UNIVERSIDADE FEDERAL DE PELOTAS
Faculdade de Ciência da Computação
Programa de Pós-Graduação em Ciência da Computação

Dissertação

AAE-DeMo: Uma proposta de arquitetura baseada em algoritmos evolutivos para descoberta de Motifs em moléculas biológicas

Augusto Garcia Schmidt

Pelotas, 2017

Augusto Garcia Schmidt

AAE-DeMo: Uma proposta de arquitetura baseada em algoritmos evolutivos para descoberta de Motifs em moléculas biológicas

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Marilton Sanchotene de Aguiar

Pelotas, 2017

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

S349a Schmidt, Augusto

AAE-DeMo : uma proposta de arquitetura baseada em algoritmos evolutivos para descoberta de Motifs em moléculas biológicas / Augusto Schmidt ; Marilton Sanchotene de Aguiar, orientador. — Pelotas, 2017.

50 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2017.

1. Algoritmos evolutivos. 2. Descoberta de motivos. 3. Bioinformática. I. Aguiar, Marilton Sanchotene de, orient. II. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Dedico este trabalho aos meus pais, meus avós e minha namorada.

Agradecimentos

Aos professores e funcionários que integram o programa de pós graduação em ciência da computação da Universidade Federal de Pelotas – UFPEL, pelo apoio a minha participação no programa de mestrado.

Ao meu orientador, Professor Dr. Marilton Sanchotene de Aguiar, por seu apoio, amizade, dedicação que foram fundamentais para realização deste trabalho.

Ao Doutorando Frederico Schmidt Kremmer do Curso de Pós Graduação em Biotecnologia da Universidade Federal de Pelotas - UFPel, que sempre acreditou nesta proposta.

Aos professores do mestrado que destinaram parte de seu precioso tempo e de alguma forma contribuíram para esta pesquisa

Ao Grupo de Aplicações em Inteligência Artificial – GAIA, pela calorosa recepção e companheirismo que tiveram comigo desde o inicio desta jornada.

A minha namorada, grande incentivadora para que eu não desistisse dos meus sonhos.

Aos meus pais, pela minha existência e ajuda prestada no tempo de necessidade.

A minha família e amigos (novos e antigos) que sempre me incentivaram e apoiaram nesta jornada.

“O homem científico não almeja resultados imediatos. Ele não espera que suas ideias mais avançadas sejam rapidamente retomadas. Seu trabalho é como o de um agricultor para o futuro. Seu dever é estabelecer bases para aqueles que estão por vir e apontar o caminho a ser seguido.”
(TESLA, Nikola 1934 p. 2)

RESUMO

SCHMIDT, Augusto Garcia. **AAE-DeMo: Uma proposta de arquitetura baseada em algoritmos evolutivos para descoberta de Motifs em moléculas biológicas.** 2017. 37f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Motivos não são entidades aleatórias encontradas em cadeias de DNA, podendo ser definidos como um fenômeno não único dentro de uma sequência genética. Os motivos, além de ter padrões recorrentes nas sequências analisadas, também possuem uma função biológica. Os algoritmos evolutivos são amplamente utilizados para encontrar soluções para otimização e padrões de pesquisa na área de ciência da computação. Encontrar motivos em sequências de genes é um dos problemas mais importantes na bioinformática e pertence à classe NP-Difícil. Portanto, é plausível investigar a hibridação de ferramentas consolidadas, mas limitadas em seu desempenho, em combinação com técnicas de algoritmos evolutivos. Este trabalho tem a premissa de mostrar uma pesquisa das principais técnicas e conceitos de algoritmos evolutivos utilizados na descoberta de padrões (motivos) na em moléculas e também um estudo aprofundado dos principais algoritmos de bioinformática que são utilizados para esta função em recentes anos por pesquisadores. Entende-se que tais técnicas em combinação, podem obter resultados interessantes para pesquisa em bioinformática. Assim, propondo uma arquitetura otimizada para descoberta de motivos em moléculas de regiões promotoras da bactéria. Usando tanto algoritmos evolutivos, como algoritmos de bioinformática e técnicas de refinação de seus principais dados fornecidos pelos algoritmos utilizados. Assim, formando uma arquitetura com melhor desempenho devido à hibridização de ferramentas consolidadas para buscar padrões em expressões genéticas.

Palavras-Chave: Algoritmos Evolutivos, Descoberta de Motivos, Bioinformática.

ABSTRACT

SCHMIDT, Augusto Garcia. **AAE-DeMo: An Architecture Proposal Based on Evolutionary Algorithms for the Discovery of Motifs in Biological Molecules.** 2017. 37p. Trabalho Individual (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Motifs are not random entities found in DNA strands, and can be defined as a non-unique phenomenon within a genetic sequence. Motifs, besides having recurrent patterns in the analyzed sequences, also have a biological function. Evolutionary algorithms are widely used to find solutions for optimization and research standards in the area of computer science. Finding motifs in gene sequences is one of the most important problems in bioinformatics and belongs to the NP-Difficult class. Therefore, it is plausible to investigate the hybridization of consolidated but limited tools in their performance, in combination with evolutionary algorithm techniques. This work has the premise of showing a research of the main techniques and concepts of evolutionary algorithms used in the discovery of patterns in molecules and also an in depth study of the main bioinformatics algorithms that have been used for this function in recent years by researchers. It is understood that such techniques in combination may yield interesting results for research in bioinformatics. Thus, proposing an architecture optimized for the discovery of motifs in molecules of promoter regions of the bacterium. Using both evolutionary algorithms, bioinformatics algorithms and refining techniques of its main data provided by the algorithms used. Thus, forming an architecture with better performance due to the hybridization of consolidated tools to look for patterns in genetic expressions.

Keywords: Evolutive Algorithms, Motifs Discovery, Bioinformatics.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

É amplamente reconhecido que a biologia está entre as áreas de pesquisa de maior acúmulo de informações nas últimas décadas. Esse acúmulo foi devido a uma série de avanços tecnológicos nos últimos anos que ocasionaram maior qualidade e quantidade de informação coletada de organismos no nível genômico, transcriptômico e proteômico (PROSDOSCIMI, 2007).

No entanto, à medida que essas informações são coletadas em grande quantidade, as técnicas usadas para análise destas informações devem se tornar mais sofisticadas atendendo a esta demanda em grande escala e com ruídos.

Neste contexto, faz-se necessário um estudo sobre algoritmos evolutivos em conjunto com técnicas de bioinformática a serem utilizadas para descoberta de padrões em expressões genéticas (K. DAVIES, 2001). Descobrir padrões não randômicos em cadeias de DNA é um dos problemas mais importantes na bioinformática e pertence a classe de problemas NP-Completo. Portanto, é plausível investigar a hibridização de ferramentas já consolidadas na área de bioinformática, mas limitadas em performance, juntamente com técnicas de algoritmos evolutivos. É neste contexto que este estudo está baseado, objetivando melhorar a tecnologia disponível atualmente aos pesquisadores de bioinformática

São objetivos deste trabalho, prover uma análise aprofundada das técnicas de algoritmos evolutivos apropriadas para descoberta de padrões em expressões genéticas, assim como também propor uma arquitetura otimizada, baseada em algoritmos evolutivos para busca de padrões em expressões genéticas, como alternativa a ferramentas computacionais para descoberta de padrões genéticos, utilizando tanto técnicas de algoritmos evolutivos, como algoritmos de bioinformática e suas principais técnicas de refino de dados.

Este texto está organizado como segue. No Capítulo 2, são apresentados: os principais conceitos e ideias de bioinformática e computação mostrando como são analisados os dados de uma perspectiva de um profissional da área de bioinformática as técnicas de algoritmos evolutivos para busca de padrões contempladas neste trabalho e seus respectivos algoritmos, exemplos de alinhamento local e matrizes de comparação. No Capítulo 3, apresenta-se a abordagem proposta utilizando as técnicas de algoritmos evolutivos com os tratamentos necessários do *dataset* com as matrizes comparativas e alinhamento local. No Capitulo 4, são apresentados os resultados alcançados, mostrando a comparação destes com as ferramentas do estado-da-arte. E, por fim, no Capítulo 5 é feita uma discussão sobre a metodologia utilizada e os resultados obtidos com a arquitetura e tal qual, as dificuldades e percepções sobre o contexto da arquitetura em geral ao longo do desenvolvimento do estudo.

2. REFERENCIAL TEÓRICO E TECNOLÓGICO

2.1 Background Biológico

A bioinformática é uma ciência nova e em desenvolvimento e tanto que o GenBank teve um crescimento exponencial nos últimos anos, mostrando assim que a bioinformática é hoje uma necessidade para a análise de dados em biologia molecular (PROSDOSCIMI, 2007).

Dois desenvolvimentos foram importantes para permitir o surgimento da Bioinformática, o primeiro deles foi o sequenciamento capilar, onde a eletroforese ocorria dentro de tubos que com a espessura de um cabelo humano, contendo uma solução polimérica por onde o DNA passa guiado por uma corrente elétrica. O outro grande desenvolvimento foi a marcação dos didesoxinucleotídeos necessários para sequenciar o DNA com moléculas de marcadores radioativos, que tornavam esta técnica um tanto quanto trabalhosa e perigosa. Era preciso correr diferentes reações para cada nucleotídeo há marcação radioativa, a mesma técnica de marcação fluorescente permitia que cada base fosse marcada com diferentes fluorocromos que emitiam luz com diferentes comprimentos de ondas se excitadas por um laser. Essa luz lida por um detector informava qual nucleotídeo passava enquanto a eletroforese era feita.

Então a união destas duas tecnologias foi capaz de produzir o primeiro equipamento que ficaria conhecido como "*sequenciador que criou a bioinformática*" (PROSDOSCIMI, 2007). O primeiro aparelho criado foi pela empresa *Applied Biosystems*, chamado de ABI Prism 3700, apresentava 96 capilares para eletroforese e cerca de 550 bases em cada coluna sendo oito vezes mais rápida que a melhor máquina concorrente na época, possibilitando o sequenciamento de um milhão de pares de base por dia. Podemos analisar, por exemplo, o crescimento exponencial do GenBank hoje em dia.

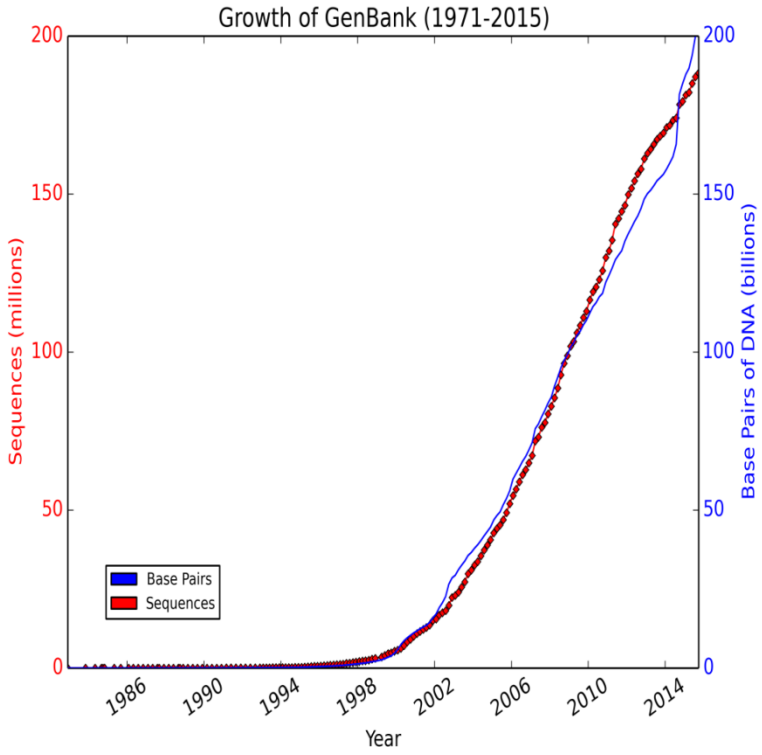

Figura 1: Crescimento do GenBank.

2.1.1 Gene

O Gene é uma unidade fundamental de informação no ácido desoxirribonucleico (DNA) e é definido por uma série de sessões de pares de bases de nucleotídeos e são utilizadas como padrão de cópia em um processo chamado transcrição. Descoberta de padrões em sequência de DNA é um dos maiores desafios na biologia molecular e na ciência da computação. De maneira mais simples: dada uma série de sequências, ache um padrão desconhecido que ocorre frequentemente. Se um padrão de m-letras ocorre frequentemente em todas as sequências, uma numeração de todos os padrões m-letras que aparecem na sequência é dada como solução. No entanto, quando se trabalha com sequências de DNA não é tão simples, pois os padrões incluem mutações, inserções, ou remoções de nucleotídeos (PROSDOSCIMI, 2007).

Um motivo de DNA é definido como um padrão encontrado em uma sequência de nucleotídeos encontrados ao longo de uma expressão genética. Normalmente os padrões são realmente pequenos (5 a 20 pares de base) e recorrentes em diferentes genes ou até mesmo na mesma expressão genética. Motivos podem ser encontrados em ambas as fitas de sequência de DNA. As sequências podem ter zero, um ou múltiplas sequências de cópias de um motivo (K. DAVIES, 2001).

2.1.2 DNA

Ácido desoxirribonucleico (DNA) é uma estrutura que carrega as informações necessárias para a construção e funcionamento de células, além desse papel o DNA também dita a hereditariedade. Dentre as instruções de construção e funcionamento estão características bioquímicas, fisiológicas e anatômicas. O DNA é considerado uma macromolécula formada por unidades chamadas nucleotídeos. Existem quatro nucleotídeos diferentes presentes no DNA: adenina, citosina, timina e guanina, popularmente conhecidas como A, C, T, G; existe ainda um quinto nucleotídeo (presente no RNAm) que é a uracila que na transcrição substitui a timina (K. DAVIES, 2001).

Existem dois tipos de células as eucarióticas e as procarióticas. As células eucarióticas estão presentes desde seres vivos complexos até seres unicelulares enquanto as procarióticas estão presentes principalmente em bactérias. Comparativamente com as células procarióticas as eucarióticas são maiores, mais complexas e possuem núcleo, aonde o DNA é armazenado. Hierarquicamente acima de nucleotídeos têm-se os aminoácidos que são formados por sequências de três nucleotídeos. Existem vinte diferentes tipos de aminoácidos que são responsáveis pela formação das proteínas. Proteínas são os agentes mais ativos dentro uma célula sendo responsáveis pela base da estruturação de diferentes células, tais como musculares, capilares, etc. Podem ainda ter papéis mais vitais como transportar oxigênio para dentro das células ou síntese de hormônios (K. DAVIES, 2001).

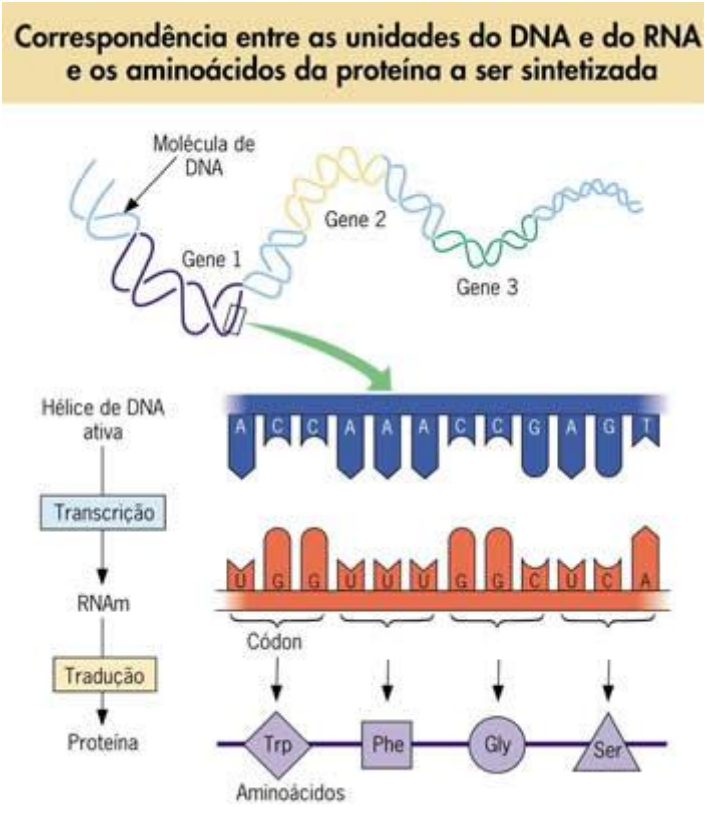

Figura 2: Transição de hélice ativa de DNA para proteína (**CÉSAR, DA SILVA JÚNIOR; SASSON, SEZAR., 2007).**

2.1.3 Códons

A informação contida no DNA é formada por um alfabeto de quatro letras que correspondem aos quatro nucleotídeos: A, C, T, G. Com essas quatro letras é possível formar aminoácidos, formando alguma das vinte abreviações utilizadas pelos vinte aminoácidos diferentes que se encontram nas expressões genéticas dos seres vivos (K. DAVIES, 2001).

Ou seja, na genética, um códon é uma sequência de três nucleotídeos de RNA mensageiro que formam um determinado aminoácido ou que indicam o ponto de início ou fim de tradução de RNA mensageiro. Isto nos mostra que cada conjunto de três nucleotídeos é responsável pela formação de um aminoácido. A Figura 3 demonstra os processos de transição de leitura da Hélice ativa de DNA para a proteína.

As expressões genéticas se expressam por trincas de bases, que foram denominadas códons. Cada códon, formado por três letras, forma um aminoácido. A Figura 3 Mostra a Tabela de Códons e seus respectivos aminoácidos.

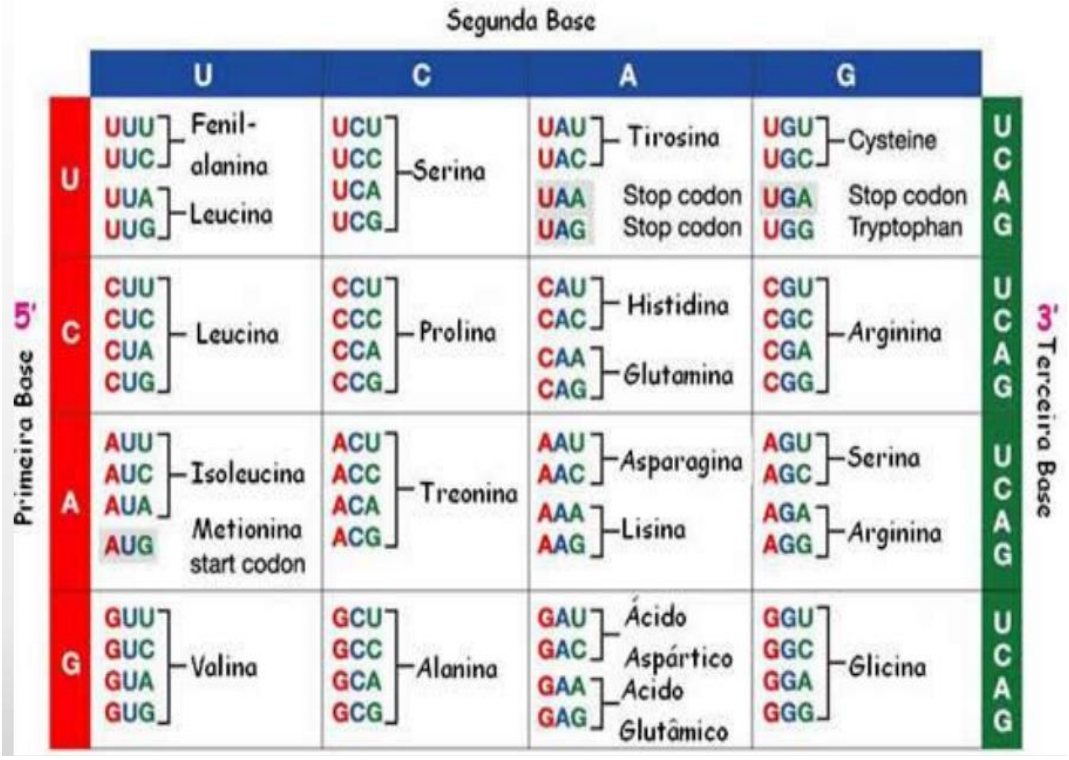

Figura 3: Tabela de Códons e seus respectivos aminoácidos.

2.1.4 Aminoácidos

Os aminoácidos são uma pequena unidade elementar na construção de uma proteína. A sequência de aminoácidos nas proteínas define a forma da mesma e também a sua função, ou seja, para o bom funcionamento de um organismo é necessário um coordenado e eficiente processo de tradução. Um único erro durante a síntese proteica pode causar disfunções ou deficiências ao organismo, como no caso da anemia falciforme em razão de ter sido substituído um aminoácido, o ácido glutâmico por uma valina (K. DAVIES, 2001).

Abaixo é demonstrado os aminoácidos e suas respectivas abreviações.

Tabela 1: Aminoácidos e suas respectivas abreviações

Aminoácido	Abreviação
Alanina	Ala (A)
Arginina	Arg (R)
Ácido Aspárgico	Asp (D)
Asparagina	Asn (N)
Ácido Glutâmico	Glu (E)
Cisteína	Cis (C)
Fenilalanina	Fen (F)
Glutamina	Gln (Q)
Glicina	Gli (G)
Histidina	His (H)
Isoleucina	Ile (I)
Leucina	Leu (L)
Lisina	Lis (K)
Metionina	Met (M)
Prolina	Pro (P)
Serina	Ser (S)
Treonina	Tre (T)
Triptofano	Trp (W)
Tirosina	Tir (Y)
Valina	Val (V)

2.1.5 Motivos

Motivos são entidades não randômicas encontradas em cadeias de DNA. Um padrão também pode ser definido como um fenômeno não único. Podem-se definir motivos como um curto segmento compartilhado por múltiplas sequências de DNA que pode conter informações sobre evolução, estrutura ou função (Yi-Ping Phoebe Chen. 2005). O reconhecimento desses motivos muitas vezes é baseado somente nos padrões compartilhados, pois dificilmente é possível obter informações de estruturas 3D, detalhes das reações químicas, mutações ou funcionalidades. Levando em consideração todos esses pontos nota-se que nem todos os padrões encontrados em cadeias de DNA são motivos.

Dentro do reconhecimento de motivos se pode citar dois tipos o intra-sequências e os entre-sequências. O segundo tendo uma carga informativa maior, pois a probabilidade de que padrões que se mantiveram em diversos indivíduos diferentes possuem uma funcionalidade mais relevante. Motivos não precisam ser exatamente iguais, eles podem ter alguma divergência entre si. Essas divergências são possíveis, pois a redução de afinidade nesse ponto pode ser compensada em um ponto à frente (Yi-Ping Phoebe Chen. 2005).

2.1.6 Representações de Motivos

No nível mais geral, padrões podem ser divididos em determinístico e probabilístico. No modelo determinístico o padrão combina ou não combina com a sequência de entrada. Já o modelo probabilístico atribui uma probabilidade de que a sequência de entrada seja compatível com o padrão sendo estudado (Yi-Ping Phoebe Chen. 2005).

Dentro do grupo de *modelos determinísticos* têm-se expressões regulares e sequência de consenso. As expressões regulares utilizadas em descoberta de motivos são um subconjunto de linguagens regulares, normalmente utilizando símbolos não ambíguos e ambíguos, espaçamentos fixos e variáveis. A utilização de expressões regulares possibilita uma representação mais fácil de padrões complexos, com grandes ou múltiplos espaçamentos (Yi-Ping Phoebe Chen. 2005).

A sequência de consenso representa um conjunto de sequências, possíveis motivos, que estão a uma máxima distância x do consenso. Essa distância x representa o número de caracteres diferentes entre as duas sequências. Cada instância nesse conjunto é chamada de ocorrência de consenso. Normalmente o número de diferenças entre uma ocorrência de consenso e a sequência de consenso é definido; e, é diretamente proporcional ao tamanho do consenso (PARIDA, 2007).

As duas abordagens apresentadas podem ser combinadas de forma que os caracteres presentes nas expressões regulares sejam um consenso de todas as ocorrências obtidas, podendo também permitir divergências entre o consenso e as ocorrências. Porém estas expressões podem se tornar muito complexas e difíceis de serem manipuladas, de fato algumas ferramentas existentes possibilita a utilização de um subconjunto ainda menor de expressões regulares.

O *modelo probabilístico* mais simples é a matriz de pontuação ou PWM (do inglês, *Position Weight Matrices*). Sua maior vantagem em comparação ao modelo determinístico é a capacidade de expressar diferentes importâncias para cada símbolo. Enquanto que no modelo determinístico se uma posição i do consenso possui a ocorrência de dois símbolos, ambos têm a mesma importância. A matriz de pontuação é uma representação de um conjunto de sequências sem espaçamentos, para cada posição da sequência são definidas as frequências de cada símbolo possível. O cálculo é dado pela Equação abaixo.

$$fxi = \frac{Nxi}{N} \quad (1)$$

onde fxi representa a frequência do símbolo *x* na posição *i*, *Nxi* representa o número de ocorrências do símbolo *x* na posição *i* de todas as ocorrências, e *N* é o número total de ocorrências. Desta forma a matriz de pontuação tem dimensões *4xN*. A Figura 4 mostra um exemplo de PWM das probabilidades de cada nucleotídeo se repetir em cada posição (PARIDA, 2007).

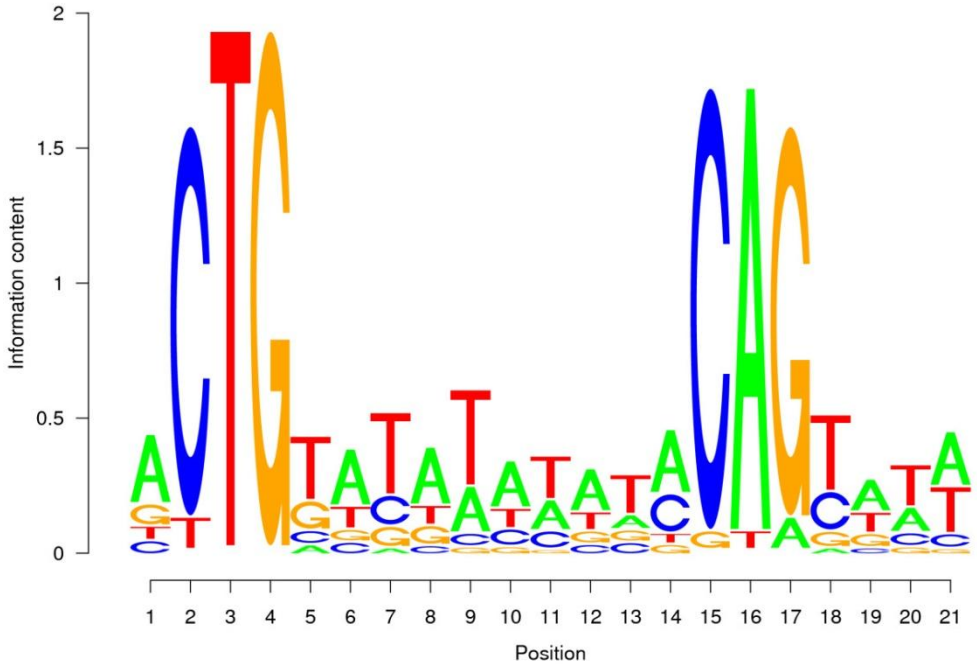

Figura 4: PWM de um modelo Probabilístico.

2.1.7 Reconhecimento de Motivos

Motivos são conjuntos de sequências de DNA, que podem ser representados das diferentes formas apresentadas anteriormente. Geralmente a sequência é analisada utilizando janela deslizante. O tamanho da janela é configurado para o tamanho do motivo e a janela é deslizada pela sequência, uma base por vez até o final da mesma. Um acerto ocorre se a subsequência na janela combina com o motivo sendo analisado, sob as considerações do usuário (PARIDA, 2007).

2.1.8 Bancos de Dados em Biologia Molecular

As bases de dados em biologia molecular são muito importantes para a comunidade científica na forma de poder tornar os dados (produzidos em todo o mundo) acessíveis de uma forma mais fácil para qualquer pesquisador. A primeira base de dados da biologia molecular surgiu por volta de 1960 quando Dayhoffe alguns colaboradores de sua equipe construíram um catálogo contendo todas as sequências de proteínas até a data. Estas sequências foram publicadas em um livro chamado *"Atlas of Protein Sequences and Structure"*, de 1965. E o fato mais interessante é que o conteúdo deste livro não deveria conter mais de um megabyte

de informação sobre proteínas, se fosse transferida para computadores de hoje em dia (BAXEVANIS, Andreas D. & OUELLETTE, BF Francis, 2001).

Com a criação de sequenciamento de DNA e, a partir da década de 1990, do sequenciamento em larga escala, foi necessário construir bancos de dados para abrigar o grande número de sequências de expressões genéticas obtidas por pesquisadores. O NCBI, por exemplo, foi criado pelo NIH (*National Institutes of Health*, o Instituto Nacional de Saúde dos Estados Unidos) em 1988 para abrigar esta demanda de informação (Wheller et al., 2002).

Sendo assim, foi criada uma colaboração internacional para montar um banco de dados de sequências de Nucleotídeos, a INSDC (*International Nucleotide Sequence Database Colaboration*). Esta instituição contém o NCBI, o EMBL (*European Molecular Biology Laboratory* ou Laboratório Europeu de Biologia Molecular) e o DDBJ (*DNA Data Bank of Japan* ou Banco de dados de DNA do Japão) (TATENO, Yoshio et al. 2002). Cada Banco é responsável pela sua demanda de informação e também possibilita a submissão de sequências de DNA entre pesquisadores e entre os próprios bancos, sendo que todos os três possuem informações atualizadas de todas as sequências disponíveis para os pesquisadores (STOESSER, Guenter et al. 2002).

Existem dois tipos de banco de dados que estão disponíveis para pesquisa de genes e proteínas (Baxevanis & Ouellette 2001). Os bancos de dados primários apresentam resultados de dados experimentais que são publicados, mas não há uma análise cuidadosa na interpretação dos mesmos. Esse é o caso do GenBank, EMBL e PDB (Protein Data Bank). Já os secundários são aqueles onde há uma compilação e interpretação dos dados de vários grupos de cientistas onde os dados são analisados mais cuidadosamente para serem mais interessantes e representativos. Estes são os casos dos bancos de dados curados, como oCOG, SWISS-PROT, e TrEMBL e ProDom.

2.2 Alinhamento de Sequências

O alinhamento de sequências é na verdade um processo de comparar várias sequências de nucleotídeos ou proteínas para observar o seu nível de semelhança. Essa técnica de comparação de sequências é implementada com uma técnica denominada algoritmo guloso e é um dos pilares de toda a bioinformática.

Existem hoje em dia centenas de aplicações de alinhamento de sequências, para identificação de genes e proteínas desconhecidas e também para comparar a o quão a ordem de genes em genomas de organismos semelhantes são relacionados (sintenia), no mapeamento destas sequências expressas dentro de expressões genéticas para identificações de genes, na montagem de genomas inteiros e em diversas outras aplicações (PROSDOCIMI, 2007).

Por exemplo, podemos alinhar duas sequências para descobrirmos o quão similar elas são até mesmo inferir (ou não) a uma delas, alguma propriedade já conhecida de outra (PROSDOSCIMI et al., 2003). O alinhamento entre as sequências pode ser feito de maneira global ou local como explicado na Figura 8.

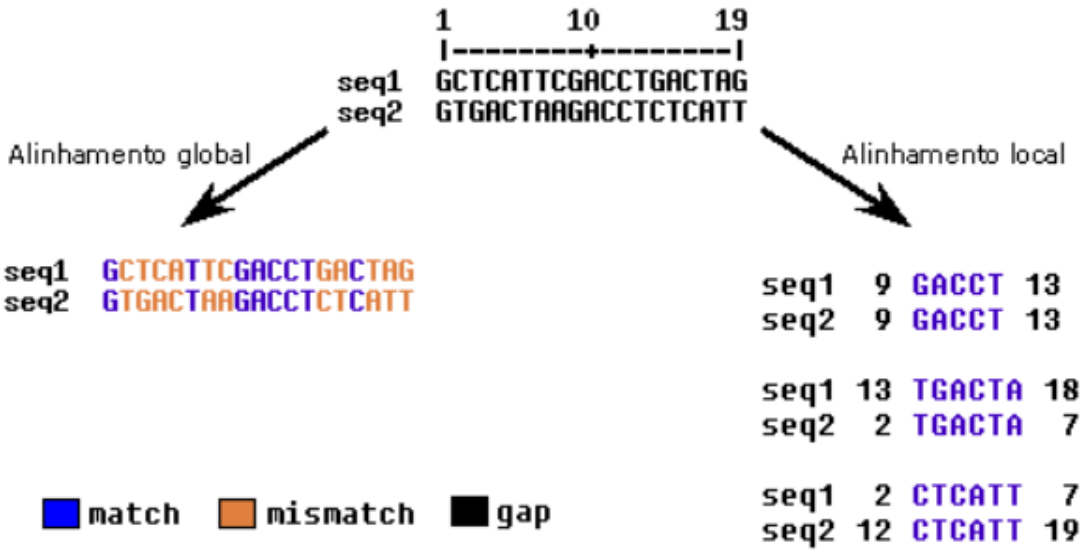

Figura 5: Alinhamento Global e Local.

2.2.1 Alinhamento Global

O alinhamento global é feito quando comparam-se aminoácidos e nucleotídeos de uma sequência com outra sequência que se está querendo analisar, ao longo de toda sua extensão.

O algoritmo Needleman-Wunsch é o mais utilizado neste tipo de alinhamento, embora outros algoritmos também o façam, o Algoritmo citado anteriormente é o mais conhecido e utilizado. Neste caso em questão de alinhamento global, são dados os valores em uma matriz de comparação de similaridades (matches), diferenças (mismatches) e falhas (gaps) encontradas dentro da sequência sendo alinhada.

2.2.2 Alinhamento Local

O alinhamento local acontece quando a comparação entre as duas sequências a serem analisadas não é efetuada ao longo de toda sua extensão, mas sim através de pequenas regiões das mesmas. Foram utilizados neste estudo duas ferramentas de alinhamento local, o BLAST e o USEARCH. Um dos programas a ser constantemente utilizado é o BLAST (ALTSCHUL, 1990). Este software foi montado de forma a explorar toda a informação contida em bases de dados de proteínas e também foi desenvolvido de modo a aumentar a velocidade de busca por similaridade, isto tendo em vista que as bases de dados já são grandes e vem crescendo exponencialmente, mesmo correndo risco de perder um pouco de sensibilidade em seu resultado final. Um exemplo de alinhamento local é demonstrado abaixo pela Figura 6.

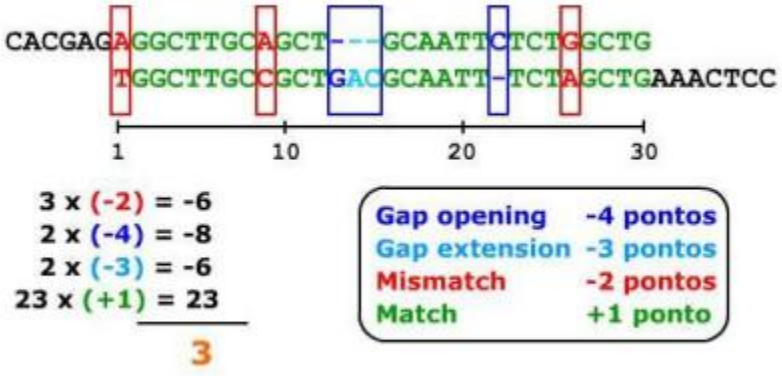

Figura 6: Alinhamento local de sequências.

A rapidez da busca deve-se ao fato do software fragmentar os dados e utilize uma heurística que quebra as sequências de entrada e das bases de dados em fragmentos (palavras) e procura, inicialmente, similaridades entre as sequências. A busca então começa com palavras de tamanho W que devem apresentar um escore T de alinhamento entre si, dado de acordo com uma matriz de valores. Assim, as palavras que apresentam esse escore T (maior responsável pela velocidade e sensibilidade da busca) são estendidas em ambas as direções para gerar um alinhamento com um score maior do que S. Outra vantagem de utilizar o alinhamento feito pelo BLAST é que, dessa forma, é possível identificar relações entre sequências que apresentam apenas regiões isoladas de similaridade.

O algoritmo USEARCH busca por similaridades em um ou mais bancos de dados (datasets), utiliza o comando usearch_global (busca global) e usearch_local (busca local) para efetuar as buscas e explora o fato de que sequências similares tendem a partilhar pequenas palavras em comum, estas palavras, normalmente possuem um tamanho k fixo, portanto o USEARCH agrupa estas palavras e prioriza sua busca ao redor das mesmas, proporcionando facilidade e rapidez a busca (EDGAR, 2010).

No BLAST existe uma diferente variedade de programas a serem utilizados, dependendo do tipo de sequência a ser analisada. Já o USEARCH, é um programa só. O BLAST requer que a banco de dados esteja formatado antes de poder analisá-lo, com os comandos formatdb ou makeblastdb, já o programa USEARCH só precisa que o banco de dados esteja no formato .fasta.

USEARCH e Blast são ferramentas similares, possuindo prós e contras e, em muitos casos, podem não ser diretamente comparáveis em algumas características, porém na questão de predição de similaridades em alinhamentos locais o USEARCH tende a se sair melhor do que o BLAST, pois o alinhamento local do BLAST algumas vezes acha um único domínio dentro de sequências de proteínas que de outra forma possuem organizações de domínios diferentes e, portanto, uma função bastante diferente (EDGAR, 2010).

2.2.3 Alinhamentos Simples e Múltiplo

Existem dois tipos de alinhamento de sequências que são utilizados. Quando apenas duas sequências são comparadas entre si, diz-se que o alinhamento é simples. E, nesses casos, normalmente prefere-se utilizar alinhamentos ótimos para gerarem melhores resultados, exceto os casos onde milhares de alinhamentos simples devem ser realizados.

De forma contrária, considera-se um alinhamento múltiplo quando três ou mais sequências devem ser analisadas e comparadas entre si. No fundo, o alinhamento múltiplo é montado a partir do alinhamento par a par de cada uma das sequências que estão sendo analisadas com todas as outras sequências, seguido por outro procedimento que irá gerar o resultado final do alinhamento de todas contra todas, sendo assim, se forem analisadas 10 expressões genéticas serão necessárias 10! (fatorial de 10) comparações de sequências, o que representa 3.628.800 comparações (PROSDOCIMI, 2007). E é exatamente por isso que os programas heurísticos são preferidos para gerar este tipo de resultado.

2.3 Background Computacional

2.3.1 Algoritmos Evolutivos

Algoritmos evolutivos são, em muitos casos, técnicas de alternativa prática para resolver uma variedade de problemas científicos. Os computadores possuem algumas vantagens sobre os seres humanos principalmente no que diz respeito à velocidade e eficácia com que executam tarefas. Para fazer um processador de símbolos desempenhar uma tarefa tão bem quanto um especialista humano, alguém deve muni-lo de conhecimento especializado, comparável ao do especialista humano. Hoje em dia os avanços de hardware e ciência cognitiva tornam possível a criação de ferramentas e técnicas capazes de efetuar a tarefa humana.

Pode-se fazer a analogia de que o algoritmo simula uma evolução natural como a tabela abaixo nos mostra (Yi-Ping Phoebe Chen, 2005).

Tabela 2: Comparação de evolução natural e algoritmos genéticos.

Evolução Natural	Algoritmo Genético
Ambiente	Dado problema
Cromossomo	Vetor binário
Fitness e fenótipo (probabilidades de sobrevivência)	Função fitness
Lacuna	Posição em um vetor
Seleção, recombinação, crossover e mutação.	Operadores genéticos
Uma população de cromossomos que se adaptaram ao ambiente	Ótima solução encontrada pelo algoritmo para aquele problema

2.3.2 Definição

Algoritmos Evolutivos (AE) ou Computação Evolucionária é uma técnica de otimização busca que utiliza três mecanismos de evolução, estes são: reprodução, mutação e a sobrevivência do mais apto. Onde, a partir da reprodução sucessiva os indivíduos melhoram com o passar das gerações.

A maior parte das abordagens de AE, a população é iniciada aleatoriamente e distribuída uniformemente, e os seus indivíduos (possíveis soluções), são analisados a cada geração por um calculo de uma função objetivo (fitness), onde quão maior o fitness do individuo, melhor suas chances de passar para população da próxima

geração. Os melhores indivíduos possuem a capacidade de passar as suas características para a próxima geração.

Portanto, os AE são de fato inspirados pelos mecanismos de evolução biológica e estes se mostraram úteis na solução de problemas de busca e otimização utilizando princípios evolucionários como reprodução, mutação, seleção e recombinação (crossover), (conhecidos comumente como operadores genéticos) para descobrir soluções a problemas com uma série de soluções iniciais. A escolha ou presença (ou não) destes operadores é o que diferencia uma técnica da outra.

Cada um destes operadores age em um ou mais cromossomos (soluções) em uma população (soluções possíveis) e passam seus genes para novas gerações de soluções. O algoritmo é interativo, portanto age na população várias e várias vezes gerando várias gerações atrás da outra até chegarem a uma solução ótima (BARRET, Steven J., 2006).

2.3.3 Operadores

Uma vez que uma população de soluções foi criada, os operadores genéticos atuam em cima da população de soluções para achar novas e melhores soluções (Yi-Ping Phoebe Chen, 2005).

A parte mais importante é qual dos indivíduos terá sobrevivido para a próxima geração que irá ser determinado pelos operadores genéticos: seleção e *crossover*. Abaixo segue a descrição de cada um dos operadores usado em aplicações de algoritmos evolutivos.

Seleção

Como em outros algoritmos de busca os AE devem lembrar as ótimas soluções que encontraram e descartar as más soluções se as mesmas não estiverem progredindo. Um seletor então avalia os melhores N cromossomos de cada população que irão progredir para a próxima geração. Isto funciona até o ponto que os filhos de cada geração atingem um valor X onde sempre todos irão passar para a próxima geração, criando assim um conceito de sociedade elitista (Yi-Ping Phoebe Chen, 2005).

Entretanto para não deixar que haja convergência nas soluções muito rapidamente um elemento “aleatório” deve ser inserido criando assim o método de seletor chamado de roleta que é um dos mais populares junto com o método torneio que será explicado logo à frente (BARRET, Steven J., 2006).

Método de roleta, este seletor funciona adicionando todos os valores de fitness de uma população em uma roleta virtual fazendo com que ela gire e selecione aleatoriamente os indivíduos que irão para a próxima geração. Como pode ser visto na Figura 7, quando a roleta é girada os indivíduos com menos fitness tem mais chance de irem para a próxima geração do que anteriormente.

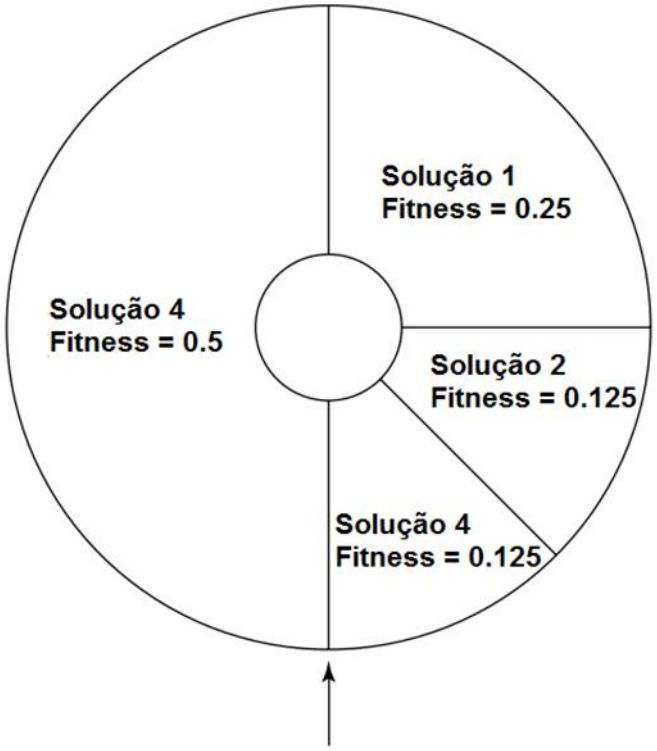

Figura 7: Método de Seleção Roleta

Isto soluciona o problema, porem os elementos estocásticos da seleção garante que a diversidade da população se mantenha. O método torneio é similar ao de roleta na aleatoriedade, mas contem um viés para os indivíduos mais aptos que passariam para a próxima geração normalmente. No método torneio são selecionados dois cromossomos de uma determinada população e ele tem seu fitness comparado.

O cromossomo com o maior fitness vence e entra para a próxima geração. A seleção aleatória dos indivíduos que participarão do torneio significa que duas soluções ruins podem ser selecionadas de uma vez. Nesta situação, mesmo sabendo que as duas soluções são de baixo fitness analisadas com o resto da população, o melhor individuo dos dois será selecionado. Desta forma as soluções com baixo fitness podem ser mesmo assim selecionadas para o torneio. Este método de seleção garante que, em um limiar de tempo de N gerações os indivíduos medianos provavelmente serão selecionados para a próxima geração e assim mantendo a informação descoberta nas gerações anteriores. Uma vez selecionada a solução faz a operação de crossover.

Método torneio, neste método, existe uma de seleção aleatória, porém somente os melhores indivíduos são passados para próxima geração. Normalmente dois indivíduos são selecionados da população e tem seu fitness comparado, então o cromossomo com maior fitness é passado para próxima geração. A seleção aleatória de indivíduos para participar do torneio, da à possibilidade de que indivíduos com performance baixa sejam selecionados e tenham chance de passar

para próxima geração, neste caso onde o melhor dos dois indivíduos passa para próxima geração, portanto, a utilização este método garante a preservação da informação descoberta em outras gerações para as gerações subsequentes. Depois de selecionadas, as soluções passam pelo método de crossover que é descrito na sessão a seguir.

Crossover (Recombinação)

O operador *crossover* é utilizado quando duas soluções “parentes” podem combinar informações para produzir duas novas soluções “filhas” que são diferentes, porém parecidas com as soluções originais encontradas. Podemos utilizar várias formas métodos que são de uso comum quando temos estas operações de crossover: recombinação em um ponto e recombinação uniforme são as mais utilizadas (BARRET, Steven J, 2006).

Recombinação em um ponto é quando um único ponto dos dois cromossomos progenitores é selecionado e são realocados em algum ponto aleatório em cada cromossomo, cortando assim o cromossomo em dois e são recombinadas na forma de dois novos indivíduos que compartilham informações com seus pais.

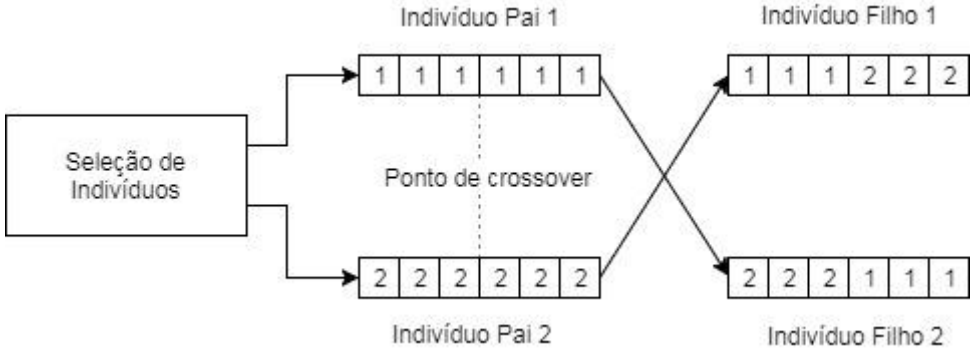

Figura 8: Crossover em um ponto.

Recombinação Uniforme

Ambos os cromossomos progenitores se recombinam para formar seus novos descendentes. Neste método, os bits dos vetores são comparados individualmente entre os cromossomos progenitores então ocorre um intercâmbio de probabilidade fixada, usualmente este valor é 0.5 para manter a integridade da informação antiga.

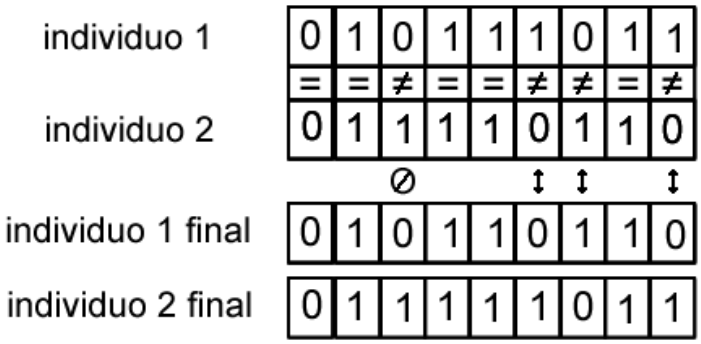

Figura 9: Recombinação Uniforme.

Mutação

Depois de realizado o processo de cruzamento sobre os indivíduos filhos, aplica-se a operação de mutação com uma probabilidade pré-determinada (P_m) onde é possível inserir material genético na população gerada aumentando a possibilidade de variação genética evitando convergência prematura no AE.

Este operador de mutação possibilita que aumente o espaço de busca do algoritmo evitando que os indivíduos fiquem estagnados em um valor. No entanto a probabilidade de mutação não deve ser alta, pois pode afetar o processo de busca da solução fazendo que o mesmo se torne completamente aleatório destruindo as chances de achar uma boa solução para o problema. Portanto, comumente é definido o valor de mutação $P_m = 0{,}1$ ou $P_m = 0{,}01$ ou $P_m = 0{,}001$ (FRANCO, Dielle Correa et al., 2013)

Foi descartada a utilização do operador de mutação na arquitetura AAE-DeMo, pois normalmente a mutação é utilizada para manter a diversidade da população, que neste caso levaria a criação de novas informações mudando o contexto da sequência original. Goldberg (1989) sugere a utilização do valor 0,05 (5%) para a variável P_m (probabilidade de mutação). Mesmo este valor sendo baixo, ele pode mudar totalmente o contexto dos indivíduos a serem analisados. As mutações são extremamente raras no contexto biológico, a frequência estimada na maioria das mutações em organismos é de um em dez mil a um em um milhão de chances de a mesma acontecer (SNUSTAD DP. et al,1997).

Existem várias metodologias hoje em dia para descoberta de motivos utilizando algoritmos inteligentes como, por exemplo, o FMGA - *Finding Motif with Genetic Algorithm* (LIU, Falcon. 2004), onde o mesmo utiliza um algoritmo genético para descoberta de motivos, usando o framework SAGA - *Sequence Alignment by Genetics Algorithm* (NOTREDAME, Cédric. 1996) para efetuar o alinhamento de sequências e prover auxílio no momento de atribuição de fitness para os indivíduos.

GAME - *Genetic Algorithm Motif Elicitation* (WEI, Zhi. 2006), utiliza os operadores genéticos de (Michalewicz, 1996), Utiliza uma função bayesiana proposta por Jensen (JENSEN, et al. 2004) para prover um score baseado em uma PWM onde as prováveis posições que possuem motivos são encontradas, e também dois operadores adicionais SHIFT e ADJUST para o algoritmo não ficar preso em posições ótimas locais. O operador ADJUST compara dois prováveis motivos em uma sequência e remove o que possuir menor fitness, já o operador SHIFT considera a mudança de posições de parte da sequência do motivo tanto para esquerda ou para direita, calculando novamente o fitness até ter um resultado melhor.

MDGA - *Motif Discovery Using a Genetic Algorithm* (CHE, Dongsheng. 2005), propõe uma nova metodologia de algoritmo genético onde pode predizer motivos na sequência a ser analisada. O fitness de cada individuo do algoritmo é a soma do conteúdo de cada coluna do alinhamento da sequência.

E por fim Clare Bates desenvolveu uma abordagem de algoritmo genético para inferência de motivos, chamada GAMI - *Genetic Algorithms approach to Motif Inference* (CONGDON, Clare Bates et al. 2005), para trabalhar com espécies diferentes e seqüências de nucleotídeos longas. O design do sistema reduz o tamanho do espaço de busca em comparação com abordagens típicas de localização de janela para inferência de motivos.

Diferentemente das metodologias citadas acima, AAE-DeMo utiliza o alinhamento de sequências dada pela ferramenta USEARCH (EDGAR, 2010), que além de fornecer o resultado do alinhamento, ela diminui o espaço de busca da sequência, partindo do pretexto de que motivos normalmente são encontrados em padrões recorrentes da sequência a ser analisada, então o foco da busca pelo motivo se dá em torno deste grupo de padrões recorrentes. No entanto, todos os estudos citados anteriormente inclusive a arquitetura AAE-DeMo possuem diferentes métodos de calculo de fitness, possuem o tamanho do motivo a ser descoberto dado de antemão e apenas um motivo por sequência é descoberto.

3. IMPLEMENTAÇÃO DA ARQUITETURA

3.1 Introdução

Para implementação da arquitetura AAE-DeMo acrônimo para (Arquitetura de Algoritmos Evolutivos para Descoberta de Motivos) foram utilizados todos os conceitos discutidos nos Capítulos anteriores e foi desenvolvida como *pipeline* onde cada resultado de saída é a entrada para próxima fase da arquitetura. O desenvolvimento da arquitetura foi no sistema operacional Linux. Todo código executado foi desenvolvido na linguagem de programação Python 3, onde foram utilizadas algumas funções da biblioteca BioPython (COCK, Peter JA et al. 2009) para manipulação de sequências. O fluxo de execução da arquitetura proposta é demonstrado abaixo pela figura 10.

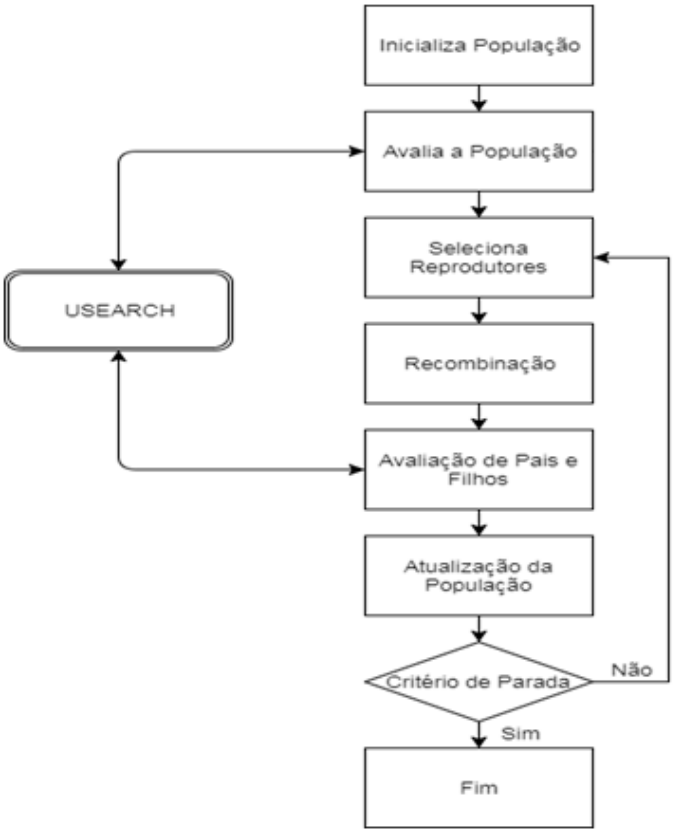

Figura 10: Fluxo de Dados da Arquitetura.

Foram utilizados dois *datasets* para teste desta arquitetura, PS00211 e PS50151, ambos os arquivos foram retirados do banco de dados PROSITE e cada uma destas sequências nestes arquivos possui um motivo já catalogado e armazenado em outro arquivo para se analisar a precisão da arquitetura. O primeiro

dataset PS00211 possui 4190 sequências, estas sequências possuem motivos de tamanho 15, já o segundo *dataset* possui 993 sequências cada uma delas com motivos de tamanho 36.

3.2 O Algoritmo Proposto

O Algoritmo foi implementado utilizando técnicas encontradas em trabalhos anteriores e foi escrito na linguagem de programação Python. A teoria do algoritmo é simples e algumas considerações e decisões foram necessárias para sua implementação.

Dado um grupo de *M* sequências de tamanho 1 de um alfabeto $\sum = \{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V, U, O, B, Z, J, X\}$ o algorimo deve encontrar a combinação de uma ou mais sequências de candidatos de motivos de tamanho W. No entanto, a dificuldade do problema cresce à medida que há a liberdade dos motivos terem diferentes tamanhos ou proteínas diferentes dentro da sequência.

Conforme apresentado na Equação 2, o número total de possíveis motivos em uma sequência *i*, deve-se entender que testar todas as possibilidades se torna inviável.

$$\prod_{\mathrm{i}} 26^{\mathrm{l}-\mathrm{w}+1} \tag{2}$$

Dois métodos para geração das populações iniciais foram implementados, utilizando duas metodologias, a primeira é a coleta de sequências pelo método de seleção por roleta; e, o outro método é pela seleção de torneio. Porém devido aos testes realizados tipicamente de 100 a 250 indivíduos são selecionados e, no entanto, o método de torneio se provou superior ao de roleta devido ao AE utilizando o método roleta demorar mais para achar uma solução ótima ou ficar preso no seu máximo local.

3.2.1 Inicialização

A primeira etapa é a criação de um arquivo com a população inicial a ser explorada pelo algoritmo evolutivo, a população inicial é composta por *n* indivíduos criados aleatoriamente, onde cada indivíduo da população representa uma possível solução para o problema.

3.2.2 Cálculo de Fitness

Após a geração das populações iniciais, o fitness de cada indivíduo é computado usando o alinhamento do indivíduo com o *dataset* de sequências.

O alinhamento é efetuado utilizando a ferramenta Usearch (EDGAR, 2010), onde o *score* do alinhamento é usado no momento do cálculo de fitness do indivíduo da população. Para contornar o problema das sequências a serem alinhadas possuírem desigualdades, foi adotada a medida de utilizar a Equação 3 para

maximizar o limiar de identidade entre as duas sequências, adotando um valor de 0.7 onde o limiar de identidade da população melhorou a qualidade das sequências a serem analisadas.

$$TS_1/TS_2 >= L \quad (3)$$

Onde TS_1se dá pelo valor *High-scoring Segment Pairs*(HSP) (KARLIN, ALTSCHUL, 1993) do tamanho da sequência do indivíduo um, dividido por TS_2, que se dá pelo valor HSP do tamanho da sequência do individuo dois, onde devem possuir um valor maior ou igual a *L* que se dá pelo limiar de similaridade definido anteriormente (0.7).

No que se refere ao contexto de alinhamento das sequências, a pontuação é referente à qualidade do alinhamento. Portanto, quanto maior for a pontuação do alinhamento, maior similaridade entre as sequências alinhadas. A escala de pontuação depende do sistema de pontos usado na matriz de substituição penalidade por gap, extensão de gap, match e mismatch. gaps são inseridos entre resíduos para que caracteres semelhantes possam ser alinhados com colunas sucessivas em uma matriz de alinhamento, extensão de gap é uma penalidade dada por cada gap subseqüente ao primeiro em uma sequência, mismatch é quando não há uma correlação entre as duas sequências e match é quando há uma correlação entre as 2 sequências.

Tabela 3: Tabela de pontuação de alinhamento.

Operação	Pontuação
Gap	-4
Extensão de Gap	-3
Mismatch	-2
Match	+1

O fitness de cada indivíduo é computado usando o *score* dado pelo alinhamento da sequência pelo Usearch, que é definido pela Equação 4*.*

$$Fitness(Motif) = \sum_{i=1}^{t} max(i) \quad (4)$$

Onde *Motif* é o candidato a motivo, *t* é o tamanho do motivo e *max(i)* é o valor da freqüência máxima da coluna i na PWM criada pelo Usearch no alinhamento.

3.2.3 Seleção de Pais

A seleção é feita por torneio dos melhores indivíduos que serão utilizados para reprodução onde setenta por cento da população é escolhida para ser aplicada a função de cruzamento nos pais. Os pares são escolhidos aleatoriamente dentro dos setenta por cento selecionados. Goldberg sugere a utilização de um valor superior a 0,60 (60%) de probabilidade de cruzamento(GOLDBERG, D., 1989).

3.2.4 Crossover

A função de *crossover* é responsável por recombinar um ou mais indivíduos para a população seguinte, foi definido para utilizar a metodologia de *crossover* de um ponto no algoritmo. Esta metodologia tem como base a divisão de dois indivíduos e a recombinação dos mesmos a partir de um ponto onde os indivíduos são divididos pela metade e recombinados.

3.2.5 Avaliação dos Pais e Filhos

Após o crossover ser executado, é aplicado a função de fitness descrita anteriormente nos filhos destes cruzamentos fazendo o alinhamento da sequência dos filhos com o *dataset* que está sendo utilizado.

3.2.6 Atualização da População

Os pais e os seus respectivos filhos são comparados, onde o que estiver com o melhor fitness tomará o seu lugar dentro da população.

3.2.7 Método de Parada

Como não é possível afirmar se neste problema a solução ótima foi encontrada, foi necessário definir um método de parada, onde a abordagem encontrada foi definir que o algoritmo somente continuará a busca por novas soluções enquanto o número de gerações em que os determinados indivíduos melhores adaptados tenham mudado for menor que um quinto da geração máxima definida pelo usuário.

Portanto se a geração máxima definida pelo usuário for de cem gerações e não houver alteração nos melhores indivíduos por 20 gerações então é considerado que não haverá mais atualizações para melhora da população e o processo é finalizado.

4. RESULTADOS ALCANÇADOS

4.1 Introdução

Para mensurar os resultados foi utilizada a linguagem de programação R para melhor visualização dos resultados retornados pela arquitetura.

Foram utilizados dois *datasets* para obtenção dos resultados desta arquitetura, PS00211 e PS50151, onde o primeiro pertence a superfamília ATP-Binding Cassette que está envolvida na exportação e importação de uma série de substratos, desde macromoléculas até pequenos íons e sua função mais importante de seu sistema de importação é prover nutrientes essenciais à bactéria. O dataset é caracterizado por 4.191 sequências, possuindo em sua escala taxonômica diferentes classificações de sequência variando entre arqueia, bacteriófago, eucariotos, procariotos (Bactérias), vírus Eucariotos, cada uma delas possuindo um motivo de 15 caracteres; e, o segundo pertence ao perfil de domínio UVR que sua principal função é o reconhecimento e processamento de reparação do DNA o dataset é caracterizado por 992 sequências, possuindo em sua escala taxonômica diferentes classificações de sequências variando desde arqueia, eucariotos, procariotos (Bactérias), cada uma delas contendo um motivo de 36 caracteres.

Ambos os datasets foram retirados do PROSITE, que é um banco de dados abrangente de famílias de domínios protéicos, gerados automaticamente a partir do banco de dados de conhecimento UniProt e foram selecionados devido a sua variedade taxonômica e também devido ao tamanho de motivo encontrados nas sequências, mais especificamente tamanho 15 no primeiro dataset e tamanho 36 no segundo dataset, de forma que os datasets sejam variados e a arquitetura proposta neste estudo não se adaptasse somente a um tipo de dataset. Os motivos publicados foram utilizados para validação dos testes.

Foram implementadas cinco possibilidades para rodar o algoritmo evolutivo, 10 gerações e 100 indivíduos, 25 gerações e 250 indivíduos, 50 gerações e 500 indivíduos, 75 gerações e 750 indivíduos e 100 gerações e 1000 indivíduos. Em ambos datasets, os melhores resultados encontrados foram quando o algoritmo era rodado com 75 gerações e 750 indivíduos.

Os resultados foram obtidos utilizando a arquitetura AAE-DeMo proposta neste trabalho em comparação com a ferramenta específica para descoberta de motivos utilizada hoje em dia, o MEME/MAST (BAILEY, Timothy L. 1994 e 1998).

MEME é uma ferramenta usada em bioinformática para descoberta de motivos agrupando padrões que aparecem repetidamente na sequência a ser analisada, porém no resultado de saída da ferramenta se o motivo possui gap de alguns caracteres, ele os separa e caracteriza os mesmos como dois motivos diferentes (BAILEY, Timothy L. 1994).

MEME é utilizado em conjunto com a ferramenta MAST que é o acrônimo de "*Motif Alignment and Search Tool*", Onde o resultado de saída do MEME é o arquivo de entrada para o MAST descrevendo os motivos previamente encontrados. Então ele analisa a descrição dos mesmos e busca no *dataset* outros possíveis motivos, calculando uma matriz de probabilidade dos possíveis motivos, levando em consideração a pontuação desta matriz em cada posição da sequência. É apresentado abaixo as configurações que foram utilizadas nas ferramentas.

MEME

Tabela 4: Configurações MEME.

Dataset	Tempo de Execução	Número Maximo de Motivos a Encontrar	Tamanho Máximo da Sequência	Tipo de busca	Tamanho Mínimo do Motivo	Tamanho Máximo do Motivo
PS00211	18.000 segundos	1	800000 caracteres	*ZOOPS	5	15
PS50151	18.000 segundos	1	800000 caracteres	*ZOOPS	10	36

*ZOOPS, é o acrônimo de *"Zero or One Occurrences Per Sequence"* zero ou uma ocorrência por sequência.

MAST

Foram utilizadas as configurações padrão da ferramenta.

USEARCH

Tabela 5: Configurações USEARCH.

Dataset	Versão do USEARCH	Tipo de Alinhamento	Mínimo de Identidade	Tamanho Máximo do Motivo	Tamanho Mínimo do Motivo
PS00211	8.1.1812.i86linux32	Local	0.7	5	15
PS50151	8.1.1812.i86linux32	Local	0.7	10	36

4.2 Análise de Desempenho

Analisando os resultados adquiridos com o *dataset* PS00211 em que todas as sequências possuem motivos de tamanho 15, pode-se perceber na Figura 12 que a média de similaridade entre o motivo descoberto e o motivo já catalogado é de 10,27 matchs (68%), o pior caso foi registrado 5 matchs e o melhor caso foram de 12 matchs, os melhores resultados foram de 75 gerações e 750 indivíduos. A Figura 11 mostra a porcentagem da semelhança por individuo analisado e o motivo já catalogado no *dataset* PS00211 com 75 gerações e 750 indivíduos.

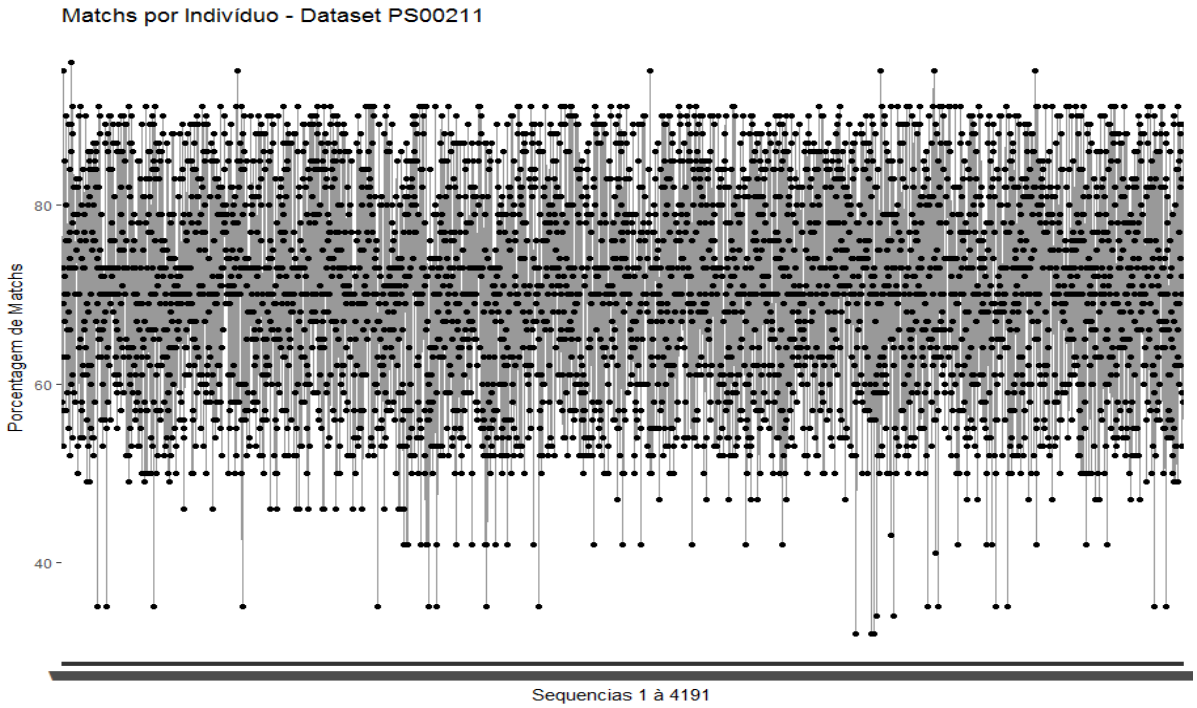

Figura 11: Match por individuo - dataset PS00211.

Analisando os resultados adquiridos com o dataset PS50151 em que todas as sequências possuem motivos de tamanho 36, pode-se perceber na Figura 12 que a média de similaridade entre o motivo descoberto e o motivo já catalogado é de 24,84 matchs (69%), o pior caso foi registrado 16 matchs e o melhor caso foram de 28 matchs.

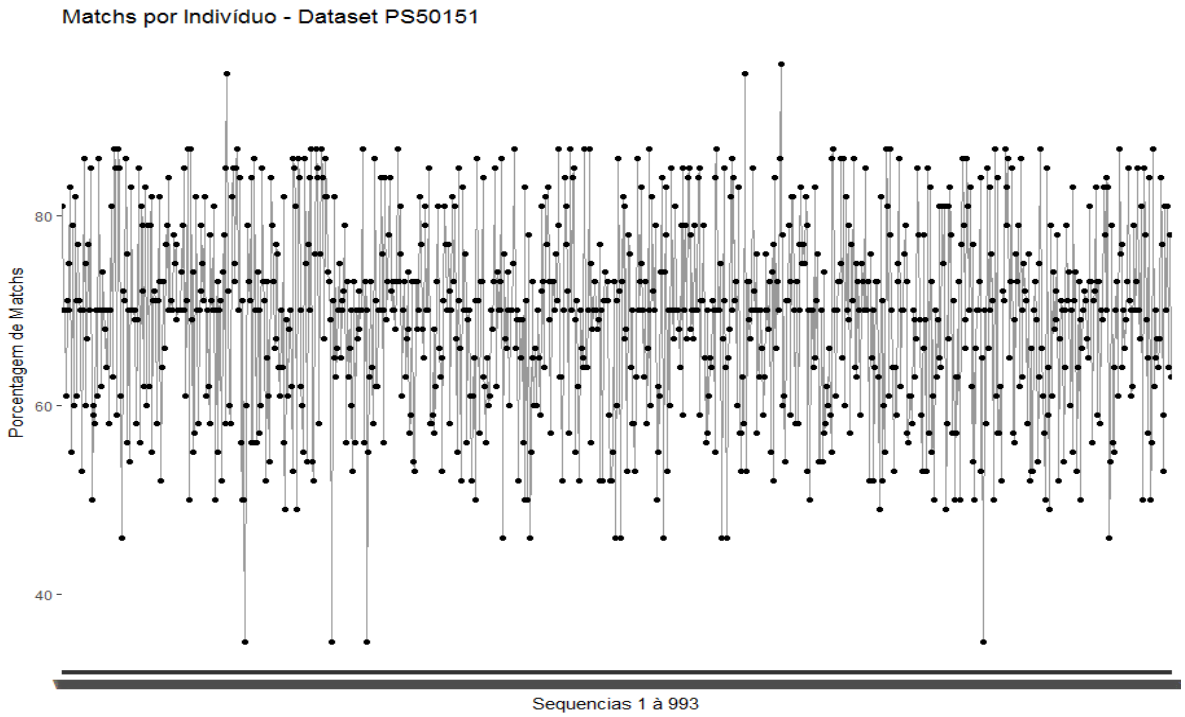

Figura 12: Match por individuo - dataset PS50151.

Na Figura 13 pode-se analisar a performance da arquitetura com 10 gerações e 100 indivíduos com o dataset PS00211, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 1,11. Demorando 22 minutos para retornar os resultados.

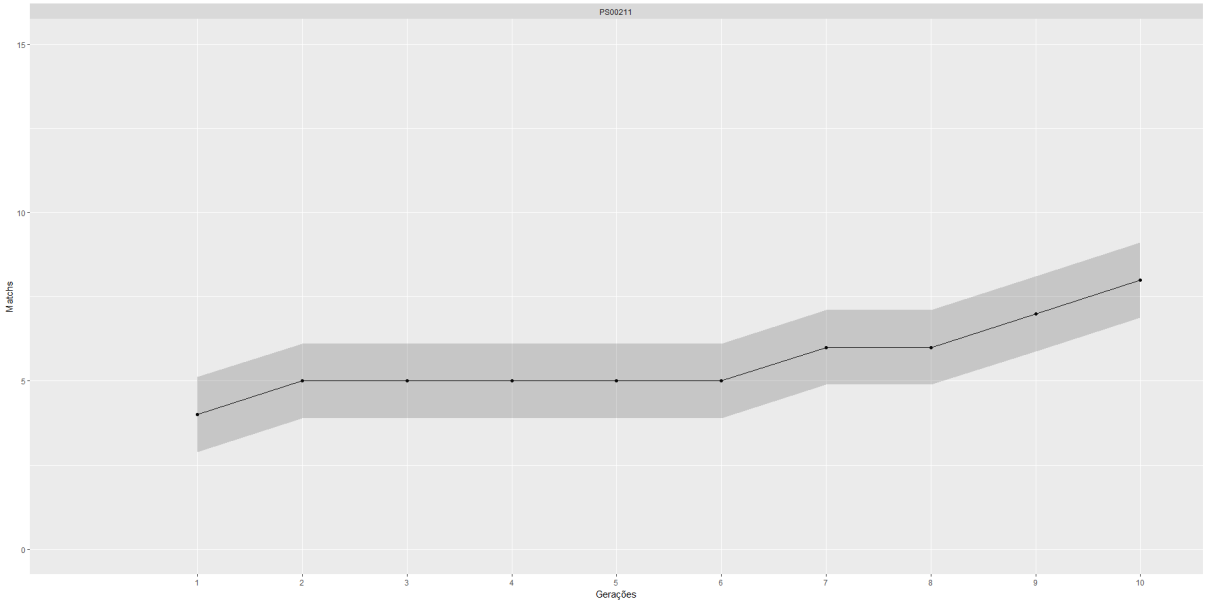

Figura 13: Performance 10 gerações 100 indivíduos - dataset PS00211.

Na Figura 14 pode-se analisar a performance da arquitetura com 25 gerações e 250 indivíduos com o dataset PS00211, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 2,40. Demorando 1 hora e 40 minutos para retornar os resultados.

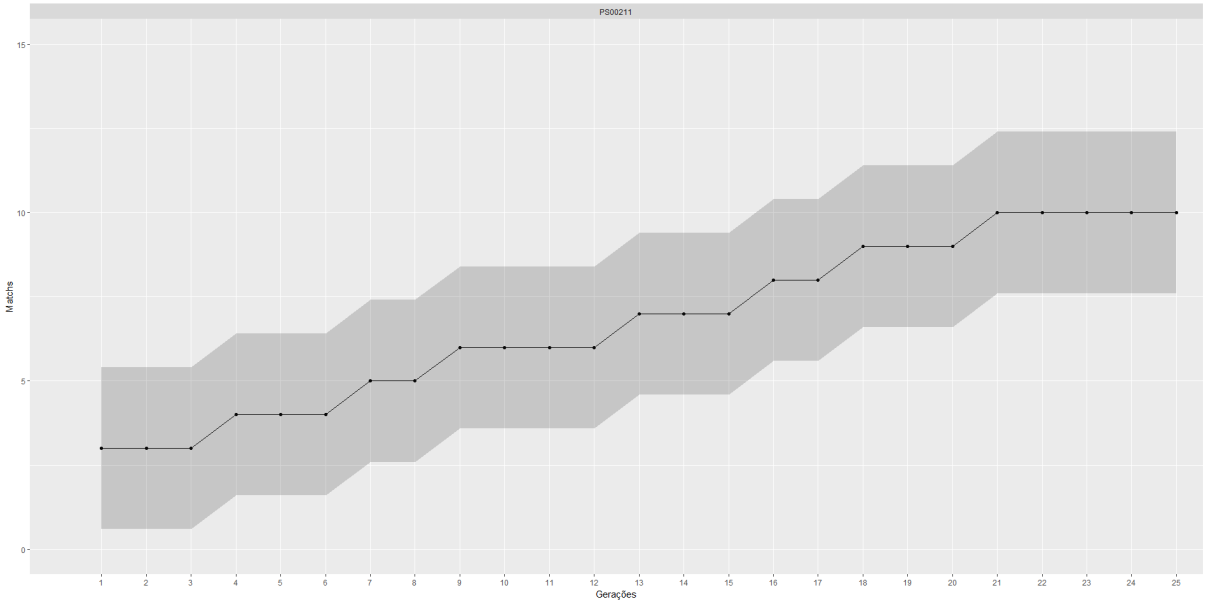

Figura 14: Performance 25 gerações 250 indivíduos - dataset PS00211.

Na Figura 15 pode-se analisar a performance da arquitetura com 50 gerações e 500 indivíduos com o dataset PS00211, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 2,95. Demorando 4 horas e 31 minutos para retornar os resultados.

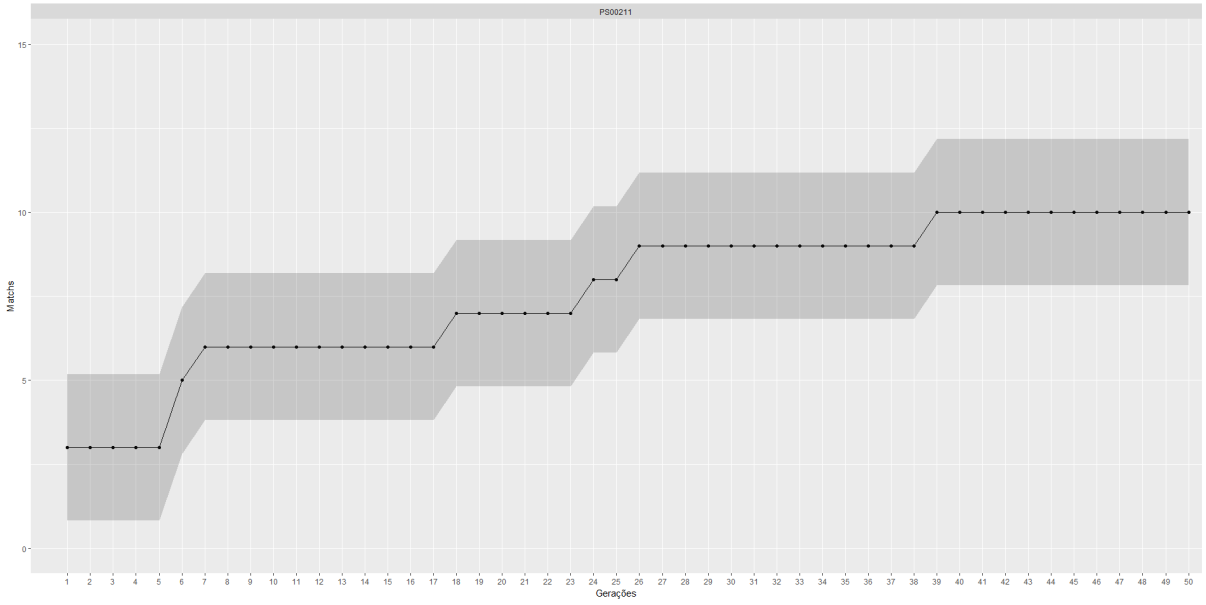

Figura 15: Performance 50 gerações 500 indivíduos - dataset PS00211.

Na Figura 16 pode-se analisar a performance da arquitetura com 75 gerações e 750 indivíduos com o dataset PS00211, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi 3,05. Demorando 6 horas e 32 minutos para retornar os resultados.

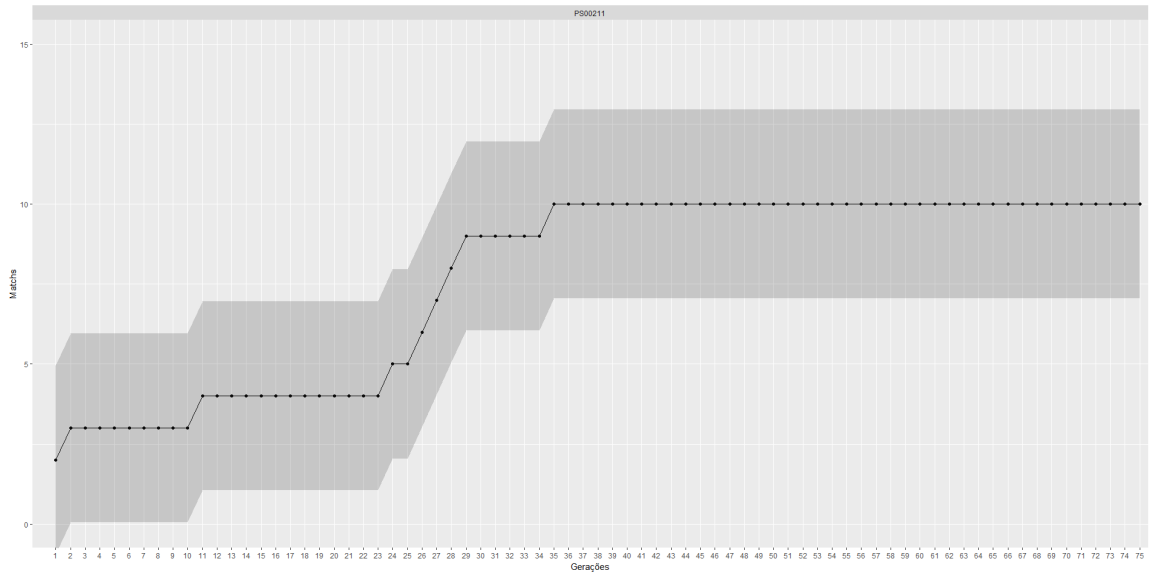

Figura 16: Performance 75 gerações 750 indivíduos - dataset PS00211.

Na Figura 17 pode-se analisar a performance da arquitetura com 100 gerações e 1000 indivíduos com o dataset PS00211, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 1,83. Demorando 10 horas e 12 minutos para retornar os resultados.

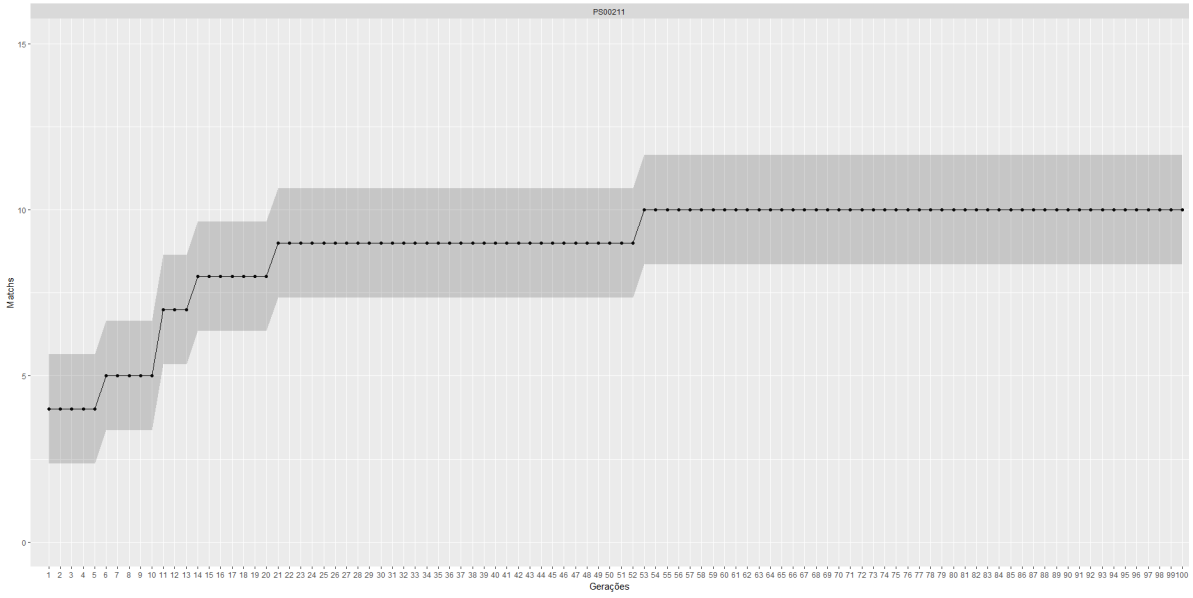

Figura 17: Performance 100 gerações 1000 indivíduos - dataset PS00211.

Na Figura 18 pode-se analisar a performance da arquitetura com 10 gerações e 100 indivíduos com o dataset PS50151, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 3,70. Demorando 47 minutos para retornar os resultados.

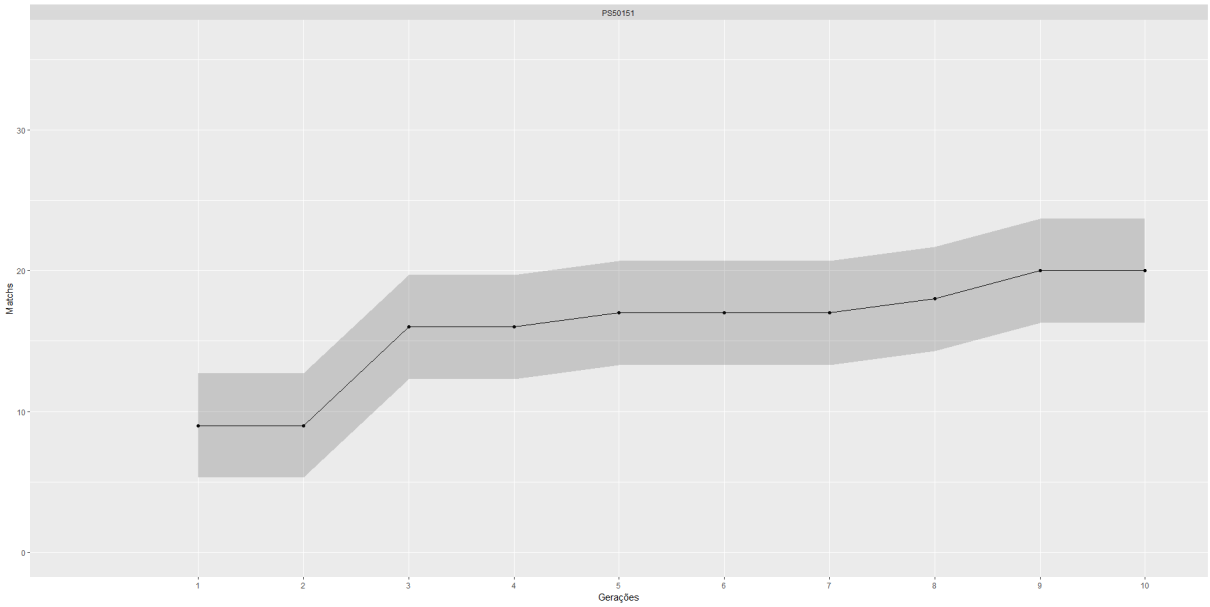

Figura 18: Performance 10 gerações 100 indivíduos - dataset PS50151.

Na Figura 19 pode-se analisar a performance da arquitetura com 25 gerações e 250 indivíduos com o dataset PS50151, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 4,36. Demorando 2 horas e 58 minutos para retornar os resultados.

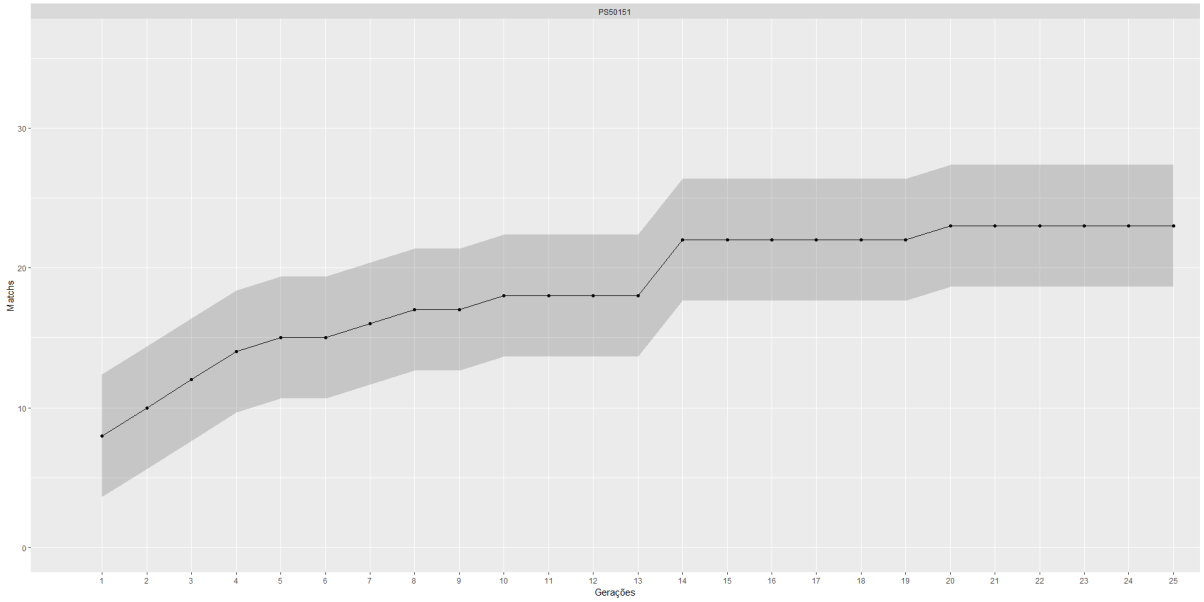

Figura 19: Performance 25 gerações 250 indivíduos - dataset PS50151.

Na Figura 20 pode-se analisar a performance da arquitetura com 50 gerações e 500 indivíduos com o dataset PS50151 , mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 3,93. Demorando 6 horas e 43 minutos para retornar os resultados.

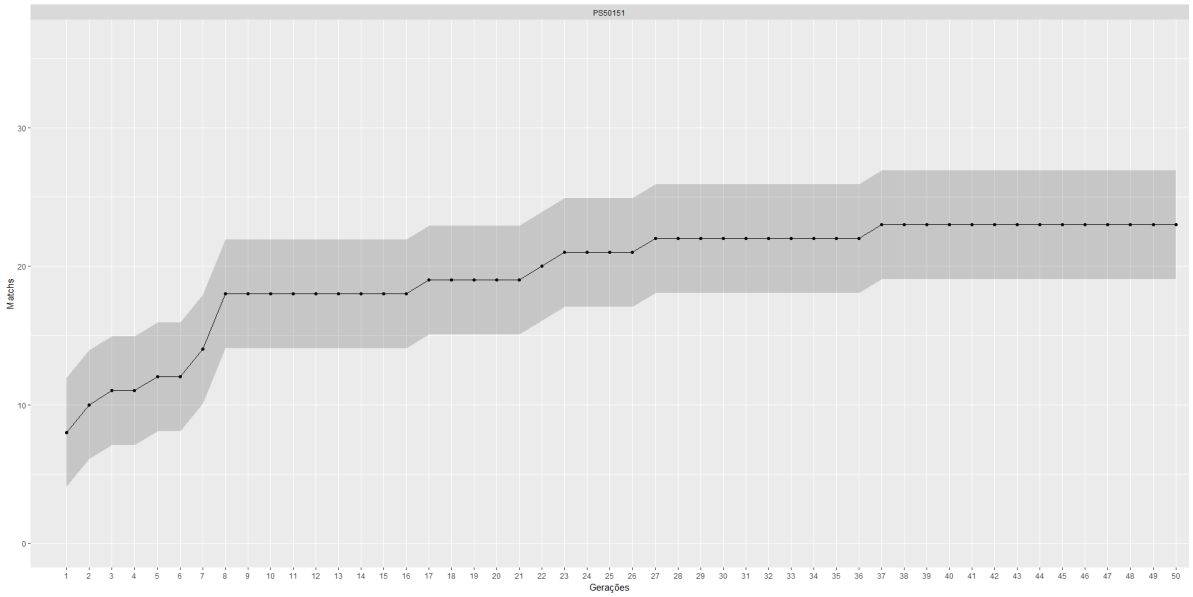

Figura 20: Performance 50 gerações 500 indivíduos - dataset PS50151.

Na Figura 21 pode-se analisar a performance da arquitetura com 75 gerações e 750 indivíduos com o dataset PS50151, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 3,67. Demorando 13 horas e 9 minutos para retornar os resultados.

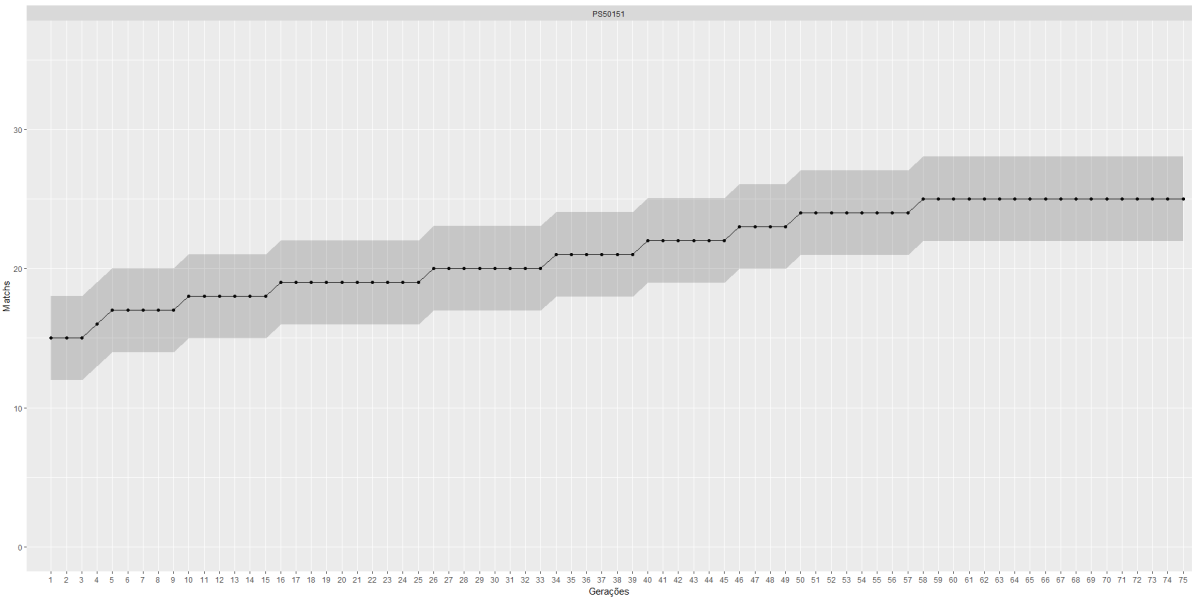

Figura 21: Performance 75 gerações 750 indivíduos - dataset PS50151.

Na Figura 22 pode-se analisar a performance da arquitetura com 100 gerações e 1000 indivíduos com o dataset PS50151, mostrando a média de motivos encontrados por geração, onde o desvio padrão encontrado foi de 3,75. Demorando 28 horas e 21 minutos para retornar os resultados.

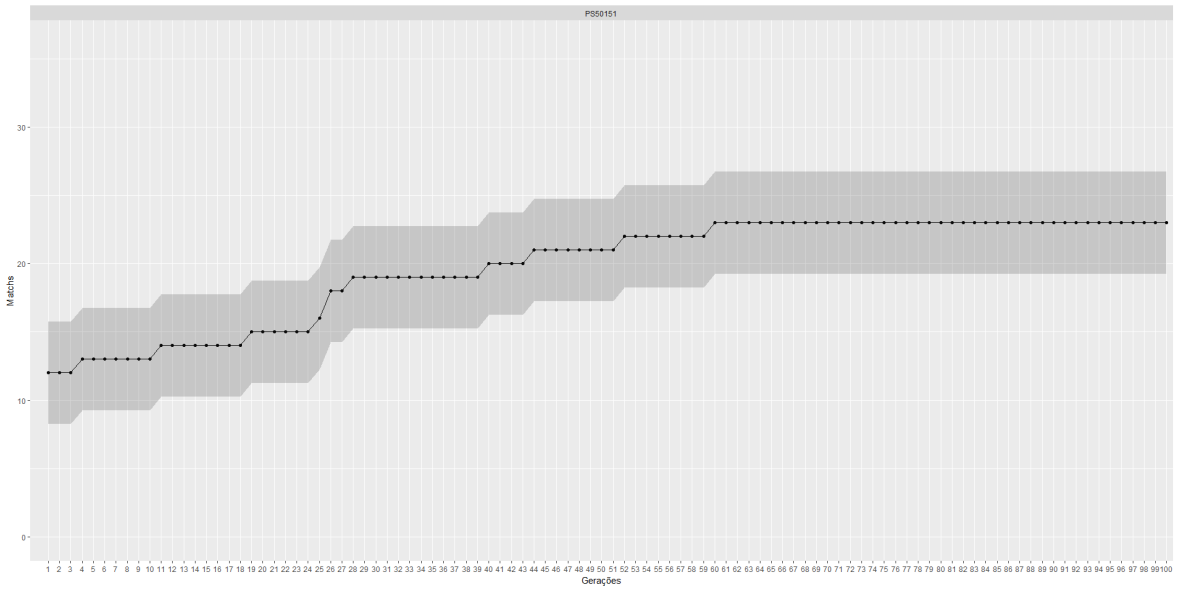

Figura 22: Performance 100 gerações 1000 indivíduos - dataset PS50151.

4.3 Análise Estatística dos Resultados

Foram utilizadas as seguintes fórmulas para análise dos dados:

Variância amostral

A variância de uma amostra $s^2 = \{x_1, \ldots, x_n\}$ de n elementos é determinada pela soma ao quadrado dos desvios dos elementos com relação à média m dividido por (n-1). A equação 5 nos mostra como é calculada a variância amostral.

$$s^2 = \sum_{i=1}^{n} \frac{(x_i - m)^2}{n-1} \tag{5}$$

Desvio padrão amostral

O desvio padrão amostral σ dos dados é a raiz quadrada da variância amostral. A equação 6 nos mostra como é calculado o desvio padrão amostral.

$$\sigma = \sqrt{\sigma^2} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \square)^2}{n-1}} \tag{6}$$

A tabela abaixo nos mostra os resultados estatísticos obtidos utilizando as equações de desvio padrão amostral e variância amostral

Tabela 6: Resultados teste t pareado.

Dataset	Geração / Indivíduos	Variância Amostral (σ)	Desvio padrão Amostral (σ^2)
PS00211	10 / 100	1,3	1,17
PS00211	25 / 250	6,02	2,45
PS00211	50 / 500	4,85	2,20
PS00211	75 / 750	8,84	2,97
PS00211	100 / 1000	2,73	1,65
PS50151	10 / 100	15,21	3,90
PS50151	25 / 250	19,82	4,45
PS50151	50 / 500	15,79	3,97
PS50151	75 / 750	9,33	3,05
PS50151	100 / 1000	14,29	3,77

A tabela abaixo são demonstrados alguns exemplos de resultados obtidos com o *dataset* PS00211 e PS50151 com 75 gerações com 750 indivíduos cada. Onde os resultados variam com um erro de 30% a 35% na detecção do motivo.

Tabela 7: Exemplos de resultados obtidos.

Id/Domínio	Motivo Publicado	Motivo Encontrado
Q6LSC4	**LSGGQQQRLCI**ARTI	TT**LSGGQQQRLCI**TS
P57031	LSGG**ERQRAAIARAL**	TGG**ERQRAAIARAL**A
Q8D3A0	IS**GGERQRAAIA**RSL	SST**GGERQRAAIA**GQ
A2RH10	**LSGGQMRRVAIAG**IL	**LSGGQMRRVAIAG**ST
P08716	LS**GGQRQ**R**IAIARAL**	SA**GGQRQ**TA**AIARAL**
B7MWA0	DQVLTQLIS**RMETASQNLEFEEAA RIRDQIQAVRRV**	GTIALEAAE**RMETASQNLEFE EAARIRDQIQAVRRV**
Q928A4	DIF**IEGMEHEMKE**AA**KALDFER**AA **ELRDALLEIK**AE	EEAS**EGMEHEMKE**AA**KALDF ER**SD**ELRDALLEIK**EE
Q9WYA3	**EEVFDYLKEKMETHSKMLDFENA AKYR**DLLLNLSNV	**EEVFDYLKEKMETHSKMLDFE NAAKYR**DDQQQLNAG
P94846	**EKIIKE**LDKKMRECTKN**LDFEEAM RLRDEIAQLRTL**	**EKIIKE**EAATEATTEGS**LDFEE AMRLRDEIAQLRTL**
Q8AA95	EKSI**ERTRKLMQEAA**KKLE**FIEAA QYRNELLK**LEDL	TEAT**ERTRKLMQEAA**DFVV**FI EAAQYRNELLK**SSTG

Teste T Pareado

Para analisar mais profundamente os conjuntos de dados obtidos como resultado das ferramentas MEME/MAST e AAE-DeMo, foi utilizado o Teste t pareado (BUSSAB e MORETIN, 2002), que serve para analisar dois conjuntos de dados quantitativos em termos de valores médios. Foi definido o seguinte contexto para aplicação do teste:

- Duas populações:

  P1 onde refere-se aos resultados da AAE-DeMo
  P2 onde refere-se aos resultados do MEME/MAST

- Uma variável de interesse, que seria o numero de matchs encontrados.

- As hipóteses:

  H_0 : As ferramentas apresentam os mesmos resultados.
  H_1 : A ferramenta AAE-DeMo teve maior número de matchs encontrados.

Abaixo são demonstrados os resultados obtidos no teste t pareado com o dataset PS00211 e PS50151.

Tabela 8: Resultados Teste T Pareado PS00211.

Grupo	Nº de Amostras	Média	Desvio Padrão	Média de Erro Padrão
AAE-DeMo	4.191	10,24 (68%)	1,83	0,04
MEME/MAST	4.191	8,62 (57%)	1,86	0,04

A média de AAE-DeMo menos MEME/MAST é igual a 1,30, com 95% de confiança no intervalo desta diferença entre 1,28 e 1,32 e o valor P mostrou uma diferença significativa nos resultados de P = 0,00013802.

Tabela 9: Resultados Teste T Pareado PS50151.

Grupo	Nº de Amostras	Média	Desvio Padrão	Média de Erro Padrão
AAE-DeMo	992	24,84 (69%)	3,67	0,12
MEME/MAST	992	22,19 (61%)	3,87	0,12

A média de AAE-DeMo menos MEME/MAST é igual a 3,73, com 95% de confiança no intervalo desta diferença entre 3,65 e 3,73 e o valor P mostrou uma diferença significativa nos resultados de P = 0,00010243.

Comparando os resultados dos testes com um nível de 95% de confiança rejeita-se a hipótese de igualdade entre as variações, ou seja, a arquitetura AAE-

DeMo apresenta uma diferença significativa entre seus resultados e os resultados da ferramenta MEME/MAST.

Portanto, a arquitetura AAE-DeMo apresentou um numero de matchs significativamente superior. Podemos notar que a média de matchs no dataset PS50151 é aproximadamente 25,0 onde que na ferramenta MEME/MAST a média foi de 22,19. E no dataset PS00211 a diferença foi de 10,24 com os resultados da arquitetura AAE-DeMo e a ferramenta MEME/MAST apresentou resultado de 8,62.

O gráfico abaixo mostra a comparação da média de matchs entre a arquitetura AAE-DeMo e a ferramenta MEME/MAST.

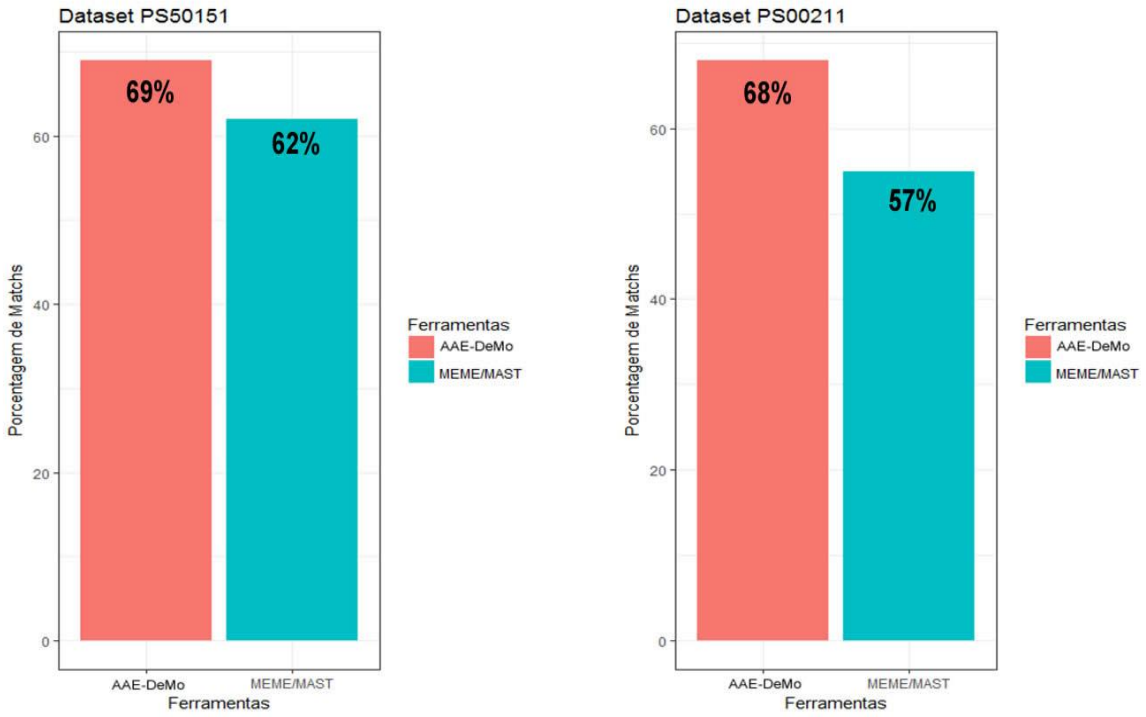

Figura 23: Comparação da Arquitetura Proposta e MEME/MAST.

Também foram comparados os sequence logos dos resultados obtidos da arquitetura AAE-DeMo com a ferramenta MEME/MAST que é consolidada na área de descoberta de motivos fazendo também uma comparação com o logo real dos motivos que realmente existem já catalogados.

Foi utilizada a ferramenta WebLogo para geração dos *sequence logos*. O Gráfico abaixo nos mostra os *sequence logos* gerados para comparação dos resultados da ferramenta e o *logo real* dos datasets.

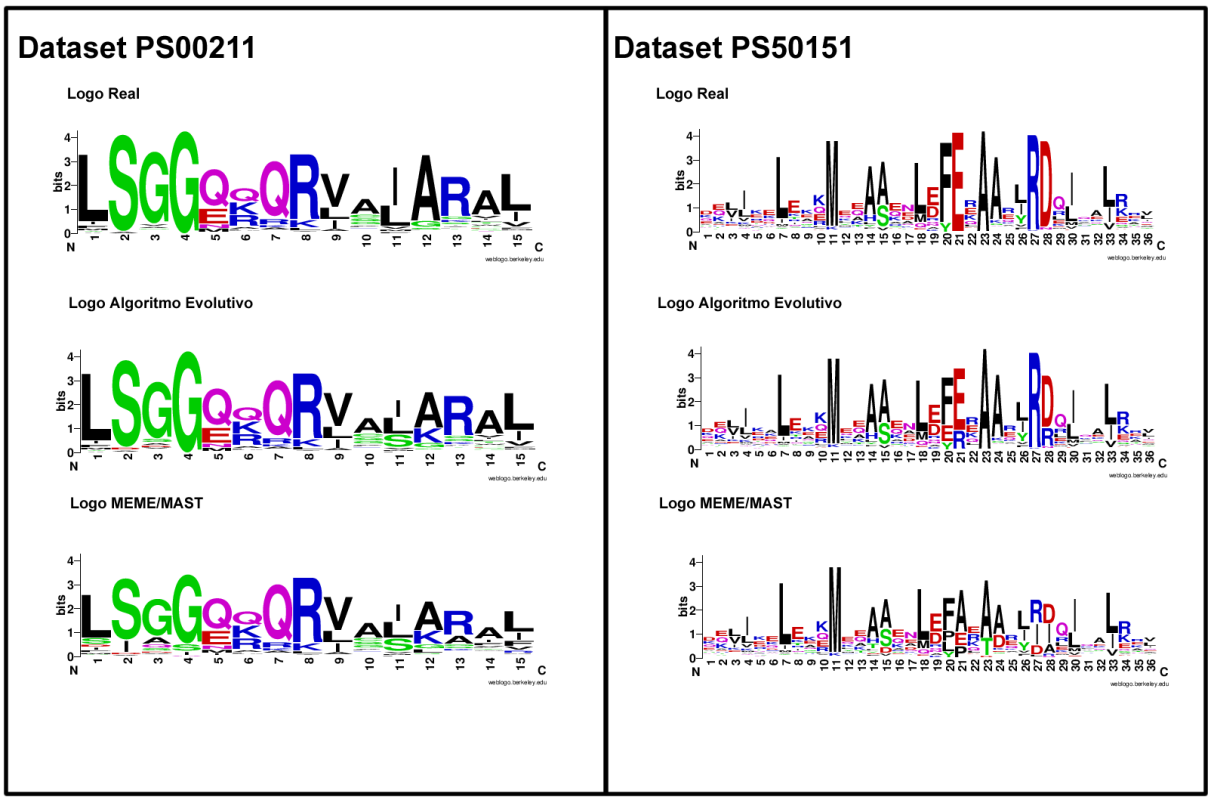

Figura 24: Comparação de Sequence Logos.

5. CONCLUSÃO

Podemos analisar que os resultados obtidos foram plausíveis em todas as etapas de execução deste trabalho, desde o resultado da arquitetura até os testes estatísticos aplicados para validar os resultados da arquitetura, os resultados apresentados pela arquitetura AAE-DeMo diferem dos resultados encontrados pela ferramenta MEME/MAST, foi validado também a diferença entre os resultados da ferramenta e da arquitetura proposta com testes estatísticos significativos. A Ferramenta AAE-DeMo também apresentou uma melhora significativa, no contexto da descoberta de motivos comparados com os resultados da ferramenta MEME/MAST.

Também foi observado que a hibridização do algoritmo evolutivo com uma ferramenta de fitness de bioinformática para prover o fitness do algoritmo evolutivo da arquitetura AAE-DeMo foi uma alternativa válida como metodologia para descoberta de motivos, porém é importante ressaltar que um algoritmo evolutivo com poucas gerações utilizando a função de parada proposta neste trabalho possui grande chance de parada prematura antes de atingir um resultado favorável e também se executado com muitas gerações existe a possibilidade de parada devido à convergência em uma máxima local, ambos os casos foram observados enquanto os testes ocorriam.

Cabe também lembrar que obviamente a área de biologia molecular ainda deve ser estudada, pois até o momento não há como mensurar a precisão nos termos biológicos de cada ferramenta para descoberta de motivos e também porque não se tem uma compreensão clara de um ponto de vista biológico de como os mecanismos regulatórios dos motivos funcionam.

Como muitos estudos, comparando os desempenhos de diferentes ferramentas, sugerem que a melhor opção para encontrar motivos é a utilização de várias ferramentas complementares em pipeline em vez de simplesmente depender de uma só ferramenta (Das, Modan K., 2007).

Portanto, a arquitetura AAE-DeMo mostra resultados razoáveis para descoberta de motivos com uma singela diferença superior ao MEME/MAST: aparentemente a descoberta de motivos com *feedback* de ferramentas de alinhamento ajudam na melhoria do fitness para cada geração do algoritmo evolutivo.

Ainda deve ser melhorado no contexto da aleatoriedade constantemente presente em várias etapas do algoritmo, podendo haver métodos melhores de abordagem para descoberta de motivos. E também devem ser estudados métodos

de paralelização para tornar mais rápido o tempo de entrega de resultados da arquitetura, onde a arquitetura possa analisar varias sequências de uma só vez.

É importante descobrir um modo de se implementar eficazmente o operador de mutação contemplando o contexto biológico aplicado às sequências, de modo que o algoritmo contemple todas as etapas dos algoritmos evolutivos descritas neste trabalho no referencial teórico.

Trabalhos futuros deverão ser realizados para descoberta de motivos de tamanhos diferentes, sendo assim não ficando preso em um tamanho de motif dado de antemão pelo pesquisador.

Outro ponto importante é o desenvolvimento de uma interface intuitiva para que a arquitetura AAE-DeMo possa ser utilizada fora do terminal de comando por qualquer usuário de forma mais simples.

REFERÊNCIAS BIBLIOGRÁFICAS

ALTSCHUL, Stephen F. et al. **Basic local alignment search tool.** Journal of molecular biology, v. 215, n. 3, p. 403-410, 1990.

BAILEY, Timothy L. et al. **Fitting a mixture model by expectation maximization to discover motifs in bipolymers.** 1994.

BAILEY, Timothy L. ; GRIBSKOV, Michael. **Combining evidence using p-values: application to sequence homology searches.** Bioinformatics (Oxford, England), v. 14, n. 1, p. 48-54, 1998.

BARRETT, Steven J. **Intelligent bioinformatics: The application of artificial intelligence techniques to bioinformatics problems.** Genetic Programming and Evolvable Machines, v. 7, n. 3, p. 283-284, 2006.

BAXEVANIS, Andreas D.; OUELLETTE, BF Francis. **Bioinformatics: a practical guide to the analysis of genes and proteins.** John Wiley & Sons, 2004.

BUSSAB, Wilton de O.; MORETTIN, Pedro A. **Estatística básica**. Saraiva, 2010.

CÉSAR, DA SILVA JÚNIOR; SASSON, SEZAR. Biologia-Volume Único 4ª edição. **Editora: Saraiva**, 2007.

CHE, Dongsheng; SONG, Yinglei; RASHEED, Khaled. **MDGA: motif discovery using a genetic algorithm.** In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. ACM, 2005. p. 447-452.

COCK, Peter JA et al. Biopython: **freely available Python tools for computational molecular biology and bioinformatics.** Bioinformatics, v. 25, n. 11, p. 1422-1423, 2009.

CONGDON, Clare Bates et al. **Preliminary results for GAMI: A genetic algorithms approach to motif inference.** In: Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB'05. Proceedings of the 2005 IEEE Symposium on. IEEE, 2005. p. 1-8.

DAS, Modan K.; DAI, Ho-Kwok. A **survey of DNA motif finding algorithms.** BMC bioinformatics, v. 8, n. 7, p. S21, 2007.

Davies, K. (2001). **Decifrando o genoma: a corrida para desvendar o dna humano.** São Paulo: Companhia das Letras. 2001.

FRANCO, Dielle Correa et al. **Desempenho do Algoritmo Genético com Iteração Retroviral para otimização de funções com representação real.** REVISTA EIXO, v. 2, n. 2, p. 13-30, 2013.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning.**1989.

EDGAR, Robert C. **Search and clustering orders of magnitude faster than BLAST.** Bioinformatics, v. 26, n. 19, p. 2460-2461, 2010.

JENSEN, Shane T.; LIU, Jun S. **BioOptimizer: a Bayesian scoring function approach to motif discovery.** Bioinformatics, v. 20, n. 10, p. 1557-1564, 2004.

KARLIN, Samuel; ALTSCHUL, Stephen F. **Applications and statistics for multiple high-scoring segments in molecular sequences.** Proceedings of the National Academy of Sciences, v. 90, n. 12, p. 5873-5877, 1993.

LIU, Falcon FM et al. **FMGA: finding motifs by genetic algorithm.** In: Bioinformatics and Bioengineering, 2004. BIBE 2004. Proceedings. Fourth IEEE Symposium on. IEEE, 2004. p. 459-466.

NOTREDAME, Cédric; HIGGINS, Desmond G. **SAGA: sequence alignment by genetic algorithm.** Nucleic acids research, v. 24, n. 8, p. 1515-1524, 1996.

PARIDA, Laxmi. **Pattern discovery in bioinformatics: theory & algorithms.**CRC Press, 2007.

PROSDOCIMI, Francisco. **Introdução à bioinformática.** Belo Horizonte: Biotecnologia ciência e desenvolvimento, 2007.

PROSDOCIMI, Francisco; Cerqueira GC; Binneck E; Silva AF; Reis AN; Junqueira ACM; Santos ACF; Nhani-Júnior A; Wust CI; Camargo-Filho F; Kessedjian JL; Petretski JH; Camargo LP; Ferreira RGM; Lima RP; Pereira RM; Jardim S; Sampaio VS and Folgueras-Flatschart AV. **Bioinformática: manual do usuário.**Biotec. Ci. Des. 29: 18-31, 2002.

SNUSTAD DP, SIMMONS MJ, JENKINS JB. **Principles of genetics.** John Wiley & Sons Inc.1997.

STOESSER, Guenter et al. **The EMBL nucleotide sequence database.** Nucleic acids research, v. 30, n. 1, p. 21-26, 2002.

TATENO, Yoshio et al. **DNA Data Bank of Japan (DDBJ) for genome scale research in life science.** Nucleic acids research, v. 30, n. 1, p. 27-30, 2002.

TESLA, Nikola. **Radio power will revolutionize the world.** Modern Mechanix and Inventions, p. 2, 1934.

WEI, Zhi; JENSEN, Shane T. **GAME: detecting cis-regulatory elements using a genetic algorithm.** Bioinformatics, v. 22, n. 13, p. 1577-1584, 2006.

Yi-Ping Phoebe Chen. **Bioinformatics Technologies. Springer Science & Business Media**, 2005.