, Mreps, SciRoKo, Sputnik e TROLL. No entanto, além de identificar as repetições faz-se necessário que as ferramentas apresentem outros recursos, como visualização gráfica, rapidez, capacidade de tratar grandes quantidades de dados genômicos e proteômicos, ambiente multiplataforma e flexibilidade nos parâmetros de identificação. Recursos estes que visam auxiliar os pesquisadores no reconhecimento e análise dos marcadores moleculares.

Neste sentido, foi apresentado na primeira parte deste trabalho o desenvolvimento da ferramenta *web* e *stand-alone* ProGeRF, que é capaz de identificar marcadores de diagnóstico com potencial de ser espécie-específico do tipo repetições em tandem em dados genômicos ou proteômicos. O algoritmo do ProGeRF é inspirado no método de busca rápida em grandes bases de dados de DNA (Ning et al. 2001), que explora o fato que os computadores atuais disponibilizam suficiente quantidade de memória RAM (do inglês, *Random Access Memory*) e, assim, permitem armazenar uma tabela *hash* que descrevem dados genômicos e proteômicos contendo vários gigabytes.

Na comparação da quantidade de detecções e tempo de execução (em segundos) entre a ferramenta ProGeRF e as ferramentas Mreps, Sputnik, GMATo, SciRoKo e TRF (Tabela 2), nota-se que o número de elementos repetitivos

identificados pelas ferramentas SciRoKo, Sputnik e Mreps é menor que das ferramentas MISA, GMATo, TRF e ProGeRF, nos dados genômicos de *Plasmodium falciparum chromosome* IV (NC 004318.1), *Saccharomyces cerevisiae chromosome* IV (NC 001136.8) e *Mycobacterium tuberculosis* H37Rv genome (NC 000962.2).

No entanto, GMATo é mais lenta do que as ferramentas MISA, TRF e ProGeRF, e não especifica a forma como trata sobreposições de elementos repetitivos. Com relação a ferramenta MISA, Wang et al. (2013) relata que loci extras são identificados redundantemente em microssatélites sobrepostos, o que faz com que o número de elementos repetitivos encontrados sejam superestimados.

Um número menor de elementos repetitivos foi encontrado pelas ferramentas SciRoKo, Sputnik e Mreps. As possíveis razões para esta baixa quantidade de repetições são: a) Sputnik não reporta hexanucleotídeos, pois o tamanho máximo identificado é pentanucleotídeo; b) de acordo com Mudunuri et al (2010) ferramentas baseadas em *scores* como SciRoKo e Sputnik que usam altas penalidade de mismatch (tais como 5, 6 e 7) e baixo peso de match (tais como 1 e 2) dificulta a identificação de pequenos microssatélites (mono-tri); e c) Mreps é limitada na quantidade repetições, uma vez que não utiliza busca exaustiva no processo de identificação de repetição, sendo que nem todos os motivos repetitivos passam pela avaliação da heurística/análise estatísticas utilizada (Kolpakov et al. 2003, Mudunuri & Nagarajaram 2007, Kofler et al. 2007).

Outro fator a ser considerado pelas ferramentas de identificação de repetições em tandem é a capacidade de tratar grandes datasets (Wang et al. 2013). Neste sentido, as ferramentas MISA, Sputnik, GMATo, SciRoKo, TRF e ProGeRF conseguiram executar e identificar repetições no arquivo fasta de *Setaria italica* de 515Mb. As ferramentas MISA, GMATo, TRF e ProGeRF encontraram os maiores números de repetições e, com relação ao tempo, ProGeRF apenas não foi mais rápida que a ferramenta MISA. No entanto, a ferramenta MISA não trata sobreposições, o que a faz ser mais rápida em comparação com a ProGeRF e a ferramenta GMATo só realiza busca por repetições perfeitas.

No que diz respeito à ferramenta TRF, esta permite a identificação de sobreposição, sendo que a redundância é, no máximo, de três vezes o tamanho do padrão. Por esta razão, TRF apresenta uma quantidade maior de repetições do que ProGeRF. Por padrão o ProGeRF não permite sobreposições e escolhe a maior

sequência de elementos repetitivos. Entretanto, o usuário pode definir a porcentagem de sobreposições através do parâmetro -v. O tempo de execução do ProGeRF foi menor que o tempo da ferramenta TRF e sete vezes menor que da ferramenta GMATo.

Quanto à capacidade das ferramentas de detectar repetições na mesma localização física do genoma percebeu-se que mais de 75% das regiões contendo microssatélites identificados pelas ferramentas SciRoKo, Sputnik e Mreps também foram identificadas pela ferramenta ProGeRF nas sequências NC_004318.1, NC_001136.8 e NC_000962.2, sendo que para as ferramentas GMATo e MISA a ferramenta ProGeRF identificou 100% dos microssatélites localizados pelas mesmas ferramentas (Tabela 3). Sputnik e TRF apresentaram uma baixa quantidade de loci coberto pela ferramenta ProGeRF nas sequências NC_001136.8 e NC_000962.2. Esta baixa cobertura é consequência da falta de parâmetros naquelas ferramentas que limitem os tamanhos máximo e mínimo de repetições.

Assim, filtramos os resultados das ferramentas Sputnik e TRF, limitando os resultados para repetições com no mínimo 5 repetições e motivos de tamanho máximo de 6 bp e tamanho mínimo de 1 bp. Desta forma, a cobertura de ProGeRF aumentou para 100% sobre os resultados da ferramenta Sputnik e mais que 80% sobre os resultados da ferramenta TRF (97% para as sequências NC 004318.1 e NC 001136.8).

Por outro lado, a cobertura das ferramentas SciRoKo, Mreps e Sputnik sobre a ferramenta ProGeRF é menor que 46% para todas as sequências, sendo que as duas últimas são menores que 9%. Isto é consistente com o fato que ProGeRF detecta um número maior de elementos repetitivos do que outras ferramentas.

No que diz respeito à identificação de marcadores imunológicos, muitas ferramentas computacionais baseadas em técnicas de aprendizado de máquina têm sido utilizadas nos últimos anos na identificação *in silico* de epítopos lineares de células B. As principais ferramentas são Lbtope (Singh et al. 2013), BEEPro (Lin et al. 2013), SVMTrip (Yao et al. 2012), B-pred (Giaco et al. 2012), BEST (Gao et al. 2012), LEPS (Wang et al. 2011), BayesB (Wee et al. 2010a), BCPred (EL-Manzalawy et al. 2008), Chen (Chen et al. 2007), ABCpred (Saha & Raghava 2006) e BepiPred (Larsen et al. 2006) e estas têm utilizado técnicas como SVM, RNA e Modelos Ocultos Markovianos.

Todavia, muitas ferramentas desenvolvidas com o objetivo de predizer epítopos lineares em proteínas têm apresentado uma baixa taxa de precisão na predição, quando aplicados em dados de protozoários. Por exemplo, apesar do BepiPred (Larsen et al. 2006) ser uma ferramenta bastante utilizada em estudos em larga escala por permitir que as buscas sejam realizadas em uma máquina local, trabalhos recentes (Wang et al. 2011, Lin et al. 2013, Goodswen et al. 2013) relatam que esta ferramenta apresenta baixa acurácia na predição de epítopos lineares de células B.

Um dos fatores críticos que pode influenciar esta baixa precisão está relacionado ao fato que o desempenho de métodos de predição de epítopos, baseados em técnicas de aprendizado de máquina, depende criticamente do conjunto de dados usados para o treinamento. Geralmente as bases de treinamento estão sub-representados com dados de protozoários e, consequentemente, as ferramentas não conseguem reconhecer os padrões dos dados genômicos e proteômicos destes micro-organismos.

A base de dados Sollner (Sollner et al. 2008) utilizada para treinar a ferramenta BEEPro (Lin et al. 2013) apresenta 49% de epítopos de bactérias, 42% de epítopos de vírus e 9% de epítopos de protozoários (Filos Sarcomastigophora - ordem Kinetoplastida - e Apicomplexa). As bases de dados AntiJen1 e AntiJen2 (Toseland et al. 2005) utilizadas para treinar a ferramenta BepiPred (Larsen et al. 2006) e testar as ferramentas BEEPro (Lin et al. 2013) e LEPS (Wang et al. 2011) apresentam, respectivamente, 26% e 27% de epítopos de bactérias, 39% e 31% de epítopos de vírus e 9% e 2,5% de epítopos de protozoários. As bases PCdataset (Wang et al. 2011) e Pellequer (Larsen et al. 2006) não apresentam em seus dados epítopos oriundos de protozoários.

Nota-se que, nas principais bases de dados utilizadas para treinamento das ferramentas de predição a presença de dados oriundos de protozoários não é maior que 9%, fato este que pode ser umas das justificativas do baixo desempenho de métodos de predição de epítopos para este táxon. No entanto, vale lembrar que a baixa quantidade de epítopos de protozoários em bases de dados de treinamento reflete a baixa quantidade de dados de pesquisas de validação experimental de epítopos de protozoários (Resende et al. 2012).

Diante disto, esta etapa do trabalho teve como objetivo verificar se ferramentas treinadas com dados de diferentes táxons podem influenciar significativamente na predição de epítopos de protozoários.

O método implementado para predição de epítopos e comparação das bases de treinamentos neste trabalho, que chamamos de SBP, foi inspirada no trabalho de Lin et al. (2013) e utiliza SVM como técnica de aprendizado de máquina, sendo que foram utilizadas como atributos de predição 14 propriedades físico-químicas dos aminoácidos e taxa de proporção de aminoácidos nas bases de dados de epítopos positivos, epítopos negativos e não-epítopos.

O SBP foi treinado e testado em diferentes bases de dados e para cada base de dados foram identificados os valores ótimos dos parâmetros SVM C e γ da função Kernel RBF. Para a base de dados DB_all_propN, por exemplo, em uma validação cruzada de tamanho cinco, SBP obteve seu melhor desempenho com os valores C= 2048 e γ=2, com uma acurácia de 83.11% (Tabela 8).

No teste realizado na base de dados de protozoários, o modelo DB_protN alcançou elevados índices de desempenho, com sensibilidade de 94%, especificidade de 77%, acurácia de 81% e MCC = 0.45. Uma vez que o modelo contém os resíduos de todos os epítopos positivos e não-epítopos não redundantes do IEDB e a base de protozoário, basicamente, é composta pelas mesmas informações (exceto que no teste é utilizado todos os resíduos das proteínas, fazendo com que exista mais não-epítopos do que epítopos positivos) era esperado que tal modelo atingisse elevados índices de desempenho.

No entanto, os modelos contendo resíduos de epítopos positivos e não-epítopos em quantidade proporcionais de todos os táxons, também atingiram bons resultados. O modelo DB_all_propN alcançou sensibilidade de 91%, especificidade de 68%, acurácia de 72% e MCC de 0.34 e o modelo DB_all_propN50 apresentou sensibilidade de 82%, especificidade de 66%, acurácia de 70% e MCC de 0.28 (Figura 17).

Uma vez que os modelos DB_all_propN e DB_protN contêm todos os resíduos de epítopos positivos extraídos do IEDB, estes poderiam apresentar certo grau de *overfitting* para dados de protozoários. No entanto, o modelo DB_all_propN50 que contém metade dos resíduos de epítopos positivos de protozoários extraídos do

IEDB, sendo 1/3 dos resíduos de epítopos positivos do modelo, também apresentou bons resultados de especificidade, sensibilidade, acurácia e MCC, o que permite dizer que o modelo DB_all_propN50 obteve bons índices de aprendizagem e, com isto, capacidade de generalização.

O teste não paramétrico de KW mostra que não existe uma diferença significativa entre a sensibilidade do modelo DB_all_prop e o modelo DB_all_propN e entre a sensibilidade do modelo DB_all_propN e DB_protN (Tabela 9). Com isto, nota-se que o modelo treinado proporcionalmente, com apenas metade dos resíduos de epítopos positivos de protozoários extraídos do IEDB, consegue predizer que um dado resíduo é um epítopo positivo corretamente com taxas similares aos modelos contendo todos os resíduos de epítopos de protozoários extraídos do IEDB.

De forma semelhante, não existe diferença significativa na especificidade, acurácia e MCC entre os modelos DB_all_propN e DB_all_propN50. Isto colabora com a ideia de que o modelo DB_all_propN50 apresenta boa capacidade de generalização, quando comparado com um modelo que contém todo os resíduos de epítopos positivos de protozoários extraídos do IEDB, caso do DB_all_propN.

O MCC é utilizado em técnicas de aprendizado de máquina como uma medida de qualidade de classificação binária. Ele retorna valores entre -1 e +1. Um coeficiente de +1 representa uma predição perfeita (total acordo), 0 uma predição completamente aleatória e -1 uma predição inversa (total desacordo). Assim sendo, mesmo que o modelo utilizando apenas epítopos positivos de bactérias tenha apresentado alta taxa de especificidade (82%) e acurácia (73%) na base de teste de protozoário, o MCC (-0,0051) nos diz que este modelo apresenta uma predição em média aleatória/ruim.

No teste de KW, os modelos DB_all_propN50 e DB_all_propN não apresentam diferença significativa na acurácia quando comparados com o modelo DB_bacN. No entanto, quando é observado o MCC, que diz que o modelo DB_bacN apresenta uma predição aleatória, os modelos DB_all_propN50 e DB_all_propN diferem significativamente dele e não diferem significativamente entre si. Neste sentido, também, é possível notar que o modelo DB_virN não apresenta diferença significativa do modelo DB_bacN, no que diz respeito ao MCC. Considerando tais informações, utilizar modelos treinados com dados de bactérias e vírus para predizer

epítopos de protozoários, de maneira geral, conduz a uma predição, em média, próxima do aleatório.

Quando observado o desempenho do SBP por modelo/espécie na base de dados protozoa_epitope (Tabela 10), pode-se notar que para algumas espécies a predição utilizando o modelo DB_virN (vírus) se mostrou viável. Para as espécies *Leishmania chagasi* e *Leishmania infantum* o modelo DB_virN apresentou grau de eficiência considerável, uma vez que a sensibilidade, especificidade e acurácia atingiram valores maiores que 58%. As espécies *Toxoplasma gondii, Trypanosoma cruzi, Plasmodium falciparum* e *Leishmania donovani*, também, apresentaram boa especificidade e acurácia quando predito utilizando o modelo DB_virN. No entanto, a sensibilidade para os dois primeiros foi menor do que das espécies *Leishmania chagasi* e *Leishmania infantum e* para os dois últimos a sensibilidade foi bem menor. Em contrapartida, o modelo DB_bacN (bactérias) na grande maioria das espécies de protozoários conduziu a uma predição ruim ou próxima do aleatório.

No que diz respeito aos modelos DB_all_propN, DB_all_propN50 e DB_protN, de maneira geral, apresentaram bons índices de sensibilidade, especificidade, acurácia e MCC em quase todas as espécies, exceto para a espécie *Leishmania panamensis* que apresentou baixo MCC no modelo DB_all_propN50 (Tabela 10). É importante ressaltar, que mesmo espécies com um número pequeno de proteínas na base de treinamento apresentaram bons índices de sensibilidade nos modelos DB_all_propN e DB_all_propN50. No entanto, para estes mesmos modelos as taxas de especificidade e MCC atingiram índices melhores quando as quantidades de proteínas de cada espécie apresentavam quantidades maiores que um.

Na comparação entre as ferramentas de predição nas bases Protozoa_epitope, Sollner e Antijen1, observa-se que a ferramenta LEPS em média apresenta alta especificidade, baixa sensibilidade e alta acurácia. No entanto, isto não reflete uma boa capacidade de predição, uma vez que na base de teste as classes (epítopos positivos e não-epítopos) estão desbalanceadas (Chawla et al. 2004).

A maioria das proteínas testadas apresenta um número maior de resíduos de não-epítopos (em média 86%, na base protozoa_epitope) do que resíduos de epítopos positivos (em média 14%, na base protozoa_epitope). Desta forma, dizer que aleatoriamente todos resíduos são não-epítopos pode proporcionar uma taxa de especificidade alta, assim como a acurácia. Por tanto, neste contexto onde grupos de

epítopos positivos e não-epítopos não são balanceados em uma proteína, uma ferramenta que apresenta alta especificidade apresentará uma acurácia bem mais elevada do que uma ferramenta que apresenta alta sensibilidade. Por esta razão, que a ferramenta LEPS sempre apresenta uma alta acurácia, assim como, o modelo DB_bacN nos testes com a base protozoa_epitope.

Em situação inversa, a ferramenta ABCpred com *threshold* de 0.6 apresenta, em média alta sensibilidade, uma especificidade baixíssima e uma acurácia baixa. O que pode-se inferir é que esta ferramenta tem predito que quase todos os resíduos das proteínas testadas são epítopos positivos e, assim, até não-epítopos são ditos epítopos positivos, razão esta que justifica a sua baixa especificidade. Uma vez que as proteínas são compostas na sua grande maioria por não-epítopos (em média 86% resíduos na base protozoa_epitope), predizer, erroneamente, que a maioria dos resíduos são epítopos positivos conduz a uma baixa acurácia, mesmo acertando grande parte dos que realmente são epítopos positivos.

A ferramenta BepiPred, na base de dado de Sollner, para vírus e bactérias apresentou alta especificidade (>70%), porém com baixa sensibilidade (≥30%). Assim, devido ao desbalanceamento entre epítopos positivos e não-epítopos nas bases de teste a ferramenta apresentou acurácia maior que 66% nos táxons de bactérias e vírus. Em dados de protozoários a ferramenta BepiPred com *threshold* de 0.8 apresentou sensibilidade e especificidade ≈ 65% e acurácia de 60%, ou seja, melhores resultados do que os de bactérias e vírus quando se leva em consideração a sensibilidade, especificidade e acurácia.

Comportamento similar aconteceu na base de dados AntiJen1, porém com uma pequena melhora na sensibilidade. No entanto, vale ressaltar que a base de dados AntiJen1 foi utilizada para construir e treinar o modelo oculto de Markov empregado na ferramenta BepiPred (Larsen et al. 2006). O MCC apresentado nos resultados da ferramenta BepiPred apenas apresentou predição ruim no táxon de bactérias da base de dados Sollner.

A base de dados Sollner contém 8,77% de proteínas de protozoários, destas 40% são de *Plasmodium falciparum*, 40% de *Toxoplasma gondii* e 20% de *T. cruzi*. A base de dados AntiJen1 contêm 9% de proteínas de protozoários, destas 70% são de *Plasmodium falciparum*, 10% de *Leishmania infantum* e 20% de *Leishmania*

donovani. Desta forma, quando observado os dados de protozoários as duas bases são compostas na grande maioria por dados de *Plasmodium falciparum*.

Percebe-se que, na comparação dos modelos com outras ferramentas utilizando as bases de dados Sollner e Antijen1, os melhores índices de predição para as ferramentas BepiPred-0.6, BepiPred-0.8, ABCpred-0.8, bem como para os modelos DB_all_propN, DB_all_propN50 e DB_protN aconteceram no táxon de protozoário. Da mesma forma, modelos SBP treinados com dados específicos de cada táxon, também, alcançaram bons resultados nos seus respectivos táxons.

Uma vez que as bases de dados utilizadas para treinar as ferramentas BepiPred e ABCpred são compostas na sua maioria por dados de vírus é possível que os melhores índices dessas ferramentas obtidos em dados de protozoários (neste caso *Plasmodium falciparum* e *Toxoplasma gondii*) seja pelo menos em parte devido à reatividade cruzada. Uma série de estudos têm confirmado que anticorpos produzidos contra *Plasmodium falciparum* apresentam reatividade cruzada com alguns retrovírus em experimentos de *Western blot*, como HIV (Fonseca et al. 2000) e HTLV-1/2 (Porter et al. 1994, Elm et al. 1998, Mahieux et al. 2000).

Reações sorológicas cruzadas com *Plasmodium falciparum* têm sido identificadas em resultados soro indeterminado para os vírus T-linfotrópicos humanos tipo I (HTLV-1, do inglês *Human T-cell Lymphotropic Virus type 1*) e tipo II (HTLV-2, do inglês *Human T-cell Lymphotropic Virus type* 2), agente causador do Linfoma de células T e Paraparesia Espástica Tropical, em áreas onde a malária é endêmica. O *Plasmodium falciparum* é capaz de induzir anticorpos que reagem com proteínas do HTLV-1/2 e conduzir a resultados falso-positivo no ensaio imunoenzimático e padrões indeterminado no *Western blot* (Lal et al. 1994, Porter et al. 1994, Elm et al. 1998, Mahieux et al. 2000).

Neste mesmo sentido, infecções agudas de malária têm sido apontadas como fator causador de resultados falso-positivos em testes utilizando ensaio imunoenzimático para o HIV, uma vez que resultados mostram que a reatividade anti-HIV foi determinada por anticorpos anti-*P. falciparum* que apresentavam reatividade cruzada com antígenos do HIV-1 (Fonseca et al. 2000, Gasasira et al. 2006).

Resultados falsos positivos têm sido reportados em diagnósticos de mononucleose infecciosa ou infecção aguda primária causadas pelo vírus Epstein-

Barr (EBV, do inglês *Epstein-Barr Virus*) utilizando IgM específico, sendo que uma das possíveis causas para estes resultados tem sido atribuído à presença de uma possível reatividade cruzada de anticorpos IgM de *Toxoplasma gondii* e anticorpos EBV (Tranchand-Bunel et al. 1999)

8 CONCLUSÕES

Baseado nos dados apresentados neste trabalho é possível concluir que:

1. O algoritmo da ferramenta ProGeRF apresenta-se como uma ferramenta eficiente, rápida e precisa. Tomando como base os recursos listados por Mudunuri et al. (2010) na Tabela 1, ProGeRF apresenta: a) Interface web dinâmica e amigável; b) Identificação de repetições perfeitas e imperfeitas; c) Tamanho da repetição detectada de 2 a 7 bp. No entanto, como proteomas podem conter elementos repetitivos maiores que 7, foi disponibilizado uma versão para proteínas que permite encontrar elementos repetitivos de até 50 aa; d) Busca de repetição por tamanhos específicos; e) Visualização do alinhamento; f) informações sobre a sequência flanqueadora; g) Estatísticas da repetição; e h) Saída gráfica;
2. Dentre as ferramentas que localizam tanto repetições perfeitas e imperfeitas, ProGeRF é a única que proporciona visualização gráfica e buscas com filtros nos resultados. Outra vantagem desta ferramenta está na possibilidade de executá-la tanto em dados genômicos quanto em dados proteômicos e em grandes arquivos.
3. Levando em consideração a localização física no genoma/proteoma e a não sobreposição de elementos repetitivos, a Ferramenta ProGeRF consegue identificar uma quantidade maior de elementos repetitivos que as demais ferramentas em um tempo, consideravelmente, rápido.
4. A técnica utilizada na predição de epítopos lineares de células B, baseada em técnicas de aprendizado de máquina do tipo SVM, treinadas com dados de bactérias de maneira geral, conduz a uma predição, em média, próxima do aleatório. No entanto, o modelo treinado com dados de vírus para algumas espécies de protozoários apresentou bom grau de eficiência;
5. As predições em dados de protozoários realizado com modelos treinados com dados de protozoários conduzem a uma predição satisfatória, com elevadas medidas de avaliação.

6. Uma vez que ferramentas de predição de epítopos são treinadas, na sua grande maioria, com dados de epítopos de vírus e bactérias, estes podem apresentar reatividade cruzada com regiões antigênicas e imunogênicas de protozoários, percebe-se a necessidade de estudos que desenvolvam ferramentas de predição de epítopos integradas com marcadores moleculares capazes de identificar determinantes antigênicos espécie-específico;

9 PERSPECTIVAS

Dada a natureza do trabalho realizado no âmbito desta tese, as perspectivas de desenvolvimento que se apresentam são diversas. No que diz respeito a ferramenta de identificação em tandem, a seguir serão detalhadas as perspectivas:

- Implementação de uma versão do modo *stand-alone* em uma linguagem multiplataforma (tipo JAVA), que também permita disponibilizar uma interface gráfica;
- Uma vez que o tempo de geração de degenerações depende do tamanho do motivo e da porcentagem de degenerações e considerando que quando se trabalha com proteomas existem na literatura repetições com tamanho bem maiores, um eixo alvo de pesquisas é melhorar a ferramenta para identificar repetições de grandes motivos;

Com relação a predição *in silico* de epítopos lineares de células B, as perspectivas são:

- Uma vez que o perfil PSSM dos aminoácidos de epítopos tem sido reportada na literatura como poderosa propriedade na identificação de padrões de epítopos, torna-se importante desenvolver uma estratégia de extrair tal propriedade de forma eficiente e usá-la no processo de treino e predição de epítopos lineares de células B;
- Disponibilização de uma ferramenta *web* de predição de epítopos lineares de células B integrada com a ferramenta de identificação de regiões repetitivas de maneira que identifique determinantes antigênicos espécies-específicos;

10 REFERÊNCIAS BIBLIOGRÁFICAS

Alix AJ 1999. Predictive estimation of protein linear epitopes by using the program PEOPLE. *Vaccine* 18: 311–314.

Aurrecoechea C, Barreto A, Brestelli J, Brunk BP, Cade S, Doherty R, Fischer S, Gajria B, Gao X, Gingle A, Grant G, Harb OS, Heiges M, Hu S, Iodice J, Kissinger JC, Kraemer ET, Li W, Pinney DF, Pitts B, Roos DS, Srinivasamoorthy G, Stoeckert CJ, Wang H, Warrenfeltz S 2012. EuPathDB: The Eukaryotic Pathogen database. *Nucleic Acids Res.* 41: D684–D691.

Benson G 1999. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.* 27: 573–580.

Blythe MJ, Flower DR 2005. Benchmarking B cell epitope prediction: Underperformance of existing methods. *Protein Sci. Publ. Protein Soc.* 14: 246–248.

Burns JM, Shreffler WG, Rosman DE, Sleath PR, March CJ, Reed SG 1992. Identification and synthesis of a major conserved antigenic epitope of Trypanosoma cruzi. *Proc. Natl. Acad. Sci. U. S. A.* 89: 1239–1243.

Camacho C, Madden T, Ma N, Tao T, Agarwala R, Morgulis A 2013. BLAST Command Line Applications User Manual.

Carmona SJ, Sartor P, Leguizamón MS, Campetella O, Agüero F 2010. A computational pipeline for diagnostic biomarker discovery in the human pathogen Trypanosoma cruzi. *BMC Bioinformatics* 11: O11.

Carmona SJ, Sartor PA, Leguizamón MS, Campetella OE, Agüero F 2012. Diagnostic peptide discovery: prioritization of pathogen diagnostic markers using multiple features. *PloS One* 7: e50748.

Carnevale S, Velásquez JN, Portillo H del, Labbé JH, Cabrera MG, Ferella M, Andersson B, Guarnera EA, Angel SO 2004. Identification and characterization of an interspersed repetitive DNA fragment in Plasmodium vivax with potential use for specific parasite detection. *Exp. Parasitol.* 108: 81–88.

Castelo AT, Martins W, Gao GR 2002. TROLL—Tandem Repeat Occurrence Locator. *Bioinformatics* 18: 634–636.

Chang C-C, Lin C-J 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans Intell Syst Technol* 2: 27:1–27:27.

Chawla NV, Japkowicz N, Kotcz A 2004. Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explor Newsl* 6: 1–6.

Chen J, Liu H, Yang J, Chou K-C 2007. Prediction of linear B-cell epitopes using amino acid pair antigenicity scale. *Amino Acids* 33: 423–428.

Chou PY, Fasman GD 1978. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv. Enzymol. Relat. Areas Mol. Biol.* 47: 45–148.

Clayton C 2010. Repetitive elements in parasitic protozoa. *BMC Biol.* 8: 64.

Coutinho HDM, Neto VM, Verde LCL 2009. Técnicas com Marcadores Moleculares Usadas nas Ciências da Saúde. *Rev. Bras. Ciênc. Saúde* 10: 177–188.

Dame JB, Williams JL, McCutchan TF, Weber JL, Wirtz RA, Hockmeyer WT, Maloy WL, Haynes JD, Schneider I, Roberts D 1984. Structure of the gene encoding the immunodominant surface antigen on the sporozoite of the human malaria parasite Plasmodium falciparum. *Science* 225: 593–599.

DaRocha WD, Bartholomeu DC, Macêdo CDS, Horta MF, Cunha-Neto E, Donelson JE, Teixeira SMR 2002. Characterization of cDNA clones encoding ribonucleoprotein antigens expressed in Trypanosoma cruzi amastigotes. *Parasitol. Res.* 88: 292–300.

Davydov YI, Tonevitsky AG 2009. Prediction of linear B-cell epitopes. *Mol. Biol.* 43: 150–158.

Edvinsson B, Lappalainen M, Evengård B, ESCMID Study Group for Toxoplasmosis 2006. Real-time PCR targeting a 529-bp repeat element for diagnosis of toxoplasmosis. *Clin. Microbiol. Infect. Off. Publ. Eur. Soc. Clin. Microbiol. Infect. Dis.* 12: 131–136.

Ellegren H 2004. Microsatellites: simple sequences with complex evolution. *Nat. Rev. Genet.* 5: 435–445.

El-Manzalawy Y, Dobbs D, Honavar V 2008. Predicting linear B-cell epitopes using string kernels. *J. Mol. Recognit. JMR* 21: 243–255.

Elm J, Desowitz R, Diwan A 1998. Serological cross-reactivities between the retroviruses HIV and HTLV-1 and the malaria parasite Plasmodium falciparum. *P. N. G. Med. J.* 41: 15–22.

Emini EA, Hughes JV, Perlow DS, Boger J 1985. Induction of hepatitis A virus-neutralizing antibody by a virus-specific synthetic peptide. *J. Virol.* 55: 836–839.

Ersfeld K 2003. Genomes and genome projects of protozoan parasites. *Curr. Issues Mol. Biol.* 5: 61–74.

Fernandes GG 2009. *Interface Humano Computador: práticas pedagógicas para ambientes virtuais*. EDUFPI, Teresina-PI.

Ferreira ME 1998. *Introdução ao uso de marcadores moleculares em análise genética*. Embrapa Recursos genéticos e Biotecnologia.

Ferreira ME, Grattapaglia D 1995. *Introdução ao uso de marcadores RAPD e RFLP em análise genética*. Embrapa-cenargen.

Flower DR, Macdonald IK, Ramakrishnan K, Davies MN, Doytchinova IA 2010. Computer aided selection of candidate vaccine antigens. *Immunome Res.* 6 Suppl 2: S1.

Folgueira C, Martínez-Bonet M, Requena JM 2010. The Leishmania infantum PUF proteins are targets of the humoral response during visceral leishmaniasis. *BMC Res. Notes* 3: 13.

Fonseca MO, Pang L, Avila S do LM de, Arruk VG, Tozetto-Mendoza TR, Ferreira AW, Saes-Alquezar A, Boulos M 2000. Cross-reactivity of anti-

Plasmodium falciparum antibodies and HIV tests. *Trans. R. Soc. Trop. Med. Hyg.* 94: 171–172.

Gao J, Faraggi E, Zhou Y, Ruan J, Kurgan L 2012. BEST: Improved Prediction of B-Cell Epitopes from Antigen Sequences. *PLoS ONE* 7: e40104.

Gasasira AF, Dorsey G, Kamya MR, Havlir D, Kiggundu M, Rosenthal PJ, Charlebois ED 2006. False-Positive Results of Enzyme Immunoassays for Human Immunodeficiency Virus in Patients with Uncomplicated Malaria. *J. Clin. Microbiol.* 44: 3021–3024.

Giaco L, Amicosante M, Fraziano M, Gherardini PF, Ausiello G, Helmer-Citterich M, Colizzi V, Cabibbo A 2012. B-Pred, a structure based B-cell epitopes prediction server. *Adv. Appl. Bioinforma. Chem. AABC* 5: 11–21.

GOODRICH MT, TAMASSIA R *Projeto de algoritmos: Fundamentos, análise e exemplos da internet*. Bookman Companhia Ed.

Goodswen SJ, Kennedy PJ, Ellis JT 2013. A guide to in silico vaccine discovery for eukaryotic pathogens. *Brief. Bioinform.* 14: 753–774.

Goto Y, Carter D, Reed SG 2008. Immunological Dominance of Trypanosoma cruzi Tandem Repeat Proteins. *Infect. Immun.* 76: 3967–3974.

Goto Y, Coler RN, Guderian J, Mohamath R, Reed SG 2006. Cloning, Characterization, and Serodiagnostic Evaluation of Leishmania infantum Tandem Repeat Proteins. *Infect. Immun.* 74: 3939–3945.

Goto Y, Coler RN, Reed SG 2007. Bioinformatic Identification of Tandem Repeat Antigens of the Leishmania donovani Complex. *Infect. Immun.* 75: 846–851.

Goto Y, Duthie MS, Kawazu S-I, Inoue N, Carter D 2011. Biased cellular locations of tandem repeat antigens in African trypanosomes. *Biochem. Biophys. Res. Commun.* 405: 434–438.

Hernández-Hernández F de la C, Rodríguez MH 2009. Avances biotecnológicos en el diagnóstico de enfermedades infecciosas. *Salud Pública México* 51: s424–s438.

He Y, Xiang Z, Mobley HLT 2010. Vaxign: the first web-based vaccine design program for reverse vaccinology and applications for vaccine development. *J. Biomed. Biotechnol.* 2010: 297505.

Karplus PA, Schulz GE 1985. Prediction of chain flexibility in proteins. *Naturwissenschaften* 72: 212–213.

Kasper DC, Prusa AR, Hayde M, Gerstl N, Pollak A, Herkner KR, Reiter-Reisacher R 2009. Evaluation of the Vitros ECiQ Immunodiagnostic System for Detection of Anti-Toxoplasma Immunoglobulin G and Immunoglobulin M Antibodies for Confirmatory Testing for Acute Toxoplasma gondii Infection in Pregnant Women. *J. Clin. Microbiol.* 47: 164–167.

Kawashima S, Kanehisa M 2000. AAindex: Amino Acid index database. *Nucleic Acids Res.* 28: 374–374.

Kawashima S, Pokarowski P, Pokarowska M, Kolinski A, Katayama T, Kanehisa M 2008. AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res.* 36: D202–D205.

Kindt TJ, Goldsby RA, Osborne BA 2008. *Imunologia de Kubi*. Editora Bookman.

Koenen M, Scherf A, Mercereau O, Langsley G, Sibilli L, Dubois P, Pereira da Silva L, Müller-Hill B 1984. Human antisera detect a Plasmodium falciparum genomic clone encoding a nonapeptide repeat. *Nature* 311: 382–385.

Kofler R, Schlötterer C, Lelley T 2007. SciRoKo: a new tool for whole genome microsatellite search and investigation. *Bioinformatics* 23: 1683–1685.

Kolaskar AS, Tongaonkar PC 1990. A semi-empirical method for prediction of antigenic determinants on protein antigens. *FEBS Lett.* 276: 172–174.

Kolpakov R, Bana G, Kucherov G 2003. mreps: efficient and flexible detection of tandem repeats in DNA. *Nucleic Acids Res.* 31: 3672–3678.

Lal R, Rudolph D, Alpers M, Sulzer A, Yaping S, Lal A 1994. Immunological Cross-Reactivity Between Structural Proteins of Human T-Cell Lymphotropic Virus Type-I and the Blood-Stage of Plasmodium-Falciparum. *Clin. Diagn. Lab. Immunol.* 1: 5–10.

Larsen JEP, Lund O, Nielsen M 2006. Improved method for predicting linear B-cell epitopes. *Immunome Res.* 2: 2.

Leclercq S, Rivals E, Jarne P 2007. Detecting microsatellites within genomes: significant variation among algorithms. *BMC Bioinformatics* 8: 125.

LEISERSON CE, Cormen TH, RIVEST RL, STEIN C 2002. *Algoritmos: teoria e prática*. CAMPUS - RJ.

Lim KG, Kwoh CK, Hsu LY, Wirawan A 2013. Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance. *Brief. Bioinform.* 14: 67–81.

Lin SY, Cheng C-W, Su EC 2013. Prediction of B-cell epitopes using evolutionary information and propensity scales. *BMC Bioinformatics* 14: S10.

Lobo FP, Lopes R da S, Luiz GFR, Rodrigues TS, Fujiwara RT, Pinto FCF, Bartholomeu DC IMMUNORANK: a unique web tool to detect and rank potentially immunogenic peptides and proteins from primary sequence data.

Lopes R da S, Moraes WJL, Rodrigues T de S, Bartholomeu DC 2015. ProGeRF: Proteome and Genome Repeat Finder Utilizing a Fast Parallel Hash Function. *BioMed Res. Int.* 2015: e394157.

Lorena AC, Carvalho ACPLF de 2007. Uma Introdução às Support Vector Machines. *Rev. Informática Teórica E Apl.* 14: 43–67.

Mahieux R, Horal P, Mauclère P, Mercereau-Puijalon O, Guillotte M, Meertens L, Murphy E, Gessain A 2000. Human T-Cell Lymphotropic Virus Type 1 Gag Indeterminate Western Blot Patterns in Central Africa: Relationship toPlasmodium falciparum Infection. *J. Clin. Microbiol.* 38: 4049–4057.

Maia LC da, Palmieri DA, Souza VQ de, Kopp MM, Carvalho FI, F&#233, De L, Costa de Oliveira A 2008. SSR Locator: Tool for Simple Sequence Repeat

Discovery Integrated with Primer Design and PCR Simulation. *Int. J. Plant Genomics* 2008.

Manzalawy Y EL-, Dobbs D, Honavar V 2008. Predicting linear B-cell epitopes using string kernels. *J. Mol. Recognit.* 21: 243–255.

Mendes TA de O 2011. Identificação de epitopos lineares de célula B conservados e polimórficos entre diferentes cepas de Trypanosoma cruzi com potencial aplicação para sorodiagnóstico e sorotipagem.

Mesquita RT, Vidal JE, Pereira-Chioccola VL 2010. Molecular diagnosis of cerebral toxoplasmosis: comparing markers that determine Toxoplasma gondii by PCR in peripheral blood from HIV-infected patients. *Braz. J. Infect. Dis.* 14: 346–350.

MILACH SCK (Ed) 1998. *Marcadores moleculares em plantas*. S.C.K. Milach.

Monzote L, Siddiq A 2011. Drug Development to Protozoan Diseases. *Open Med. Chem. J.* 5: 1–3.

Mudunuri SB, Nagarajaram HA 2007. IMEx: Imperfect Microsatellite Extractor. *Bioinformatics* 23: 1181–1187.

Mudunuri SB, Rao AA, Pallamsetty S, Nagarajaram HA 2010. Comparative Analysis of Microsatellite Detecting Software: A Significant Variation in Results and Influence of Parameters. In: *Proc. Int. Symp. Biocomput.*, ISB '10. ACM, New York, NY, USA, pp. 38:1–38:7.

Ning Z, Cox AJ, Mullikin JC 2001. SSAHA: a fast search method for large DNA databases. *Genome Res.* 11: 1725–1729.

Odorico M, Pellequer J-L 2003. BEPITOPE: predicting the location of continuous epitopes and patterns in proteins. *J. Mol. Recognit. JMR* 16: 20–22.

Oliveira-Ferreira J, Vargas-Serrato E, Barnwell JW, Moreno A, Galinski MR 2004. Immunogenicity of Plasmodium vivax merozoite surface protein-9 recombinant proteins expressed in E. coli. *Vaccine* 22: 2023–2030.

Oliveira EJ, Pádua JG, Zucchi MI, Vencovsky R, Vieira MLC 2006. Origin, evolution and genome distribution of microsatellites. *Genet. Mol. Biol.* 29: 294–307.

Parker JM, Guo D, Hodges RS 1986. New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correlation of predicted surface residues with antigenicity and X-ray-derived accessible sites. *Biochemistry (Mosc.)* 25: 5425–5432.

Pellequer JL, Westhof E 1993. PREDITOP: a program for antigenicity prediction. *J. Mol. Graph.* 11: 204–210, 191–192.

Pellequer JL, Westhof E, Regenmortel MH Van 1991. Predicting location of continuous epitopes in proteins from their primary structures. *Methods Enzymol.* 203: 176–201.

Penna G de O, Teixeira MG, Pereira SM 1998. *Doenças infecciosas e parasitárias: Aspectos clínicos, vigilância epidemiológica e medidas de controle - GUIA DE BOLSO*. Brasília: Ministério da Saúde: Fundação Nacional de Saúde.

Peters B, Sidney J, Bourne P, Bui H-H, Buus S, Doh G, Fleri W, Kronenberg M, Kubo R, Lund O, Nemazee D, Ponomarenko JV, Sathiamurthy M, Schoenberger S, Stewart S, Surko P, Way S, Wilson S, Sette A 2005. The Immune Epitope Database and Analysis Resource: From Vision to Blueprint. *PLoS Biol.* 3.

Ponomarenko JV, Regenmortel MHV van 2009. B-Cell Epitope Prediction. In: *Struct. Bioinforma.*, John Wiley & Sons.

Porter KR, Liang L, Long GW, Bangs MJ, Anthony R, Andersen EM, Hayes CG 1994. Evidence for anti-Plasmodium falciparum antibodies that cross-react with human T-lymphotropic virus type I proteins in a population in Irian Jaya, Indonesia. *Clin. Diagn. Lab. Immunol.* 1: 11–15.

Rabello ALT 1990. O exame parasitológico de fezes, a biópsia retal e o teste imunoenzimático no diagnóstico da Esquistossomose mansoni humana.

Reneker J, Shyu C-R 2005. Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals. *BMC Bioinformatics* 6: 111.

Resende DM, Rezende AM, Oliveira NJ, Batista IC, Corrêa-Oliveira R, Reis AB, Ruiz JC 2012. An assessment on epitope prediction methods for protozoa genomes. *BMC Bioinformatics* 13: 309.

Rota ML, Kantety RV, Yu J-K, Sorrells ME 2005. Nonrandom distribution and frequencies of genomic and EST-derived microsatellite markers in rice, wheat, and barley. *BMC Genomics* 6: 23.

Saha S, Raghava GPS 2006. Prediction of continuous B-cell epitopes in an antigen using recurrent neural network. *Proteins* 65: 40–48.

Schlötterer C 2000. Evolutionary dynamics of microsatellite DNA. *Chromosoma* 109: 365–371.

Schlötterer C 2004. The evolution of molecular markers — just a matter of fashion? *Nat. Rev. Genet.* 5: 63–69.

Sears C, Rebert C, Mazovetskiy G, Fenkart H, Dorfman J, Thilo J, Otto M, Lauke PH, McDonald T 2015. Bootstrap · The world's most popular mobile-first and responsive front-end framework.

Sharma PC, Grover A, Kahl G 2007. Mining microsatellites in eukaryotic genomes. *Trends Biotechnol.* 25: 490–498.

Singh H, Ansari HR, Raghava GPS 2013. Improved Method for Linear B-Cell Epitope Prediction Using Antigen's Primary Sequence. *PLoS ONE* 8: e62216.

Skinner ME, Uzilov AV, Stein LD, Mungall CJ, Holmes IH 2009. JBrowse: A next-generation genome browser. *Genome Res.* 19: 1630–1638.

Sollner J, Grohmann R, Rapberger R, Perco P, Lukas A, Mayer B 2008. Analysis and prediction of protective continuous B-cell epitopes on pathogen proteins. *Immunome Res.* 4: 1.

Tavares RG, Staggemeier R, Borges ALP, Rodrigues MT, Castelan LA, Vasconcelos J, Anschau ME, Spalding SM 2011. Molecular techniques for

the study and diagnosis of parasite infection. *J. Venom. Anim. Toxins Trop. Dis.* 17: 239–248.

Thiel T, Michalek W, Varshney RK, Graner A 2003. Exploiting EST databases for the development and characterization of gene-derived SSR-markers in barley (Hordeum vulgare L.). *TAG Theor. Appl. Genet. Theor. Angew. Genet.* 106: 411–422.

Tomov T 2008. jq{Grid} - Introduction.

Toseland CP, Clayton DJ, McSparron H, Hemsley SL, Blythe MJ, Paine K, Doytchinova IA, Guan P, Hattotuwagama CK, Flower DR 2005. AntiJen: a quantitative immunology database integrating functional, thermodynamic, kinetic, biophysical, and cellular data. *Immunome Res.* 1: 4.

Tranchand-Bunel D, Gras-Masse H, Bourez B, Dedecker L, Auriault C 1999. Evaluation of an Epstein-Barr Virus (EBV) Immunoglobulin M Enzyme-Linked Immunosorbent Assay Using a Synthetic Convergent Peptide Library, or Mixotope, for Diagnosis of Primary EBV Infection. *J. Clin. Microbiol.* 37: 2366–2368.

Vexenat A de C, Santana JM, Teixeira ARL 1996. Cross-reactivity of antibodies in human infections by the kinetoplastid protozoa Trypanosoma cruzi, Leishmania chagasi and Leishmania (Viannia) braziliensis. *Rev. Inst. Med. Trop. São Paulo* 38: 177–185.

Vita R, Zarebski L, Greenbaum JA, Emami H, Hoof I, Salimi N, Damle R, Sette A, Peters B 2010. The Immune Epitope Database 2.0. *Nucleic Acids Res.* 38: D854–D862.

Wang H-W, Lin Y-C, Pai T-W, Chang H-T 2011. Prediction of B-cell Linear Epitopes with a Combination of Support Vector Machine Classification and Amino Acid Propensity Identification. *BioMed Res. Int.* 2011.

Wang X, Lu P, Luo Z 2013. GMATo: A novel tool for the identification and analysis of microsatellites in large genomes. *Bioinformation* 9: 541–544.

Wee LJK, Simarmata D, Kam Y-W, Ng LFP, Tong JC 2010a. SVM-based prediction of linear B-cell epitopes using Bayes Feature Extraction. *BMC Genomics* 11 Suppl 4: S21.

Wee LJ, Simarmata D, Kam Y-W, Ng LF, Tong JC 2010b. SVM-based prediction of linear B-cell epitopes using Bayes Feature Extraction. *BMC Genomics* 11: S21.

Wickstead B, Ersfeld K, Gull K 2003. Repetitive Elements in Genomes of Parasitic Protozoa. *Microbiol. Mol. Biol. Rev.* 67: 360–375.

Wirawan A, Kwoh CK, Hsu LY, Koh TH 2010. INVERTER: INtegrated Variable numbER Tandem rEpeat findeR. In: Chan JH, Ong Y-S, Cho S-B (Eds.), *Comput. Syst.-Biol. Bioinforma.*, Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 151–164.

Yang X, Yu X 2009. An introduction to epitope prediction methods and software. *Rev. Med. Virol.* 19: 77–96.

Yao B, Zhang L, Liang S, Zhang C 2012. SVMTriP: a method to predict antigenic epitopes using support vector machine to integrate tri-peptide similarity and propensity. *PloS One* 7: e45152.

Zhang Q, Wang P, Kim Y, Haste-Andersen P, Beaver J, Bourne PE, Bui H-H, Buus S, Frankild S, Greenbaum J, Lund O, Lundegaard C, Nielsen M, Ponomarenko J, Sette A, Zhu Z, Peters B 2008. Immune epitope database analysis resource (IEDB-AR). *Nucleic Acids Res.* 36: W513–W518.

Zhao L, Li J 2010. Mining for the antibody-antigen interacting associations that predict the B cell epitopes. *BMC Struct. Biol.* 10: S6.

ANEXO

Hindawi Publishing Corporation
BioMed Research International
Volume 2015, Article ID 394157, 9 pages
http://dx.doi.org/10.1155/2015/394157

Research Article

ProGeRF: Proteome and Genome Repeat Finder Utilizing a Fast Parallel Hash Function

Robson da Silva Lopes,[1] Walas Jhony Lopes Moraes,[1] Thiago de Souza Rodrigues,[2] and Daniella Castanheira Bartholomeu[3]

[1]*Department of Computer Science, Federal University of Mato Grosso, 78600-000 Barra do Garcas, MT, Brazil*
[2]*Federal Center of Technological Education of Minas Gerais, Belo Horizonte, MG, Brazil*
[3]*Department of Parasitology, Federal University of Minas Gerais, 31270-829 Belo Horizonte, MG, Brazil*

Correspondence should be addressed to Robson da Silva Lopes; robsonsilvalopes@hotmail.com

Received 3 June 2014; Revised 19 January 2015; Accepted 31 January 2015

Academic Editor: Satoru Miyano

Repetitive element sequences are adjacent, repeating patterns, also called motifs, and can be of different lengths; repetitions can involve their exact or approximate copies. They have been widely used as molecular markers in population biology. Given the sizes of sequenced genomes, various bioinformatics tools have been developed for the extraction of repetitive elements from DNA sequences. However, currently available tools do not provide options for identifying repetitive elements in the genome or proteome, displaying a user-friendly web interface, and performing-exhaustive searches. ProGeRF is a web site for extracting repetitive regions from genome and proteome sequences. It was designed to be efficient, fast, and accurate and primarily user-friendly web tool allowing many ways to view and analyse the results. ProGeRF (Proteome and Genome Repeat Finder) is freely available as a stand-alone program, from which the users can download the source code, and as a web tool. It was developed using the hash table approach to extract perfect and imperfect repetitive regions in a (multi)FASTA file, while allowing a linear time complexity.

1. Introduction

Repetitive elements are found in large quantities in eukaryotic genome, both in coding and noncoding region, and also in intergenic regions of prokaryotes [1]. In humans repetitive elements represent approximately 7% of the genome [2]; in parasites protists the proportion of repetitions varies from 11% to 65% of the DNA [3] while in protozoa such as *Theileria parva*, *Plasmodium berghei*, *T. cruzi,* and *Toxoplasma gondii*, this value varies between 4% and 30% of repeating sequences in genomes [4, PMID: 16020725].

Repetitive sequences can be categorized into two groups: interspersed repeats and tandem DNA repeat. Interspersed repeats are mainly active or inactive copies of transposable elements dispersed throughout the genome and are divided into DNA transposons and retrotransposons [5], while the tandem repeats are ribosomal DNA sequences and satellite DNA [4, 6].

Normally, tandem repeats are classified according to the repetitive motifs length in microsatellites, minisatellites, and macrosatellites. Microsatellites (also known as short tandem repeats (STRs) or simple sequence repeats (SSRs)) are small stretches of DNA sequences (usually <200 bp), with motif lengths between 1 and 6 bp. Minisatellites are large repetitive sequences, with motif lengths of 5 to 25 bp, and the macrosatellites are large regions of repeats with lengths larger than 25 bp [4, 7, 8].

Microsatellites can be classified as perfect, imperfect, and compound. Perfect repetitive elements are formed from identical repetitive units. Imperfect repetitive elements are units with small mutations and may have been caused by insertions, deletions, or replacements. Repetitive compounds elements are composed of sequences in which two or more repetitions (perfect or imperfect) are arranged successively with or without nucleotide bases between them [8].

Repetitive elements, mainly microsatellites, have been widely used as molecular markers in phylogenetic studies, analyses of genetic populations, construction of genetic maps, paternity testing, and forensic medicine [7, 9]. The main explanation given for the emergence of variation in the amount of repetitions is a sliding model (slippage) of DNA polymerase during DNA replication [10].

Given the importance of identifying repeating regions and the possibility of identifying them in silico, many tools for identifying repeating regions have been developed. Work carried out by Lim et al. [2], Mudunuri et al. [8], and Leclercq et al. [1] reviewed and tested the main tools for identifying repeats. The following are the most commonly used tools for extracting repeat regions of a genome: TRF [11], TROLL [12], Misa [13], Mreps [14], SciRoKo [15], Sputnik [16], SSR Locator [17], IMEX [18], and GMATo [19].

However, it should be taken into consideration that all of the above software tools are unable to obtain all of the possible sequences because they (a) locate only perfect repetitions (GMATo, TROLL, and Misa); (b) make use of probabilistic or statistical patterns heuristics that do not meet all possible repetitions (TRF and Mreps); and (c) are unable to execute on large FASTA files (SciRoKo and Mreps). Finally, none of these tools can be executed in both DNA and protein datasets.

Thus, this paper presents a fast and efficient algorithm inspired by the concepts of "Sequence Search and Alignment by Hashing Algorithm," SSAHA [20], that stores information about the locations of DNA words into a hash table and based on circular doubly linked lists for a fast and exhaustive identification of repetitive elements, both perfect and imperfect, in large DNA or protein FASTA files.

2. Methods

2.1. Definitions. Some definitions are presented below to facilitate understanding.

Sliding Window Method. To identify a given full-length DNA or protein sequence, the sliding window approach is employed to obtain sequences with variable length, where Q represents the sequence obtained for a sliding window and is called a DNA or amino acid word and $|Q|$ is word length.

Hash Table. This consists of an array where the data to be searched is stored and is accessed via a special index called a *key*. In our case, we store information about each motif. Hash table is allocated dynamically for each motif and there are $r^{|Q|}$ positions, where r is the radix (four for DNA and twenty for amino acids), and $|Q|$ is the length of the word (which in our case is the sliding window length). With this, the hash table can have a position for each combination of nucleotides or amino acids of size $|Q|$.

Hash Function. A hash function that maps DNA or amino acids to digits is based on the [21] conversion, where a hash function m is defined as a function that maps each DNA base or amino acid into digits, which in turn corresponds to a position (index) in the hash table. For DNA, each nucleotide is mapped as $m(\mathrm{A}) = 0$, $m(\mathrm{C}) = 1$, $m(\mathrm{G}) = 2$, and $m(\mathrm{T}) = 3$ and for amino acid residue it is mapped as $m(\mathrm{G}) = 0$, $m(\mathrm{P}) = 1$, $m(\mathrm{A}) = 2$, $m(\mathrm{V}) = 3$, $m(\mathrm{L}) = 4$, $m(\mathrm{I}) = 5$, $m(\mathrm{M}) = 6$, $m(\mathrm{C}) = 7$, $m(\mathrm{F}) = 8$, $m(\mathrm{Y}) = 9$, $m(\mathrm{W}) = 10$, $m(\mathrm{H}) = 11$, $m(\mathrm{K}) = 12$, $m(\mathrm{R}) = 13$, $m(\mathrm{Q}) = 14$, $m(\mathrm{N}) = 15$, $m(\mathrm{E}) = 16$, $m(\mathrm{D}) = 17$, $m(\mathrm{S}) = 18$, and $m(\mathrm{T}) = 19$. The DNA or amino acid word (sliding window) is converted into a number applying the general positional number system conversion function $h()$ to $Q_p = \{q_0 q_1 q_2 \cdots q_{|Q|-1}\}$, where $h()$ is defined by

$$h\left(Q_p\right) = \sum_{i=0}^{|Q|-1} m\left(q_i\right) r^{(|Q|-1)-i}. \quad (1)$$

Here Q is a DNA or amino acid word, m is the hash function, q is one base of word, p is the DNA or amino acid word start position on the sequence, r is the radix (four for DNA and twenty for amino acids), and $|Q|$ is the length of the word (which in our case is the sliding window length). For instance, the DNA word ACTGC is $(0 * 4^4) + (1 * 4^3) + (3 * 4^2) + (2 * 4^1) + (1 * 4^0) = 121$.

Single Bucket. It consists of a 5-tuple, in which the information of each repetitive pattern for a given motif is recorded. It is formed by $\langle sp, fp, mt, g, lt\rangle$, where sp and fp are the initial and final positions of the repetitive pattern, respectively, mt is the repetitive motif, g is the amount of gaps within the repetitive sequence, and lt is the number of repetitions of the motif mt inside of this substring. Each index k of the repetitive elements hash table contains a list of single buckets, where every single bucket represents a repetitive sequence of motifs mapped to the value k. A circular doubly linked list has been utilized to implement the list of single bucket lists, thus ensuring the insertion and deletion of a bucket quickly.

2.2. Architecture. ProGeRF is available in two execution modes, as illustrated in Figure 1: as a stand-alone program, from which the users can download the source code, compile, and run in their machine in a Linux environment and as a web tool available at the web address http://64.79.105.19/ligp/. At this address, it is also possible to download the stand-alone version.

Repeat extraction module has been used in this two execution modes. This module consists of three algorithms: one developed in Perl and two developed in C language. The perfect and imperfect repetitions are identified by algorithms in C language, called RepeatFinderDNA and RepeatFinderProteome. The first algorithm works on a FASTA file with DNA sequences and the second algorithm works on a FASTA file with amino acids sequences.

The Perl script, called ProGeRF, receives the input parameters, performs the call to the RepeatFinder algorithms, and after treating overlaps calculates statistics and generates the output file.

2.3. Algorithms. The ProGeRF algorithm receives as input parameters (a) a (multi)FASTA file, (b) minimum size of the repetitive pattern, (c) the minimum and maximum sizes of the motif (word DNA or amino acids length or sliding window length), (d) maximum amount of gaps accepted

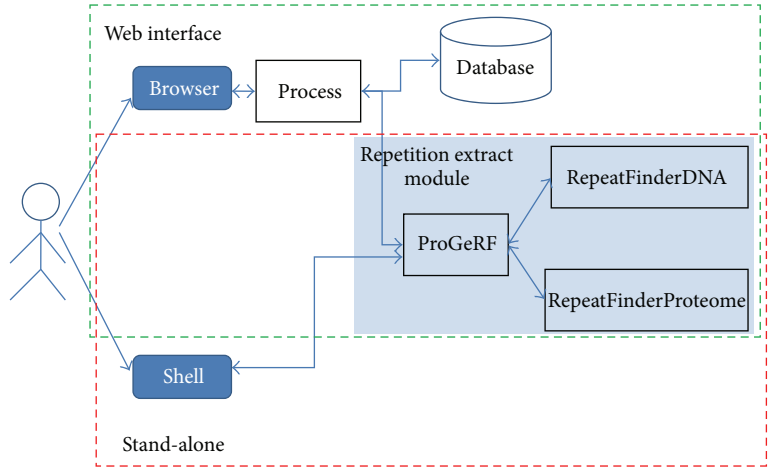


Figure 1: ProGeRF architecture. Structure of the tool both for the web environment and for the stand-alone mode. The dark blue rectangles with rounded corners represent interfaces with the system. The transparent rectangles with a blue background represent algorithms done in C or Perl. The process script receives data from the web environment, treats the data, saves them in a MySql database, and calls the repetition extract module.

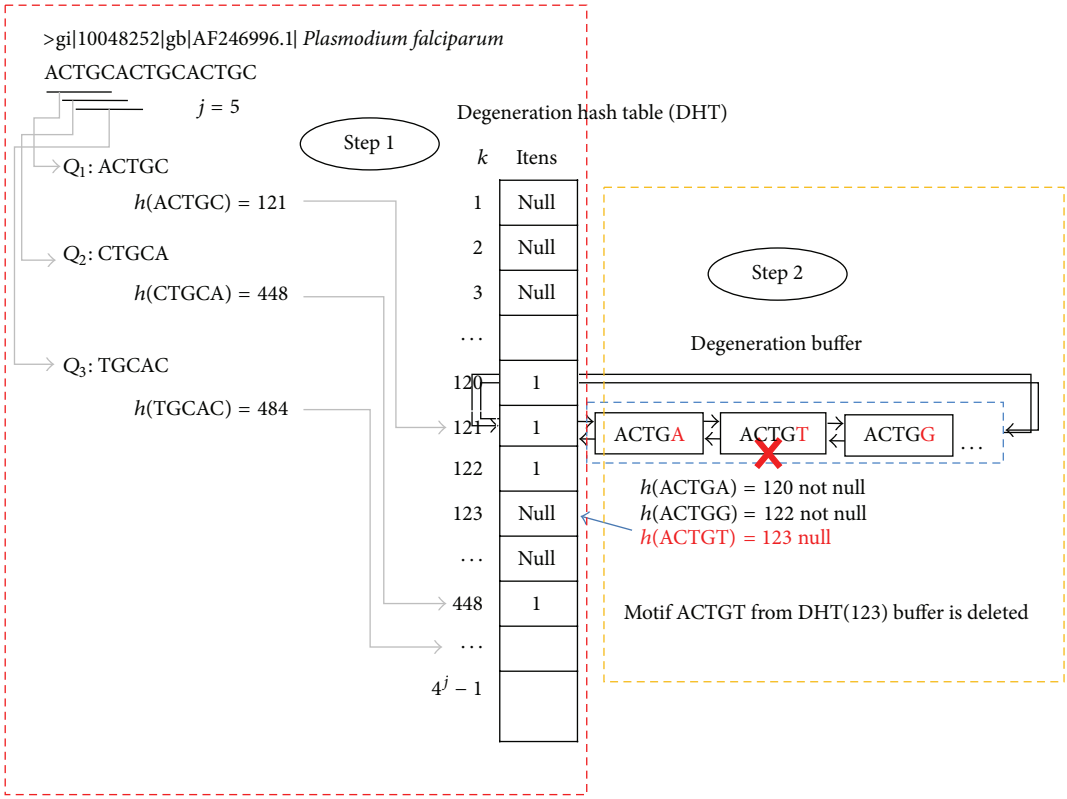


Figure 2: Creating degeneration hash table: Step 1: sliding window maps each motif of the sequence for a position in the degeneration hash table and sets value 1 to mapped position. Step 2: generate possible degeneration of the sliding window and store in the buffer at position k of the sliding window; only the degeneration that mapped to a position of the hash table presents a value of 1.

between each motif of a repeat, (e) percentage of maximum degeneration accepted for a motif, (f) overlapping percentage, and (g) run mode, that is, whether using a FASTA file of DNA or of amino acids.

RepeatFinder procedure executes, in parallel, for each motif size within the range of minimum and maximum values, to identify sequences with all motif lengths in this range of values in the FASTA file.

An overview of the ProGeRF algorithm is as shown in Figure 2.

(1) Dynamically allocate two hash tables (repetitive element hash table and degeneration hash table) of

radix$^{|Q|}$ positions, where radix is four for DNA and twenty for amino acids and $|Q|$ motif length. Each position in the tables is mapped to a unique combination of nucleotide/amino acids of length $|Q|$.

(2) Read the first sequence from FASTA file.

(3) Creating degeneration hash table *(DHT)*: for each sliding window Q_p, along the first sequence, where $p = 1, 2, 3, \ldots, n - j + 1$, n is the sequence length and j is the sliding window size ($j = |Q|$). RepeatFinder procedure converts each Q_p to an integer key k, as previously discussed. With this, the position k of DHT is set to 1; this process is illustrated by Figure 2, Step 1.

(4) Repeat the previous process for $p = 1, 2, 3, \ldots, n - j + 1$, where n is the sequence length and j is the sliding window size.

(5) For each position marked with 1 in the degeneration hash table, run the generating degeneration procedure. This procedure generates all possible degenerations of a motif up to a maximum percentage of defined imperfection. Each motif degenerate generated is converted to an integer key k', and if at position k of the degeneration hash table is marked as 1 the motif degenerate generated is inserted into a degeneration buffer. Otherwise, if position k' is marked as null, the motif is not inserted in the degeneration buffer, Figure 2, Step 2.

(6) Creating repetitive elements hash table (REHT), illustrated by Figure 3: for each sliding window Q_p, where $p = 1, 2, 3, \ldots, n - j + 1$, do the following:

  (a) calculate $k = h(Q_p)$;

  (b) Step 1: if a single bucket does not exist at position k of REHT then create a single bucket and set sp and fp with the initial and final positions of the motif Q_p. However, if there is a bucket and $0 < p - fp \leq ga$ (where p is sliding window position, fp the value registered in the bucket final position, and ga the maximum gap allowed between motif), then

    (i) set $fp = p + |Q| - 1$, that is, the final position of the current sliding window;

    (ii) increase the field lt of bucket;

    (iii) set $gp = gp + p - fp$; that is, record the total number of gaps.

    However, if there is a single bucket and the condition $0 < p - fp \leq ga$ is not satisfied and if the bucket field lt is not greater than or equal to the minimum amount of repetitions defined, then the last single bucket is deleted.

  (c) Step 2: check whether the current sliding window is a degeneration of some motif ever recorded in buckets of REHT. For this, degeneration buffer at position k of degeneration hash table is traversed.

  (d) Step 3: for each existing degeneration in the buffer, the function $h()$ is applied and then converted into an integer k, after which Step 1 is performed. However, the single bucket is not deleted if the condition $0 < p - fp \leq ga$ is not satisfied.

(7) Save the REHT results in a file and later erase its data.

(8) Repeat steps 1–7 for the other sequences in the FASTA file.

(9) In dealing with overlaps, join all the files from step 7 into a single file, sort the rows by the initial position of the repetition and for each row that represents a repetitive element, and check the following:

  (a) if the current repetitive element has an initial position less than the final position of the previous repetitive element then compute the degree of overlap;

  (b) if the degree of overlap is within the permitted value, skip to the next repetition. Otherwise, delete the smallest repetitive element and pass on to the next line.

(10) Print the remaining reps in the file.

2.4. Implementation. Hash tables were developed to perform a dynamic allocation of memory which allows the program to read FASTA files of any size. Furthermore, degeneration buffer and the buckets were implemented through circular doubly linked lists, which allow you to insert or remove degenerations or single buckets in the hash table quickly, without the need to scroll through the whole list.

Time complexity to create the degeneration hash table is approximately linear in function of the number of nucleotides or amino acids sequence, because the algorithm runs once the input sequence to identify the existing motif, scoring with 1 the position in the degeneration hash table of motif found. Then, it traverses the degeneration hash table, and at positions marked with 1, the possible degenerations are generated for the corresponding motif.

The algorithm accepts a maximum of 35% degeneration, that is, at most two degenerate characters in a motif of size 7. The amount of possible combinations for a motif of size j with degeneration by up to 2 characters is given by

$$c = \frac{1}{2}\left((r-1)^2 j^2 + (r-1)\, j\right), \tag{2}$$

where r is the radix (four for DNA and twenty for amino acids) and $j = |Q|$, that is, the length of the word (in our case it is the sliding window length). Because c does not vary with the size of the input sequence, it can be considered constant, so the time complexity to generate the degeneration hash table is of the order $O(n)$.

The step of generating REHT also presents linear time complexity depending on the size of the sequence input. Because the sliding window traverses the FASTA sequence once for every sliding window, the corresponding motif is inserted or deleted in the bucket in constant time and then tested at most c possible degenerations, and as c can

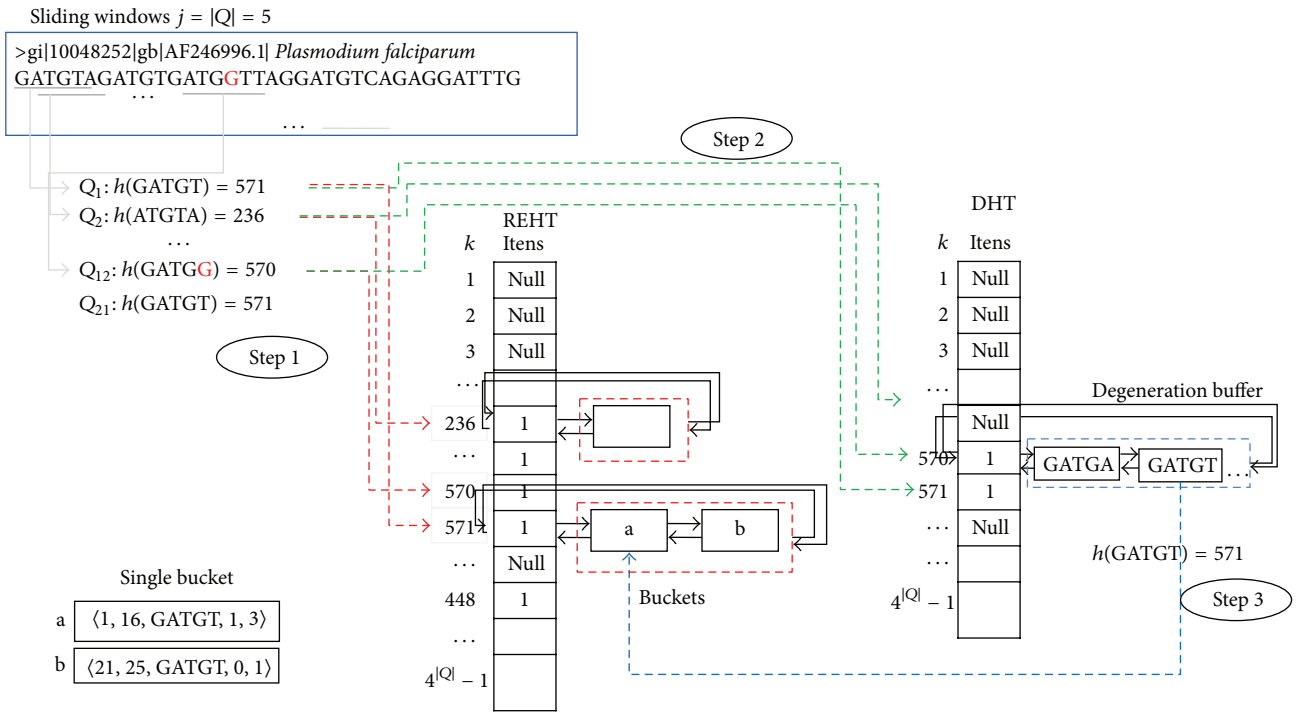


FIGURE 3: Creating repetitive element hash table: Step 1: sliding window maps each motif of the sequence for a position in the repetitive element hash table and sets value 1 to mapped position, and add or remove the sliding window to single bucket; Step 2: check whether the current sliding window is a degeneration of some motif ever recorded in buckets of REHT; Step 3: for each existing degeneration in the buffer the function $h()$ is applied and then converted into an integer k and, soon after, Step 1 is performed.

be considered constant, we have time complexity in order $O(n)$. Therefore, the RepeatFinder procedure presents time complexity of the order $O(n)$.

2.5. Interface and Output. ProGeRF is designed to have web and command line interface. The command line interaction may be performed by indicating the (multi)FASTA file address containing DNA or amino acids sequence(s), the motif length range, the minimum repeated times for all motif lengths or the minimum repeated time for each one, the maximum gaps allowed between motifs, the maximum degeneration percentage, the motif shifting percentage, and the run mode that defines DNA or amino acids input sequence and the output file name.

For example, the command sequence perl progerf.pl –q Linfantum_JPCM 5.FASTA –o output –i 2 –y 6 –r 5 –g 3 –v 0 –d 20 –m n will search repetitive elements in the file Linfantum_JPCM5.FASTA of motif with length range between 2 and 6, with maximum gaps of 3, motif overlap of 0%, degeneration of 20%, and run mode of nucleotide, and the result will be saved in the output file.

The results file presents a table, wherein each column represents the following information in order: sequence ID, size of the DNA/protein sequence, minimal repetitions allowed, repetition amount, repetitive element start and final position, number of gaps, statistics (only nucleotide run mode), and complete repetitive element.

The web mode, available at http://64.79.105.19/ligp/, offers a user-friendly interface developed using bootstrap packages for layout formatting, a JBrowse plugin [22] and jqGrid [23]. Web interface provides the same flexibility as command line mode. However, it is platform independent and can be run in any browser; the parameter setting is performed through forms, buttons, text boxes, and a combo box.

Web interface provides three ways of sending the FASTA file containing DNA or amino acid sequences.

(1) File upload: the users can send a FASTA file from their own computer.

(2) Copy and paste sequence: the user copies a sequence of interest and pastes in the text box.

(3) Automatic download from the NCBI data base: the user enters one or more GI numbers separated by commas, and the tools will download the sequences from the NCBI data base and run the repetition extraction algorithm. GI number (GenInfo identifier) is a unique number that identifies a particular sequence in the NCBI databases.

The results on the web page can be viewed in two ways: tabular format using the jqGrid script and graphical format, through the JBrowse plugin [22].

jqGrid is an Ajax-enabled JavaScript control that provides solutions for representing and manipulating tabular data on the web dynamically [23]. With jqGrid, the user can make

Table 1: Comparison of amount detection and execution times (in seconds) of Mreps, Misa, Sputnik, GMATo, SciRoKo, TRF, and ProGeRF. The features were run on a Dell Inspiron, Intel core 2 duo 2.2 GHz processor with 2 MB cache, 3 GB RAM, 320 GB hard drive, Ubuntu operating system 14.04 LTS 32 bits.

Sequence	Mreps	Misa	Sputnik	GMATo	SciRoKo	TRF	ProGeRF
	Rep (time)	Rep (time)	Rep (time)	Rep (time)	Rep (time)	Rep (time)	Rep (time)
NC_004318.1 (1204 kb)	9608 (2.8)	22867 (3.2)	7420 (0.7)	23539 (10.3)	3763 (1.1)	30244 (72.4)	26164 (3.9)
NC_001136.8 (1531 kb)	935 (1.4)	10640 (3.3)	1427 (0.9)	10721 (7.7)	185 (0.7)	8101 (4.5)	11552 (2.4)
NC_000962.2 (4411 kb)	1412 (3.9)	6832 (8.9)	3140 (1.46)	6846 (12.1)	72 (1.5)	19496 (24.5)	11422 (4.0)
Setaria[*] (515 Mb)	— (—)	2054241 (868.3)	480644 (105.7)	2073643 (9859.1)	47770 (129.0)	2438036 (1481.5)	2319812 (1352.0)

[*] Whole genome. The value in brackets is the runtimes in seconds.

queries for a particular motif pattern, setting several query filters and sorting the results by any of the columns.

JBrowse is a browser for genome viewing, developed in JavaScript, in which the user can navigate through the genome annotations on the web. In JBrowse, it is possible to zoom, navigate, and select range of subsequence within a genome [22].

3. Results and Discussion

We present two experiments in this paper. The first experiment demonstrates the efficiency of ProGeRF compared with other microsatellite identification tools, and the second experiment demonstrates the use of the repetitive element identification algorithm in protein FASTAs files.

Our current implementation features a Dell Inspiron, Intel core 2 duo 2.2 GHz processor with 2 MB cache, 3 GB RAM, 320 GB hard drive, and the Ubuntu operating system 14.04 LTS 32 bits.

For the first experiment, the tools Misa [13], Mreps [14], GMATo [19], SciRoKo [15], Sputnik [16], TRF [11], and ProGeRF were executed on each of the following genomic sequences: *Plasmodium falciparum* chromosome IV (NC_004318.1), *Saccharomyces cerevisiae* chromosome IV (NC_001136.8), *Mycobacterium tuberculosis* H37Rv genome (NC_000962.2) used in the work of Mudunuri and Nagarajaram [18] downloaded from ftp://ftp.ncbi.nih.gov/genomes, and the whole *Setaria italica* genome used in the work of Wang et al. [19], download from phytozome http://www.phytozome.net/.

For tools that allow for configuring the parameters minimum size, maximum size, and a minimum number of repetitions of five motifs, the values set for these parameters were 1, 6 and 5, respectively. For the remaining parameters, the following values were used according each tool: (a) Misa: maximum difference between 2 SSRs of 0; (b) Mreps: a resolution of 5; (c) SciRoKo: mode mismatched fixed penalty, with other parameters' score using default values; (d) Sputnik: a maximum size of 5 (maximum allowed by the tool), a minimal score: 5, maximal recursion: 0, minimum length of SSR to report: 10, and points for a mismatch and points for a match: 1; (e) TRF: matching weight: 2, mismatching penalty: 7, indel penalty: 7, match probability: 80, indel probability: 10, Minscore: 2, and MaxPeriod: 15; and (f) ProGeRF: a maximum number of gaps allowed 1, overlap of 0%, a degeneration of 20, and nucleotide mode.

IMEX tool presented error during the execution of the versions 1.0 and 2.0 in Ubuntu operating system 14.04 of 32 bits; thus it has not been possible to compare the results of this tool. In the first three sequences, Table 1, ProGeRF was a little slower than SciRoKo, Sputnik, Misa, and Mreps. However, the time can still be considered good, if we note the much larger number of repetitions tracked than the other tools. The number of repetitive elements of tools SciRoKo, Sputnik, and Mreps are smaller than of tools Misa, GMATo, TRF, and ProGeRF, but GMATo is slower than Misa, TRF, and ProGeRF. It is important to mention that GMATo tool is nonspecific in its treatment of overlaps and Wang et al. [19] relate that the extra loci from Misa are mined redundantly in the overlapped microsatellites.

The smaller numbers of repetitive elements found by tools SciRoKo, Sputnik, and Mreps are due to the fact that (a) Sputnik does not report hexanucleotide since maximum allowed is pentanucleotide; (b) according Mudunuri et al. [8] score based tools as SciRoKo and Sputnik that use higher mismatch penalties (such as 5, 6, and 7) and less match weights (such as 1, 2) fail to identify many smaller microsatellites (mono-tri); and (c) Mreps is highly constrained by its internal minimum size threshold, since detection starts at 11 bp for dinucleotides, 12 bp for trinucleotides, and up to 15 bp for hexanucleotides [14, 15, 18].

For three files, Table 1, a smaller number of repetitive elements has been identified by ProGeRF compared with the TRF tool, approximately 118 thousand differences in number. However, the TRF tool allows the occurrence of overlap where the redundancy is, at most, three pattern sizes and therefore presents a much larger number of repetitions than ProGeRF.

By default, ProGeRF does not allow overlaps and chooses the biggest repetitive elements sequence. However, the user can define the overlap percentage allowed, through the parameter −v. Nevertheless, the runtime of ProGeRF was lower than TRF and 7 times smaller than that of GMATo.

We evaluated whether the detections returned by tools on sequences NC_004318.1, NC_001136.8, and NC_000962.2, Table 1, occur at the same physical locations in genomes. More than 75% of SciRoKo, Sputnik, and Mreps detections are also detected by ProGeRF on the three sequences and GMATo and Misa detections are full coverage by ProGeRF on the three sequences, Table 2.

Sputnik and TRF present low amount of loci covered by ProGeRF on sequences NC_001136.8 and NC_000962.2. This

Table 2: Loci and nucleotide coverage between tools.

Sequence			B						
		Tools	Mreps	Misa	Sputnik	GMATo	SciRoKo	TRF	ProGeRF
Plasmodium Chr4 NC_004318.1	A	Mreps	—	78 (45)	53 (33)	78 (45)	41 (36)	98 (74)	89 (60)
		Misa	47 (63)	—	21 (4)	100 (99)	18 (40)	88 (79)	100 (98)
		Sputnik	91 (86)	70 (74)	—	70 (74)	58 (69)	0 (0)	83 (84)
		GMATo	49 (62)	100 (99)	21 (40)	—	19 (40)	89 (79)	100 (98)
		SciRoKo	96 (95)	93 (76)	95 (71)	92 (76)	—	95 (96)	98 (91)
		TRF	51 (41)	54 (32)	0 (0)	54 (32)	16 (21)	—	68 (46)
		ProGeRF	46 (56)	86 (67)	21 (31)	86 (67)	16 (32)	87 (78)	—
SAC Chr4 NC_001136.8	A	Mreps	—	60 (40)	42 (26)	68 (40)	18 (20)	95 (74)	86 (62)
		Misa	7 (12)	—	3 (7)	100 (99)	1 (4)	33 (37)	100 (99)
		Sputnik	30 (35)	29 (30)	—	29 (30)	12 (1)	74 (73)	38 (39)
		GMATo	7 (12)	100 (99)	3 (7)	—	1 (4)	33 (37)	100 (99)
		SciRoKo	91 (89)	77 (61)	86 (59)	77 (61)	—	99 (99)	94 (72)
		TRF	15 (16)	41 (26)	13 (12)	41 (26)	2 (5)	—	48 (35)
		ProGeRF	8 (16)	92 (80)	4 (7)	92 (80)	1 (5)	34 (39)	—
MTB H37Rv NC_000962.2	A	Mreps	—	9 (3)	15 (7)	9 (3)	3 (3)	91 (71)	75 (58)
		Misa	2 (3)	—	1 (1)	100 (99)	0.5 (1)	13 (13)	100 (99)
		Sputnik	6 (7)	2 (2)	—	2 (2)	1 (1)	66 (64)	14 (14)
		GMATo	2 (3)	100 (99)	1 (1)	—	0.5 (1)	13 (13)	100 (99)
		SciRoKo	73 (74)	47 (36)	63 (41)	47 (36)	—	100 (100)	79 (72)
		TRF	8 (7)	4 (1)	10 (7)	4 (1)	0.4 (0.5)	—	18 (13)
		ProGeRF	9 (16)	60 (35)	4 (4)	60 (35)	0.5 (1)	29 (33)	—

Percentage of the total number of detections (perfect and imperfect) of tools A also detected (i.e., covered) by tools B. The value in brackets is the proportion of nucleotides detected by A and covered by B.

low coverage is consequence of the lack of a parameter to set maximum size and minimum number of repetitions, which allows them to find a larger number of repetitive elements.

Therefore, we filter the results of Sputnik and TRF tools, limiting the results to minimal repeat of 5, minimal size of 1, and maximum size of 6. Thus, ProGeRF coverage increases to 100% over results of Sputnik and more than 80% over results of TRF (97% for sequences NC_004318.1 and NC_001136.8).

On the other hand, the coverage of ProGeRF by SciRoKo, Mreps, and Sputnik is lower than 46% for all sequences and much lower than 9% when observing the last two sequences. This is consistent with the fact that ProGeRF detects more repetitive elements than others tools.

In the second experiment, we run the ProGeRF web version in the protein mode in circumsporozoite protein (ACO49545.1), merozoite surface protein 1 (XP_001352170.1), and merozoite surface protein 9 (AAN36363.1).

Table 3 presents the result of executing the circumsporozoite protein (ACO49545.1), merozoite surface protein 1 (XP_001352170.1), and merozoite surface protein 9 (AAN36363.1), in which the repetitive element PNAN (PRO-ASN-ALA-ASN) was identified in the circumsporozoite protein as in previous work [24]. In other proteins, repetitive elements have been identified with low repetition frequency. Figure 4 shows the result that is available to the user in the web environment: (A) visualization of results through the jqGrid plugin: clicking over the repetitive element opens the graphical view; (B) repetitive elements are mapped and displayed graphically through JBrowse. In the web environment an identification code is generated for each execution. The code can be used to review the result when necessary and it is still possible to receive a link with the code by email to notify the user.

Table 3: Repetitive protein elements found by the web tool ProGeRF.

ID sequence	Locus	Motif	Rep.
XP_001352170.1	62–97	GASAQS	6
ACO49545.1	146–297	PNAN	38
AAN36363.1	693–712	KEKEE	4

The parameters used were motif size between 2 and 6, repetitions of the least 4 motifs, and zero for the gaps, overlap, and degeneration.

Regarding the tool in web mode, no other web tool offers the user the possibility to consult executions previously carried out and the integration/visualization of results using a dynamic and friendly environment for navigation genome with jBrowse.

4. Conclusion

ProGeRF, the proposed identification algorithm for repetitive elements, presents itself as an efficient, fast, accurate, and

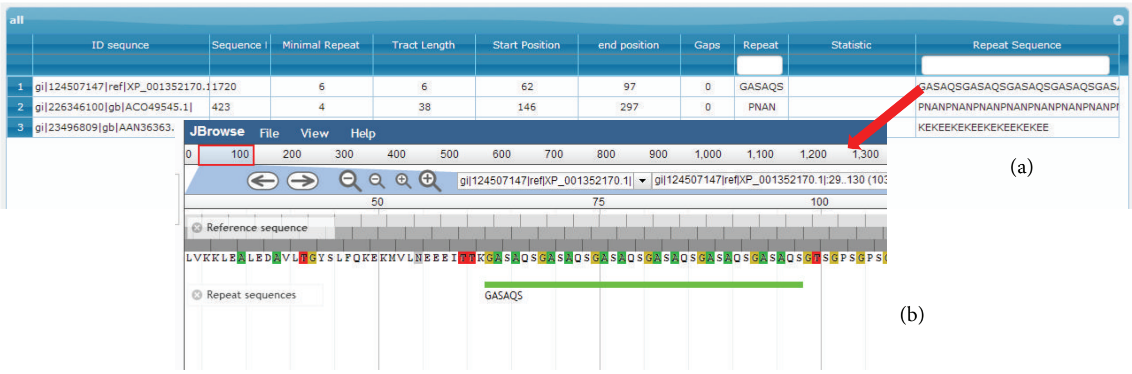


Figure 4: Screen shot from circumsporozoite protein (ACO49545.1), merozoite surface protein 1 (XP_001352170.1), and merozoite surface protein 9 (AAN36363.1) element repetitive search: (a) visualization of results through the jqGrid plugin: by clicking over the repetitive element the graphical view is opened; (b) repetitive elements are mapped and displayed graphically through JBrowse.

easy to use tool and is available in either stand-alone or web mode. It offers a dynamic and user-friendly web interface, the identification of perfect and imperfect repetitive elements, repeat size detection from 1 to 12, repeating the search for specific sizes, preview of the alignment, the flanking sequence, repetition statistics, and a graphical output.

Among the tools that locate both perfect and imperfect repeats ProGeRF is the one that provides graphical visualization and allows for the filtering of the results. Another advantage is the possibility of executing it on genomic and proteomic data and the ability to treat large genomic/proteomic data files.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) [23038008852], FAPEMIG, and CNPq. The authors are very grateful for the Postgraduate Program DINTER UFMT/UFMG.

References

[1] S. Leclercq, E. Rivals, and P. Jarne, "Detecting microsatellites within genomes: significant variation among algorithms," *BMC Bioinformatics*, vol. 8, article 125, 2007.

[2] K. G. Lim, C. K. Kwoh, L. Y. Hsu, and A. Wirawan, "Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance," *Briefings in Bioinformatics*, vol. 14, no. 1, Article ID bbs023, pp. 67–81, 2013.

[3] C. Clayton, "Repetitive elements in parasitic protozoa," *BMC Biology*, vol. 8, article 64, 2010.

[4] B. Wickstead, K. Ersfeld, and K. Gull, "Repetitive elements in genomes of parasitic protozoa," *Microbiology and Molecular Biology Reviews*, vol. 67, no. 3, pp. 360–375, 2003.

[5] J. R. Lupski and P. T. Stankiewicz, *Genomic Disorders: The Genomic Basis of Disease*, Springer, Berlin, Germany, 2007, http://www.springer.com/gp/book/9781588295590.

[6] G.-F. Richard, A. Kerrest, and B. Dujon, "Comparative genomics and molecular dynamics of DNA repeats in eukaryotes," *Microbiology and Molecular Biology Reviews*, vol. 72, no. 4, pp. 686–727, 2008.

[7] C. Schlotterer, "Evolutionary dynamics of microsatellite DNA," *Chromosoma*, vol. 109, no. 6, pp. 365–371, 2000.

[8] S. B. Mudunuri, A. A. Rao, S. Pallamsetty, and H. A. Nagarajaram, "Comparative analysis of microsatellite detecting software: a significant variation in results and influence of parameters," in *Proceedings of the International Symposium on Biocomputing (ISB '10)*, ACM, New York, NY, USA, February 2010.

[9] E. J. Oliveira, J. G. Pádua, M. I. Zucchi, R. Vencovsky, and M. L. C. Vieira, "Origin, evolution and genome distribution of microsatellites," *Genetics and Molecular Biology*, vol. 29, no. 2, pp. 294–307, 2006.

[10] Y. D. Kelkar, N. Strubczewski, S. E. Hile, F. Chiaromonte, K. A. Eckert, and K. D. Makova, "What is a microsatellite: a computational and experimental definition based upon repeat mutational behavior at A/T and GT/AC repeats," *Genome Biology and Evolution*, vol. 2, no. 1, pp. 620–635, 2010.

[11] G. Benson, "Tandem repeats finder: a program to analyze DNA sequences," *Nucleic Acids Research*, vol. 27, no. 2, pp. 573–580, 1999.

[12] A. T. Castelo, W. Martins, and G. R. Gao, "TROLL—tandem repeat occurence locator," *Bioinformatics*, vol. 18, no. 4, pp. 634–636, 2002.

[13] T. Thiel, W. Michalek, R. K. Varshney, and A. Graner, "Exploiting EST databases for the development and characterization of gene-derived SSR-markers in barley (*Hordeum vulgare* L.)," *Theoretical and Applied Genetics*, vol. 106, no. 3, pp. 411–422, 2003.

[14] R. Kolpakov, G. Bana, and G. Kucherov, "mreps: efficient and flexible detection of tandem repeats in DNA," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3672–3678, 2003.

[15] R. Kofler, C. Schlötterer, and T. Lelley, "SciRoKo: a new tool for whole genome microsatellite search and investigation," *Bioinformatics*, vol. 23, no. 13, pp. 1683–1685, 2007.

[16] M. La Rota, R. V. Kantety, J.-K. Yu, and M. E. Sorrells, "Nonrandom distribution and frequencies of genomic and EST-derived microsatellite markers in rice, wheat, and barley," *BMC genomics*, vol. 6, no. 1, article 23, 2005.

[17] L. C. D. Maia, D. A. Palmieri, V. Q. D. Souza, M. M. Kopp, F. I. F. D. Carvalho, and A. Costa de Oliveira, "SSR locator: tool for

simple sequence repeat discovery integrated with primer design and PCR simulation," *International Journal of Plant Genomics*, vol. 2008, Article ID 412696, 9 pages, 2008.

[18] S. B. Mudunuri and H. A. Nagarajaram, "IMEx: imperfect microsatellite extractor," *Bioinformatics*, vol. 23, no. 10, pp. 1181–1187, 2007.

[19] X. Wang, P. Lu, and Z. Luo, "GMATo: a novel tool for the identification and analysis of microsatellites in large genomes," *Bioinformation*, vol. 9, no. 10, pp. 541–544, 2013.

[20] Z. Ning, A. J. Cox, and J. C. Mullikin, "SSAHA: a fast search method for large DNA databases," *Genome Research*, vol. 11, no. 10, pp. 1725–1729, 2001.

[21] J. Reneker and C.-R. Shyu, "Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals," *BMC Bioinformatics*, vol. 6, article 111, 2005.

[22] M. E. Skinner, A. V. Uzilov, L. D. Stein, C. J. Mungall, and I. H. Holmes, "JBrowse: a next-generation genome browser," *Genome Research*, vol. 19, no. 9, pp. 1630–1638, 2009.

[23] T. Tomov, *jqfGridg—Introduction*, 2008.

[24] A. A. Holder, "Developments with anti-malarial vaccines," *Annals of the New York Academy of Sciences*, vol. 700, no. 1, pp. 7–21, 1993.