UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
ESCOLA DE ENGENHARIA DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

UMA ABORDAGEM DE ALINHAMENTO MÚLTIPLO DE SEQUÊNCIAS UTILIZANDO EVOLUÇÃO DIFERENCIAL

ANTÔNIO LUIZ VIEIRA DA SILVA JÚNIOR

RECIFE
2015

ANTÔNIO LUIZ VIEIRA DA SILVA JÚNIOR

UMA ABORDAGEM DE ALINHAMENTO MÚLTIPLO DE SEQUÊNCIAS UTILIZANDO EVOLUÇÃO DIFERENCIAL

Dissertação submetida ao Programa de Pós-graduação em Engenharia Biomédica da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de **Mestre em Engenharia Biomédica**.

Orientador: Wellington Pinheiro dos Santos

Co-Orientador: Ricardo Yara

RECIFE

2015

Catalogação na fonte
Bibliotecária Margareth Malta, CRB-4 / 1198

S586a Silva Júnior, Antônio Luiz Vieira da.
Uma abordagem de alinhamento múltiplo de sequências utilizando evolução diferencial / Antônio Luiz Vieira da Silva Júnior. - Recife: O Autor, 2015.
69 folhas, il., gráfs., tabs.

Orientador: Prof. DSc. Wellington Pinheiro dos Santos.
Coorientador: Prof. Dr. Ricardo Yara.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Biomédica, 2015.
Inclui Referências.

1. Engenharia Biomédica. 2. Alinhamento múltiplo de sequência. 3. Bioinformática. 4. Computação evolutiva. 5. Evolução diferencial. 6. Algoritmo genético. I. Santos, Wellington Pinheiro dos. (Orientador). II. Yara, Ricardo. III. Título.

UFPE

610.28 CDD (22. ed.) BCTG/2015-161

PROPESQ
Pró-Reitoria para Assuntos de Pesquisa e Pós-Graduação

Universidade Federal de Pernambuco

Ata Da Defesa De Dissertação De Mestrado, Realizada No Programa De Pós-Graduação Em ***ENGENHARIA BIOMÉDICA*** *do Centro de* ***TECNOLOGIA E GEOCIÊNCIAS*** *da Universidade Federal de Pernambuco, no dia 27 de FEVEREIRO 2015*

Aos 27 (vinte e sete) dias do mês de fevereiro de dois mil e quinze (2015), às 14 horas, no Laboratório da Computação Biomédica do Programa de Pós-graduação em Engenharia Biomédica da Universidade Federal de Pernambuco, reunida a Banca Examinadora, em sessão pública, teve início a defesa de Dissertação intitulada "**UMA ABORDAGEM DE ALINHAMENTO MÚLTIPLO DE SEQUENCIAS UTILIZANDO EVOLUÇÃO DIFERENCIAL**", do (a) aluno (a) **Antônio Luiz Vieira da Silva Júnior**, que havia cumprido anteriormente todos os demais requisitos regimentais para a obtenção do grau de Mestre em Engenharia Biomédica. A Banca Examinadora foi composta pelo Prof. Dr. Ricardo Emmanuel de Souza, do Departamento de Engenharia Biomédica da Universidade Federal de Pernambuco - UFPE, prof. Dr. Sérgio Ricardo de Melo Queiroz, do Departamento de Ciência da Computação da UFPE, e o Prof. Dr. Wellington Pinheiro dos Santos, do Departamento Engenharia Biomédica-UFPE, na qualidade de presidente. Depois de cumpridas as formalidades, o candidato foi convidado a discorrer sobre o conteúdo da Dissertação. Concluída a explanação, o candidato foi argüido pela Banca Examinadora que, em seguida, reuniu-se para deliberar e conceder ao mesmo a menção (Aprovado/Reprovado) APROVADO da referida Dissertação. E, para constar, lavrei a presente Ata que vai por mim assinada, Secretária de Pós-Graduação, e pelos membros da Banca Examinadora e demais presentes.

Recife, 27 de fevereiro de 2015.

Resumo

Alinhamento Múltiplo de sequências (MSA) é uma das tarefas mais importantes em bioinformática. A MSA é uma técnica fundamental para o estudo da função, estrutura e evolução de biomoléculas. A partir do uso de métodos de MSA é possível a criação de modelos estatísticos para a classificação de famílias de proteína , análise filogenética e a previsão de estruturas secundárias de proteínas. Como trata-se de um problema do tipo NP-difícil, torna-se inviável o uso de métodos exatos para a busca da melhor solução. Por isso, é importante o uso de métodos de optimização baseado em heurística para resolver o problema de MSA. Nesta dissertação, propomos uma abordagem para alinhamento múltiplo de sequências por meio da otimização de uma função objetivo utilizando Evolução Diferencial. Embora a ideia de usar algoritmos evolutivos não seja nova, a abordagem apresentada difere pelo uso da Evolução Diferencial e pela definição do alinhamento como uma dispersão de lacunas ao longo das sequências, sem levar em consideração fenômenos biológicos, como os de inserção ou surgimento de bases, deleção ou mutação de bases. A solução proposta tem provado ser capaz de fazer melhorias significativas em alinhamentos quando comparadas com o método do estado da arte Clustal.

Palavras-chave: Alinhamento múltiplo de sequência. Bioinformática. Computação evolutiva. Evolução diferencial. Algoritmo genético.

Abstract

Multiple sequence alignment (MSA) is one of the most important tasks in bioinformatics. The MSA is a fundamental technique to the study of function, structure and evolution of biomolecules. By using of MSA methods it's possible to create statistical models for classification of protein families, phylogenetic analysis and the prediction of secondary structures of proteins. Being a NP-hard problem, it is infeasible due to its completely, the use of exact methods to search for optimal solutions. Because of this it is important to use heuristic-based optimization methods to solve the MSA problem. In this dissertation, we propose an approach to multiple sequence alignment by optimizing an objective function using Differential Evolution. Although the idea of using Evolutionary Algorithms is not new, the approach presented differs from the use of Differential Evolution and definition of alignment as a dispersion of gaps along the sequences, without considering biological events such as insertion or emergence of bases, deletion or mutation of bases. The proposed solution has proven to be able to make significant improvements in alignments when compared to the state-of-the art Clustal method.

Keywords: Multiple sequence alignment. Bioinformatics. Evolutionary computing. Differential evolution. Genetic algorithm.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Acrônimos

AE	Algoritmo Evolucionário
B1B	Best/1/Bin
B2B	Best/2/Bin
BLAST	Basic Local Alignment Search Tool
CDE	Chaos-differential evolution
CRS	Controlled Random Search
DE	Differential Evolution
EP	Evolutionary Programming
GA	Genetic Algorithmns
GP	Genetic Programming
MSA	Multiple Sequence Alignment
PWM	Positional Wight Matrix
PSO	Particle swarm optimization
R1B	Rand/1/Bin
R2B	Rand/2/Bin
RB1B	Rand-To-Best/1/Bin

Sumário

1

INTRODUÇÃO

O alinhamento múltiplo de sequências (*Multiple Sequence Alignment*, MSA) é um dos mais fundamentais e mais desafiadores problemas em biologia computacional. O problema do MSA pode ser descrito como o arranjo de três ou mais sequências de nucleotídeos ou aminoácidos, sobrepostas. Este arranjo é obtido pelo deslocamentos dos elementos destas sequências obtidos pela inserção de espaços vazios, chamados de lacunas (*gaps*). O MSA é uma técnica utilizada para o estudo da função, estrutura e evolução de biomoléculas. Algumas das aplicações do MSA estão na análise filogenética (GUSFIELD, 1997), na criação de modelos estatísticos para classificação de famílias de proteínas (SIPPL, 1999), na predição de estruturas secundárias de proteínas (JONES, 1999) e em ferramentas de extração de similaridade entre sequências (ALTSCHUL et al., 1997).

Classificado como um problema do tipo NP-difícil (CARRILLO; LIPMAN, 1988; NOTREDAME, 2002), a complexidade do MSA torna inviável o uso de métodos exatos para a busca de soluções ótimas. Portanto, são usualmente utilizadas as seguintes estratégias: Alinhamentos Progressivos (FENG; DOOLITTLE, 1987), destacando-se o Clustal (THOMPSON; HIGGINS; GIBSON, 1994; THOMSEN; FOGEL; KRINK, 2002; HIGGINS; BLEASBY; FUCHS, 1992), MUSCLE (EDGAR, 2004a,b) e T-COFFE (NOTREDAME; HIGGINS; HERINGA, 2000); e os Algorítimos Iterativos onde se destaca a utilização de meta-heurísticas, como as dos Algoritmos Evolucionários (AE) (DORTMUND, 1995), sendo SAGA (NOTREDAME; HIGGINS, 1996), COFFE

(NOTREDAME; HOLM; HIGGINS, 1998) e PWMAligner (MASULLI; PETERSON; TAGLIAFERRI, 2010) os mais conhecidos.

Inspirados nos princípios da evolução natural de Darwin e na genética, os Algoritmos Evolucionários (AE) são direcionados a encontrar ótimos globais, evitando a parada prematura em ótimos locais. Existem várias formas de AEs, dentre o que se aplicam a este trabalho temos: Algoritmos Genéticos (*Genetic Algorithmns*, GA) (GOLDBERG, 1989) e os algoritmos de Evolução Diferencial (*Differential Evolution*, DE) (STORN; PRICE, 1997; PRICE; STORN; LAMPINEN, 2005). A DE se destaca pela sua facilidade de uso, rápida velocidade de convergência e poucos parâmetros de controle.

1.1 Objetivo geral

Neste trabalho, propomos uma abordagem para alinhamento múltiplo de sequências por meio da otimização de uma função objetivo utilizando Evolução Diferencial. Embora a idéia de usar algoritmos evolutivos não seja novo, a abordagem apresentada difere pelo uso da Evolução Diferencial.

1.2 Objetivos específicos

a) Propor uma nova representação baseada somente nas posições das lacunas

b) Discretização da DE para a aplicação em um problema de otimização combinatória

1.3 Estrutura do documento

No Capítulo 2, é apresentado o conceito de Alinhamento de Sequências, listando as técnicas canônicas de alinhamento, os tipos de alinhamentos de sequências e a classificação das técnicas de alinhamento múltiplo de sequências.

No Capítulo 3, é apresentado o conceito de Algoritmos Evolucionários assim como sua definição algorítmica, uma descrição de seus passos computacionais e por fim

duas formas de Algoritmos Evolucionários relacionados a esta dissertação: Algoritmos Genéticos e Evolução Diferencial. Para cada uma destas duas formas de Algoritmos Evolucionários são abordadas sua definição algorítmica canônica e uma descrição dos seus passos computacionais característicos.

No Capítulo 4, temos a descrição da metodologia empregada. Aqui são apresentados como foram obtidas as amostras utilizadas nos experimentos e detalhes do protocolo de experimentos.

No Capítulo 5, são apresentados os resultados obtidos e uma breve discussão sobre os resultados atingidos é realizada.

No Capítulo 6, temos uma breve apresentação das contribuições, publicações e possibilidades de trabalhos futuros.

2

ALINHAMENTO DE BIOSEQUÊNCIAS

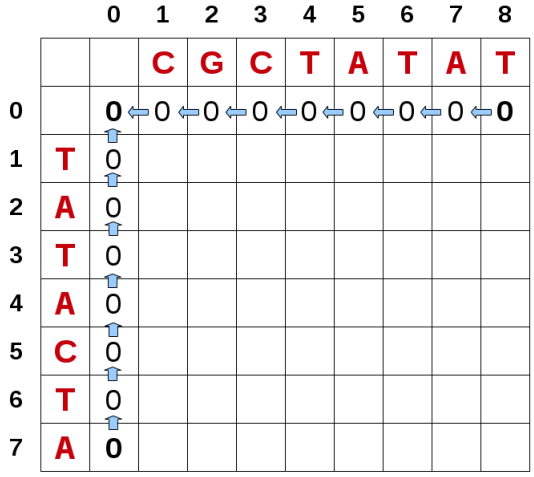
No início do século XXI, a Biologia Molecular desenvolveu-se rapidamente e novos métodos e técnicas de sequenciamento de genes (PLUS AND MINUS METHOD , SANGER F ET AL 1975 J MOL BIOL 94:441) foram criados. O avanço tecnológico, associado a esta área da ciência, permitiu o levantamento de informação genômica de várias espécies, incluindo o do código genético humano, microrganismos como bactérias, fungos e vírus, assim como de espécies de interesse econômico. Vários tipos de estudos podem ser realizados a partir da relação das informações obtidas pelos sequenciamentos. Para a realização destas relações entre sequências é necessária a utilização de técnicas de alinhamento entre sequências.

O alinhamento de sequências consiste no processo de comparar duas sequências (de nucleotídeos ou proteínas) de forma a se observar seu nível de identidade. O alinhamento entre duas sequências pode ser feito de forma global ou local.

2.1 Alinhamento Global entre duas sequências

No Alinhamento Global a comparação entre as sequências é realizada ao longo de toda a extensão da sequência. O algoritmo Needleman-Wunsch (NEEDLEMAN; WUNSCH, 1970) é o mais conhecido para realizar esse tipo de alinhamento. Considerado como um dos primeiros métodos de alinhamento o algoritmo de Needleman-Wunsch faz

Figura 2.1: Needleman-Wunsch - Fase de inicialização.



uso da técnica de programação dinâmica para encontrar de forma direta e ótima um alinhamento entre duas sequências biomoleculares. A execução do algoritmo ocorre em três etapas: inicialização, preenchimento da matriz (*scoring*) e Alinhamento (*traceback*).

Vamos considerar duas sequências ($A = [TATACTA]$ e $B = [CGCTATAT]$) como exemplo para a realização do alinhamento global a partir da aplicação da técnica de Needleman-Wunsch. As constantes m e n são respectivamente os comprimentos das sequências A e B, portanto $m = 7$ e $n = 8$. O primeiro passo do algoritmo é criar uma matriz com $m+1$ linhas e $n+1$ colunas, ou seja, para as sequências A e B dadas como exemplo teremos uma matriz 8×9. Uma inicialização desta matriz é realizada de acordo com a condição inicial estabelecida na Equação 2.1, onde $F(i, j)$ é a Função de Pontuação para preenchimento da matriz de pontuação $m+1 \times n+1$.

$$\text{Condição Inicial de Needleman-Wunsch} = \begin{cases} F(0,0) = 0 \\ F(i,0) = -id \\ F(0,j) = -jd \end{cases} \tag{2.1}$$

O resultado da inicialização respeitando a condição inicial da Equação 2.1, pode ser vista na Figura 2.1.

Para a segunda etapa (*scoring*) são aplicadas a Função de Pontuação $F(i, j)$ e uma Matriz de Mérito $W(A_i, B_j)$ é consultada. A Função de pontuação para o algoritmo de

Needleman-Wunsch é definida na Equação 2.2.

$$F(i,j) = \max\{F(i-1,j-1) + W(A_i,B_j),\ F(i-1,j) - d,\ F(i,j-1) - d\} \tag{2.2}$$

onde:

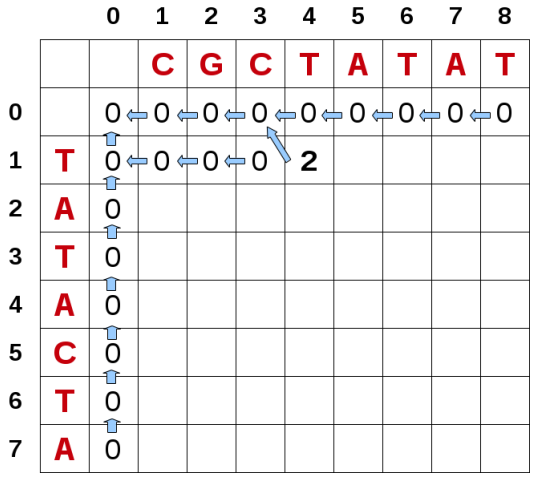
- d é uma constante que penaliza a existência de lacunas nos alinhamentos. Consideramos que $d = 0$;
- $W(A_i,B_j)$ é a Matriz de Mérito que retorna o score correspondente ao par de elementos (A_i,B_j).
- A_i é o i-ésimo elemento da sequência A e B_j é o j-ésimo elemento da sequência B.

A Matriz de Mérito tem como objetivo auxiliar na comparação das similaridades (*matches*) e diferenças (*mismatches*) entre as duas sequências. A Matriz de Mérito é construída a partir da relação logarítmica entre as probabilidades de que um par de ácidos nucleicos (ou aminoácidos, se for o caso) estejam alinhados e a probabilidade de que o par se apresente independente. A Equação 2.3 apresenta esta relação, onde a e b são os ácidos nucleicos das duas sequências, p_{ab} é a probabilidade dos elementos aparecerem ao mesmo tempo, q_a e q_b são as probabilidades independentes de cada aminoácido e $S(a,b)$ é a razão logarítmica entre essas probabilidades. $S(a,b)$ é a chamada razão *log-odds* e descreve a probabilidade de que a e b sejam achados alinhados em vez de serem achados não alinhados (SIPPL, 1999).

$$S(a,b) = \log(\frac{p_{ab}}{q_a q_b}) \tag{2.3}$$

Para o exemplo de alinhamento local que iremos executar a seguir será considerada a Matriz de Mérito da Equação 2.4, tendo seus valores definidos somente para efeito de exemplo do algoritmo.

Figura 2.2: Needleman-Wunsch - Fase de preenchimento (*scoring*).



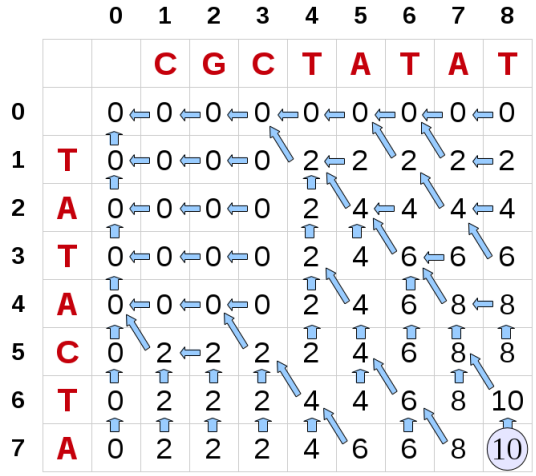
$$W = \begin{array}{c} \\ \mathbf{A} \\ \mathbf{C} \\ \mathbf{T} \\ \mathbf{G} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{A} & \mathbf{C} & \mathbf{T} & \mathbf{G} \end{array} \\ \begin{pmatrix} 2 & -1 & -1 & -1 \\ -1 & 2 & -1 & -1 \\ -1 & -1 & 2 & -1 \\ -1 & -1 & -1 & 2 \end{pmatrix} \end{array} \tag{2.4}$$

A partir da Função de Pontuação e Matriz de Mérito é realizada o preenchimento do resto da matriz $m+1 \times n+1$ como mostra a Figura 2.2. Como exemplo, para $F(1,4)$ teremos: Se $F(1-1,4-1)+W(A_1,B_4) = F(0,3)+W(T,T) = 0+2 = 2$, $F(1-1,4)-d = F(0,4)-0 = 0-0 = 0$, $F(1,4-1)-d = F(1,3)-0 = 0-0 = 0$, e $\max\{2,0,0\} = 2$, logo $F(1,4) = 2$.

A matriz totalmente preenchida caracterizando o término da fase de *scoring* pode ser visualizada na Figura 2.3. Analisando a Figura 2.3 é possível identifica o valor 10 como a pontuação máxima possível do alinhamento global entre as duas sequências de exemplo. Após o término da etapa de *scoring* a pontuação máxima sempre será encontrada na parte inferior direita da matriz, ou seja, na posição (m,n).

A última etapa do algoritmo de Needleman-Wunsch é a etapa de *traceback*. O *traceback* permite encontrar o alinhamento que resulta na pontuação máxima alcançada. É possível encontrar casos onde existam vários alinhamentos possíveis com a mesma pontuação máxima.

Figura 2.3: Needleman-Wunsch - Fim da fase de preenchimento (*scoring*).



Para construir o alinhamento, o *traceback* começa na posição (m,n) da matriz (nesta caso existe um 10 nesta posição) e olhamos para as células vizinhas da posição atual. Dentre estes vizinhos esta o vizinho precedente direto da posição atual. Identificando o vizinho precedente deve-se levar em consideração as seguintes regras para o alinhamento:

- se o elemento precedente esta acima da posição atual então A_i alinha com lacuna;
- se o elemento precedente esta na diagonal da posição atual então B_j alinha com A_i;
- se o elemento precedente esta a esquerda da posição atual então B_j alinha com lacuna.

Após o término do processo iterativo do *traceback* na posição $(0,0)$ da matriz, é possível identificar todo o caminho característico para o alinhamento que será gerado, como mostra a Figura 2.4. O alinhamento resultado utilizando as informações da fase de *traceback* pode ser visualizado na Figura 2.5.

Com o alinhamento resultante em mãos é possível constatar que o valor na posição (m,n) (Valor Máximo) é dado pela soma das parcelas correspondentes à pontuação de cada alinhamento individual utilizando a Matriz de Mérito: $W(A,-)+W(T,T)+W(-,C)+W(A,A)+W(T,T)+W(A,A)+W(T,T)+W(C,-)+W(G,-)+W(C,-)=0+0+0+2+2+2+2+0+2+0=10$.

Figura 2.4: Needleman-Wunsch - Fim da fase de alinhamento (*traceback*).

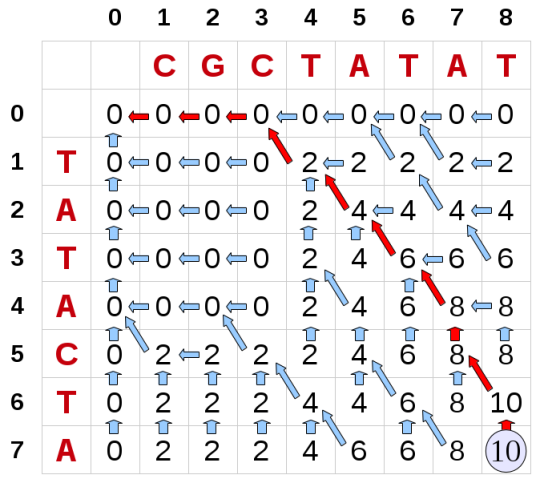


Figura 2.5: Needleman-Wunsch - Alinhamento Resultante.

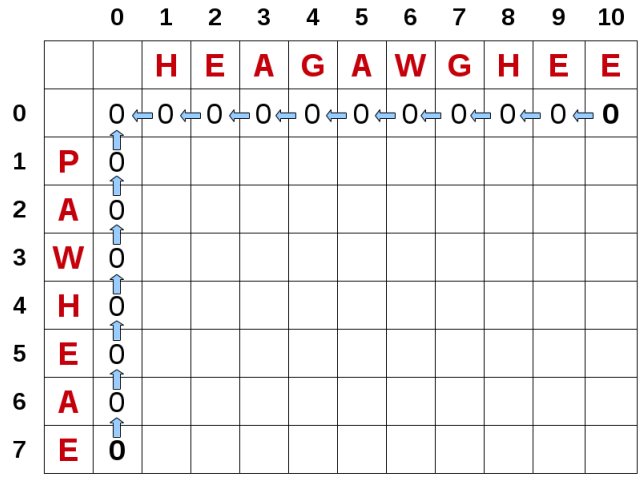
A	-	-	-	T	A	T	A	C	T	A
B	C	G	C	T	A	T	A	-	T	-

2.2 Alinhamento Local entre duas sequências

No Alinhamento Local a comparação entre as sequências não é realizada ao longo de toda a extensão da sequência, mas sim em pequenas regiões. O algoritmo Smith-Waterman (SMITH; WATERMAN, 1981) é o mais conhecido para realizar esse tipo de alinhamento. Também faz uso da técnica de programação dinâmica de forma similar ao Algorirmo de Nedleman-Wunsch para encontrar de forma direta e ótima um alinhamento local entre duas sequências biomoleculares. A execução do algoritmo ocorre também em três etapas: inicialização, preenchimento da matriz (*scoring*) e alinhamento (*traceback*).

Vamos considerar duas sequências de aminoácidos ($A = [PAWHEAE]$ e $B = [HEAGAWGHEE]$) como exemplo para a realização do alinhamento local a partir da aplicação da técnica de Smith-Waterman. As constantes m e n são respectivamente os comprimentos das sequências A e B, portanto $m = 7$ e $n = 10$. O primeiro passo do algoritmo é criar uma matriz com $m+1$ linhas e $n+1$ colunas, ou seja, para as sequências A e B dadas como exemplo teremos uma matriz 8×11. Uma inicialização desta matriz é realizada de acordo com a condição inicial estabelecida na Equação 2.5, onde $F(i, j)$ é a Função de Pontuação para preenchimento de uma matriz de pontuação $m+1 \times n+1$.

Figura 2.6: Smith-Waterman - Fase de inicialização.



$$\text{Condição Inicial de Smith-Waterman} = \begin{cases} F(0,j) = 0, \\ F(i,0) = 0 \end{cases} \tag{2.5}$$

O resultado da inicialização respeitando a condição inicial da Equação 2.5, pode ser vista na Figura 2.6.

Para a segunda etapa (*scoring*) são aplicadas a Função de Pontuação $F(i,j)$ e uma Matriz de Mérito $W(A_i,B_j)$ é consultada. A Função de pontuação para o algoritmo de Smith-Waterman é definida na Equação 2.6.

$$F(i,j) = \max\{0,\ F(i-1,j-1) + W(A_i,B_j),\ F(i-1,j) - d,\ F(i,j-1) - d,\} \tag{2.6}$$

onde:

- d é uma constante que penaliza a existência de lacunas nos alinhamentos. Para o exemplo, vamos considerar que $d = 8$;
- $W(A_i,B_j)$ é a Matriz de Mérito que retorna o score correspondente ao par de elementos (A_i,B_j).
- A_i é o i-ésimo elemento da sequência A e B_j é o j-ésimo elemento da sequência B.

Para o exemplo de alinhamento local é adotada a matriz BLOSUM50 (HENIKOFF; HENIKOFF, 1992) como Matriz de Mérito. A matriz BLOSUM50 pode ser visualizada na Equação 2.7. Para fins de agilidade na consulta e melhor compreensão a Matriz de Mérito será representada somente pelos elementos presentes nas sequências exemplos x e y, sendo $W(A_i, B_j)$ definido finalmente na Equação 2.8.

$$
\text{BLOSUM50} = \begin{array}{c} \\ A \\ R \\ N \\ D \\ C \\ Q \\ E \\ G \\ H \\ I \\ L \\ K \\ M \\ F \\ P \\ S \\ T \\ W \\ Y \\ V \\ B \\ J \\ Z \end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccccccccc} A & R & N & D & C & Q & E & G & H & I & L & K & M & F & P & S & T & W & Y & V & B & J & Z \end{array} \\
\begin{pmatrix}
5 & -2 & -1 & -2 & -1 & -1 & -1 & 0 & -2 & -1 & -2 & -1 & -1 & -3 & -1 & 1 & 0 & -3 & -2 & 0 & -2 & -2 & -1 \\
-2 & 7 & -1 & -2 & -4 & 1 & 0 & -3 & 0 & -4 & -3 & 3 & -2 & -3 & -3 & -1 & -1 & -3 & -1 & -3 & -1 & -3 & 0 \\
-1 & -1 & 7 & 2 & -2 & 0 & 0 & 0 & 1 & -3 & -4 & 0 & -2 & -4 & -2 & 1 & 0 & -4 & -2 & -3 & 5 & -4 & 0 \\
-2 & -2 & 2 & 8 & -4 & 0 & 2 & -1 & -1 & -4 & -4 & -1 & -4 & -5 & -1 & 0 & -1 & -5 & -3 & -4 & 6 & -4 & 1 \\
-1 & -4 & -2 & -4 & 13 & -3 & -3 & -3 & -3 & -2 & -2 & -3 & -2 & -2 & -4 & -1 & -1 & -5 & -3 & -1 & -3 & -2 & -3 \\
-1 & 1 & 0 & 0 & -3 & 7 & 2 & -2 & 1 & -3 & -2 & 2 & 0 & -4 & -1 & 0 & -1 & -1 & -1 & -3 & 0 & -3 & 4 \\
-1 & 0 & 0 & 2 & -3 & 2 & 6 & -3 & 0 & -4 & -3 & 1 & -2 & -3 & -1 & -1 & -1 & -3 & -2 & -3 & 1 & -3 & 5 \\
0 & -3 & 0 & -1 & -3 & -2 & -3 & 8 & -2 & -4 & -4 & -2 & -3 & -4 & -2 & 0 & -2 & -3 & -3 & -4 & -1 & -4 & -2 \\
-2 & 0 & 1 & -1 & -3 & 1 & 0 & -2 & 10 & -4 & -3 & 0 & -1 & -1 & -2 & -1 & -2 & -3 & 2 & -4 & 0 & -3 & 0 \\
-1 & -4 & -3 & -4 & -2 & -3 & -4 & -4 & -4 & 5 & 2 & -3 & 2 & 0 & -3 & -3 & -1 & -3 & -1 & 4 & -4 & 4 & -3 \\
-2 & -3 & -4 & -4 & -2 & -2 & -3 & -4 & -3 & 2 & 5 & -3 & 3 & 1 & -4 & -3 & -1 & -2 & -1 & 1 & -4 & 4 & -3 \\
-1 & 3 & 0 & -1 & -3 & 2 & 1 & -2 & 0 & -3 & -3 & 6 & -2 & -4 & -1 & 0 & -1 & -3 & -2 & -3 & 0 & -3 & 1 \\
-1 & -2 & -2 & -4 & -2 & 0 & -2 & -3 & -1 & 2 & 3 & -2 & 7 & 0 & -3 & -2 & -1 & -1 & 0 & 1 & -3 & 2 & -1 \\
-3 & -3 & -4 & -5 & -2 & -4 & -3 & -4 & -1 & 0 & 1 & -4 & 0 & 8 & -4 & -3 & -2 & 1 & 4 & -1 & -4 & 1 & -4 \\
-1 & -3 & -2 & -1 & -4 & -1 & -1 & -2 & -2 & -3 & -4 & -1 & -3 & -4 & 10 & -1 & -1 & -4 & -3 & -3 & -2 & -3 & -1 \\
1 & -1 & 1 & 0 & -1 & 0 & -1 & 0 & -1 & -3 & -3 & 0 & -2 & -3 & -1 & 5 & 2 & -4 & -2 & -2 & 0 & -3 & 0 \\
0 & -1 & 0 & -1 & -1 & -1 & -1 & -2 & -2 & -1 & -1 & -1 & -1 & -2 & -1 & 2 & 5 & -3 & -2 & 0 & 0 & -1 & -1 \\
-3 & -3 & -4 & -5 & -5 & -1 & -3 & -3 & -3 & -3 & -2 & -3 & -1 & 1 & -4 & -4 & -3 & 15 & 2 & -3 & -5 & -2 & -2 \\
-2 & -1 & -2 & -3 & -3 & -1 & -2 & -3 & 2 & -1 & -1 & -2 & 0 & 4 & -3 & -2 & -2 & 2 & 8 & -1 & -3 & -1 & -2 \\
0 & -3 & -3 & -4 & -1 & -3 & -3 & -4 & -4 & 4 & 1 & -3 & 1 & -1 & -3 & -2 & 0 & -3 & -1 & 5 & -3 & 2 & -3 \\
-2 & -1 & 5 & 6 & -3 & 0 & 1 & -1 & 0 & -4 & -4 & 0 & -3 & -4 & -2 & 0 & 0 & -5 & -3 & -3 & 6 & -4 & 1 \\
-2 & -3 & -4 & -4 & -2 & -3 & -3 & -4 & -3 & 4 & 4 & -3 & 2 & 1 & -3 & -3 & -1 & -2 & -1 & 2 & -4 & 4 & -3 \\
-1 & 0 & 0 & 1 & -3 & 4 & 5 & -2 & 0 & -3 & -3 & 1 & -1 & -4 & -1 & 0 & -1 & -2 & -2 & -3 & 1 & -3 & 5
\end{pmatrix}
\end{array}
\tag{2.7}
$$

$$
W = \begin{array}{c} \\ \mathbf{P} \\ \mathbf{A} \\ \mathbf{W} \\ \mathbf{H} \\ \mathbf{E} \end{array}
\begin{array}{c}
\begin{array}{ccccc} \mathbf{H} & \mathbf{E} & \mathbf{A} & \mathbf{G} & \mathbf{W} \end{array} \\
\begin{pmatrix}
2 & -1 & -1 & -2 & -4 \\
-2 & -1 & -5 & 0 & -3 \\
-3 & -3 & -3 & -3 & 15 \\
10 & 0 & -2 & -2 & -3 \\
0 & 6 & -1 & -3 & -3
\end{pmatrix}
\end{array}
\tag{2.8}
$$

A partir da Função de Pontuação e Matriz de Mérito é realizada o preenchimento do resto da matriz $m+1 \times n+1$ como mostra a Figura 2.7. Como exemplo, para a $F(2,4)$ teremos: Se $F(2-1,4-1)+W(A_2,B_4) = F(1,3)+W(P,G) = 0+0 = 0$, $F(2-1,4)-d = F(1,4)-8 = 0-8 = 0$, $F(2,4-1)-d = F(2,3)-8 = 5-8 = -3$, e $\max\{0,0,-8,-3\}$, logo

Figura 2.7: Smith-Waterman - Fase de preenchimento (*scoring*).

		0	1	2	3	4	5	6	7	8	9	10
			H	E	A	G	A	W	G	H	E	E
0		0	0	0	0	0	0	0	0	0	0	0
1	P	0	0	0	0	0	0	0	0	0	0	0
2	A	0	0	0	5	**0**						
3	W	0										
4	H	0										
5	E	0										
6	A	0										
7	E	0										

Figura 2.8: Smith-Waterman - Fim da fase de preenchimento (*scoring*).

		0	1	2	3	4	5	6	7	8	9	10
			H	E	A	G	A	W	G	H	E	E
0		0	0	0	0	0	0	0	0	0	0	0
1	P	0	0	0	0	0	0	0	0	0	0	0
2	A	0	0	0	5	0	5	0	0	0	0	0
3	W	0	0	0	0	2	0	20	12	4	0	0
4	H	0	10	2	0	0	0	12	18	22	14	6
5	E	0	2	16	8	0	0	4	10	18	**28**	20
6	A	0	0	8	21	13	5	0	4	10	20	27
7	E	0	0	6	13	18	12	4	0	4	16	26

$F(2,4) = 0$.

A matriz totalmente preenchida caracterizando o término da fase de *scoring* pode ser visualizada na Figura 2.8. Analisando a Figura 2.8 é possível identifica o valor máximo encontrado. Diferentemente do Algoritmo de Needleman-Wunsch onde o valor máximo estará na posição (m,n), na técnica de Smith-Waterman buscamos em toda a matriz o maior valor obtido. No caso o maior valor obtido foi 28.

A última etapa do algoritmo de Smith-Waterman é a etapa de *traceback*. O *traceback* permite encontrar o alinhamento que resulta na pontuação máxima alcançada. A etapa de *traceback* é iniciada a partir da posição de maior valor. A partir desta posição olhamos para as células vizinhas da posição atual. Dentre estes vizinhos esta o vizinho

Figura 2.9: Smith-Waterman - Fim da fase de alinhamento (*traceback*).

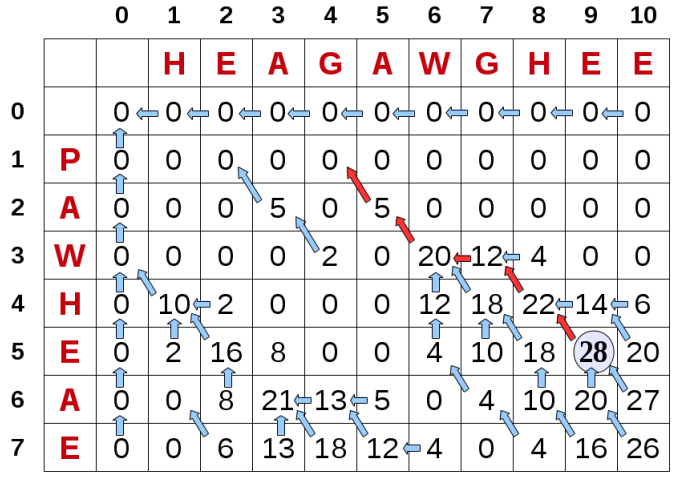


Figura 2.10: Smith-Waterman - Alinhamento Resultante.

A	A	W	-	H	E
B	A	W	G	H	E

precedente direto da posição atual. A posição corrente da varredura da matriz assume a posição do vizinho precedente e o processo se repete. Um caminho é percorrido até alcançar uma célula com valor 0 (zero). Neste processo de varredura deve-se levar em consideração as seguintes regras para o alinhamento:

- se o elemento precedente esta acima da posição atual então A_i alinha com lacuna;
- se o elemento precedente esta na diagonal da posição atual então B_j alinha com A_i;
- se o elemento precedente esta a esquerda da posição atual então B_j alinha com lacuna.

Na Figura 2.9 é possível identificar todo o caminho característico do alinhamento para a geração do alinhamento. O alinhamento resultado utilizando as informações da fase de *traceback* pode ser visualizado na Figura 2.10.

2.3 Alinhamento com Heurística entre duas sequências

A complexidade quadrática dos algoritmos de Needleman-Wunsch, Smith-Waterman e de suas variações fazem com que eles se tornem inviáveis em aplicações como busca em bases de dados moleculares (NCBI, 2015). Algumas destas bases possuem milhões de sequências (NCBI, 2015). O problema típico dessa aplicação é a comparação de uma dada sequência nova com todas as depositadas na base de dados. Isto significa que milhares de comparações precisam ser feitas (SETUBAL; MEIDANIS, 1997).

O primeiro programa de busca por similaridades em banco de dados largamente utiizado foi o FASTA (LIPMAN; PEARSON, 1985; BAXEVANIS; OUELLETTE, 2001). O método heurístico aplicado contruibui para a sua velocidade de execução. O princípio básico do programa FASTA é comparar cada sequência do banco de dados com a sequência procurada e retornar aquelas que tenham semelhança significativa (SETUBAL; MEIDANIS, 1997).

Para encontrar essa semelhança significativa primeiramente o algoritmo FASTA busca sub-sequências curtas (chamadas de k-tuplas), que ocorrem tanto na sequência de busca quanto na base de dados de sequências. Usando então a matriz BLOSUM50 (HENIKOFF; HENIKOFF, 1992), o algoritmo pontua os primeiros 10 alinhamentos sem lacunas que contenham as k-tuplas mais similares. Os alinhamentos sem lacunas encontrados são avaliados por sua capacidade de serem unidos a um alinhamento com lacunas sem reduzir a pontuação atual para abaixo de um determinado limiar.

Para aqueles alinhamentos unidos que possuem uma pontuação do limiar, um alinhamento local ótimo daquela região usando o algoritmo de Smith Waterman (SMITH; WATERMAN, 1981) é realizado e a pontuação para aquele alinhamento é retornada. Um dos problemas do FASTA é que ele possui alguns problemas relacionados à perda de regiões de sequências analisadas.

Uma outro caso de aplicação de heurística para resolver o problema de alinhamento entre duas sequências é o BLAST. O termo BLAST (ALTSCHUL et al., 1990) é uma acrônimo de *Basic Local Alignment Search Tool* (Ferramenta Básica de Busca de

Alinhamentos Locais). O BLAST é uma derivação do Algortimo de Smith-Waterman mais rápida do que o FASTA. O BLAST prioriza o alinhamento de locais específicos da sequência com relevância estatística, em lugar de realizar alinhamentos globais permitindo assim identificar relações de similiaridade entre sequências, mesmo que esta ocorra em somente algumas regiões isoladas (ALTSCHUL et al., 1990). A principal diferença do BLAST e o FASTA esta em como o BLAST trata a lista de *k*-tuplas identificadas.

Os algoritmos de Needleman-Wunsch, Smith-Waterman, FASTA e BLAST são comumente chamados de técnicas de alinhamento por pares (*pairwise*) por sempre envolver somente duas sequências de cada vez na realização do alinhamento. Mesmo sendo as técnicas FASTA e BLAST aplicadas para procura em um banco de dados de sequências, o processamento da comparação entre a sequência de consulta e as sequências que existem nos bancos de dados sempre serão efetuados aos pares.

2.4 Alinhamento Múltiplo de Sequências

O Alinhamento Múltiplo de Sequência (***Multiple Sequence Alignment***, MSA) é uma extensão do alinhamento por pares, permitindo que três ou mais sequências sejam alinhadas simultaneamente. Uma pequena similaridade entre pares de sequências alinhadas pode se tornar altamente significativa na presença de outras sequências. Os alinhamentos múltiplos podem revelar as semelhanças sutis que os alinhamentos por pares não são capazes de exibir.

De acordo com NOTREDAME (2002) a realização do MSA envolve três questões: Escolha das sequências, escolha de uma função objetivo capaz de qualificar biologicamente um alinhamento e a computação de um alinhamento ótimo.

A escolha das sequências é guiada pelo grau de similaridade existente entre elas. Sequências de organismos bastante distintos possuem baixa similaridade entre si, já sequência de organismos semelhantes possuem alta similiaridade. Alguns métodos de alinhamento múltiplo exigem a escolha de sequências de alta similaridade. O uso de sequências distantes em similaridade pode inserir ruído, incorporando um viés na execução do algoritmo, o que impossibilita o alinhamento de resíduos realmente relevantes

(NOTREDAME, 2002).

A função objetivo de MSA busca mensurar a qualidade biológica de um alinhamento. Uma função objetivo frequentemente utilizada é baseada no modelo de pontuação conhecido por *sum of pairs* (soma dos pares) (NEEDLEMAN; WUNSCH, 1970) . Este modelo de pontuação leva em consideração as correspondências entre bases de uma mesma coluna e a atribuição de penalidades para inserções e exclusões de bases. A computação da pontuação total, chamada de *score*, é obtida pelo cálculo de correspondências e pelas penalidades atribuídas às lacunas (*gaps*) é responsável por mensurar a qualidade do alinhamento.

Podemos agrupar as classes de algoritmos de MSA em exatos, progressivos e iterativos.

2.4.1 Algoritmos de MSA exatos

São o grupo de técnicas que empregam técnicas de programação dinâmica multidimensional. A abordagem faz uso do algoritmo de Needleman-Wunsch (NEEDLEMAN; WUNSCH, 1970) para reduzir a computação e estimar qual região de uma matriz diagonal multidimensional de busca deve ser explorada.

Embora sejam capazes de obter alinhamentos mais precisos e próximos do ótimo, as abordagens exatas possuem grandes custo computacional. Na manipulação de uma matriz multidimensional de busca na qual cada sequência representa uma dimensão extra, faz do MSA um problema de complexidade $O(LN)$ para N sequências de comprimento L tornando-o proibitivo mesmo para um pequeno número de sequências.

2.4.2 Algoritmos de MSA progressivos

Inicialmente descrito por Hogeweg e Hesper (HOGEWEG; HESPER, 1984) os algoritmos progressivos possuem uma boa relação custo/qualidade (NOTREDAME; HIGGINS; HERINGA, 2000; HIGGINS; SHARP, 1988; CORPET, 1988).

O processo se resume em três etapas. Na primeira etapa são realizados alinhamentos em pares (*pairwise*) entre todas as sequências para produzir scores

responsáveis por medir a similaridade entre as sequências (FENG; DOOLITTLE, 1987; HOGEWEG; HESPER, 1984; SIMOSSIS; KLEINJUNG; HERINGA, 2003; TAYLOR, 1988). Na segunda etapa, as sequências mais similares são agrupadas para formar um dendrograma que irá guiar o alinhamento na terceira etapa. A terceira etapa alinha as sequências e grupos de sequências da etapa anterior. Os grupos de sequências são convertidos em uma listagem exibindo a frequência das bases (ou aminoácidos) em cada coluna.

O Clustal (THOMPSON; HIGGINS; GIBSON, 1994; THOMSEN; FOGEL; KRINK, 2002; HIGGINS; BLEASBY; FUCHS, 1992) é a implementação de heurística progressiva mais conhecida e é capaz de alinhar poucas milhares de sequências de comprimento moderado (SIEVERS et al., 2011). Uma das deficiências do Clustal está em não reavaliar lacunas inseridas inicialmente, este possível erro cometido no início do processo de alinhamento é mantidos até o final do processo de construção e não pode ser reparado. Este tipo de problema é conhecido na literatura como *once a gap always a gap* (NOTREDAME; HIGGINS; HERINGA, 2000).

Para combater o problema do Clustal foram desenvolvidos algoritmos progressivos utilizando novas heurísticas. Exemplos destes algoritmos progressivos são: O T-Coffee (NOTREDAME; HIGGINS; HERINGA, 2000) e o PRALINE (SIMOSSIS; HERINGA, 2005). A melhoria da precisão obtida por estas técnicas também veêm associada ao elevado custo computacional, limitando estes novos algoritmos a operarem com uma quantidade de sequências reduzida (CUTELLO et al., 2011).

2.4.3 Algoritmos de MSA iterativos

Algoritmos iterativos de MSA produzem inicialmente alinhamentos não ótimos e através de um processo de várias iterações vão melhorando o alinhamento inicial até que um limiar pré-definido de qualidade seja alcançado. Nos algoritmos de MSA iterativos estocásticos um grau de aleatoriedade é incorporado a solução. Um exemplo de técnicas de MSA iterativas estocásticas são aquelas que modelam Algoritmos Evolucionários, como os Algoritmos Genéticos ou Evolução Diferencial.

3

ALGORITMOS EVOLUCIONÁRIOS

A utilização de AEs na solução de problemas de MSA tem-se apresentado com relativa frequência no estado da arte: Zhang e Wong (ZHANG; WONG, 1997), em 1997, apresentaram uma solução baseada no alinhamento exato de colunas, alcançando bons resultados. Contudo, a solução limita-se ao tratamento de sequências com alto grau de similaridade. Thomsen, Fogel e Krink (THOMSEN; FOGEL; KRINK, 2002), em 2002, apresentaram uma solução que utilizava como entrada alinhamentos já executados pelo Clustal V (HIGGINS; BLEASBY; FUCHS, 1992), com o objetivo de melhorar resultados já alcançados. Zhang e Huang (ZHANG; HUANG, 2004), em 2004, propuseram um GA tendo uma estratégia de inicialização da população e um esquema de operador de cruzamento baseados em um ponto. Meshoul et al. (MESHOUL; LAYEB; BATOUCHE, 2005) e Abdesselem et al. (LAYEB; MESHOUL; BATOUCHE, 2006) propuseram, respectivamente em 2005 e 2006, algoritmos que mesclam conceitos de computação quântica e algoritmos evolucionários com o objetivo de obter melhores resultados. Ainda em 2006, Hu (HU, 2009) apresentou uma solução utilizando uma forma híbrida de DE (*Chaos-differential evolution*, CDE), a fim de evitar a convergência prematura do processo de busca. Gondro e Kinghorn (GONDRO; KINGHORN, 2007), em 2007, propuseram um GA tendo dois operadores para crossover, um para combinações horizontais e outro para combinações verticais, e quatro operadores de mutação, todos operando sobre lacunas do alinhamento, garantindo resultados superiores ao do ClustalW (THOMPSON;

HIGGINS; GIBSON, 1994). A desvantagem deste esquema é o número de gerações: cerca de 20000. Lee et al. (LEE et al., 2008), em 2008, propuseram o algoritmo GA-ACO, um GA que fazia uso do algoritmo de colônia de formigas (DORIGO; DI CARO; GAMBARDELLA, 1999) para otimizar o espaço de busca. Botta e Negro (MASULLI; PETERSON; TAGLIAFERRI, 2010), em 2010, apresentaram uma solução onde uma matriz com pesos posicionais (*Positional Wight Matrix*, PWM) representava um indivíduo da população, representando a probabilidade de uma determinada posição da sequência ser associada a uma coluna do alinhamento.

3.1 Algoritmos Evolucionários

3.1.1 Conceito

Os AEs são uma classe de técnicas de otimização estocástica inspirados nos mecanismos de evolução natural das populações de espécies aplicados ao contexto da otimização. Na metáfora evolucionária temos uma população de indivíduos existentes em um ambiente com recursos limitados. A competição por recursos leva a uma seleção natural que irá destacar os indivíduos mais aptos (mais adaptados ao ambiente). Estes indivíduos mais aptos irão possuir preferência em relação aos demais no momento da geração de novos indivíduos para a próxima geração. Este processo de geração de novos indivíduos ocorre a partir de operações de recombinação e mutação. Os novos indivíduos iram competir entre si ou também com seus parentes para sobreviverem. Ao longo das gerações, o processo de seleção natural permitirá um aumento da aptidão geral dos indivíduos da população.

Duas forças fundamentais foram a base do sistema evolucionário: os operadores de variação (recombinação e mutação) que criam a diversidade necessária para o processo de busca e o operador de seleção que permite a redução da diversidade populacional, atuando como uma força que melhora a qualidade da população. O equilíbrio entre estes dois elementos tende a melhorar o valor de aptidão nas populações seguintes.

Algoritmo 1: Algoritmo Evolucionário típico.

Algoritmo *Algoritmos Evolucionário Típico*
1 INICIALIZA população com soluções candidatas aleatórias;
2 AVALIA cada candidato;
3 **enquanto** critério de parada não for satisfeito **faça**
4 SELECIONA parentes;
5 RECOMBINA pares de parentes;
6 MUTAÇÃO nos descendentes resultantes da recombinação;
7 AVALIA novos candidatos;
8 SELECIONA indivíduos para a próxima geração;
9 **fim enquanto**
fim

3.1.2 Definição algorítmica

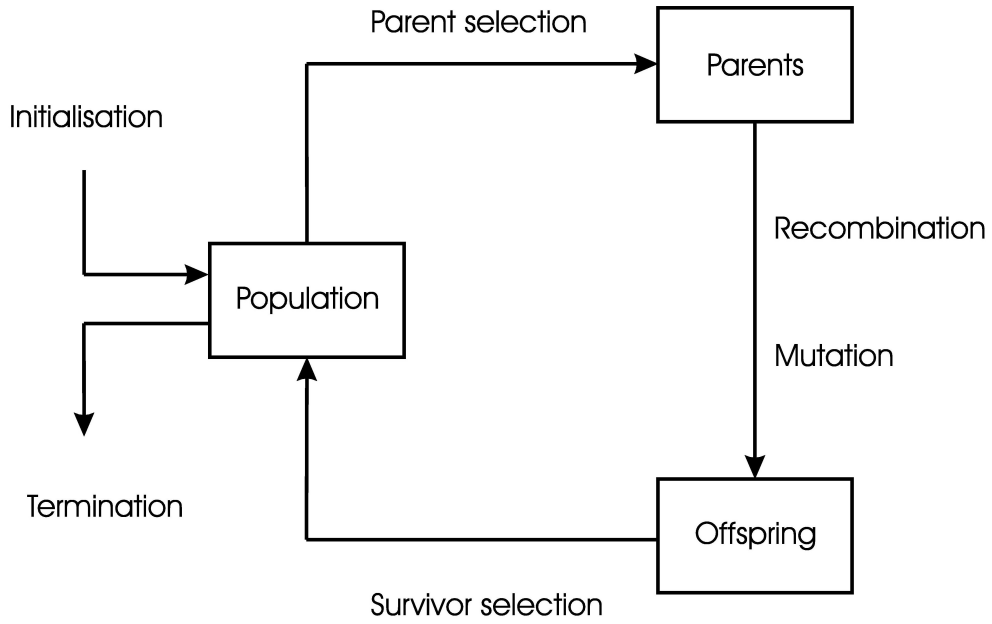
Em Algoritmo 1 temos uma representação algorítmica do esquema geral de um AE aplicado a um problema de otimização. A primeira atividade no AE é a inicialização aleatória (Linha 1) de uma população de indivíduos representantes de soluções candidatas ao problema de otimização. Após esta etapa temos a avaliação da aptidão destes indivíduos (Linha 2).

Em seguida, entramos em um laço de repetição (das Linhas 3-9) executado até que a condição de parada tenha sido satisfeita. A condição de parada pode ser um limiar de aptidão mínimo a ser atingido ou um número máximo de gerações. A primeira atividade dentro do laço é a execução de alguma forma de seleção de indivíduos (Linha 4) que serão os parentes dos indivíduos da próxima geração. Para geração destes novos indivíduos, são aplicados os operadores de variação de Recombinação (Linha 5) e Mutação (Linha 6). Após a aplicação dos operadores de variação, um novo cálculo (Linha 7) de aptidão é realizado semelhante ao executado na Linha 2, a diferença é que agora o cálculo é aplicado aos novos indivíduos gerados. Na última atividade (Linha 8) temos a aplicação de um mecanismo de seleção de indivíduos que irá selecionar aqueles que irão fazer parte da próxima geração. Na Figura 3.1 temos o esquema geral ilustrativo dos AEs onde é possível visualizar as mesmas atividades elucidadas no pseudocódigo mas sem as atividades de avaliação de aptidão.

Existem vários formas AEs, chamados de dialetos. Os dialetos aplicados a este

trabalho são os Algoritmos Genéticos (*Genetic Algorithmns*, GA) (GOLDBERG, 1989) e Evolução Diferencial (*Differential Evolution*, DE) (STORN; PRICE, 1997; PRICE; STORN; LAMPINEN, 2005) abordados em ordem a partir das próximas sessões.

Figura 3.1: Fluxo de um Algoritmo Evolucionário típico.



Fonte: Adaptado de (EIBEN; SMITH, 2003).

3.1.3 Representação

A representação é um dos primeiros aspectos a serem considerados no uso de AEs. A representação do indivíduo é chamada de Cromossomo. O Cromossomo é uma estrutura de dados, geralmente vetores ou cadeias de valores binários que codificam uma possível solução para o problema de otimização. A estrutura da representação de um indivíduo depende do problema abordado. Cada método de busca também possui características que ajudam na especificação do tipo de representação a ser utilizada.

O termo Fenótipo é utilizado para designar as características expressas por um indivíduo levando em consideração o seu Cromossomo, no caso, uma solução possível no espaço de busca. O termo Genótipo é utilizado para designar uma representação do Fenótipo na forma de valores nos Cromossomos.

3.1.4 Inicialização

O processo de inicialização define como a população inicial de indivíduos será inicialmente no espaço de busca. É um passo computacional que depende da representação adotada no modelo do problema. A forma mais comum de inicialização é a aleatória. Neste caso os indivíduos são distribuídos aleatoriamente no espaço solução. O problema desta estratégia é ocorrência de regiões não inicialmente exploradas no espaço de busca e outras regiões com maiores concentração de indivíduos. Uma outra forma de inicialização é obedecer uma distribuição uniforme dos indivíduos no espaço de busca, permitindo a vantagem de cobrir bem o espaço solução, contudo execuções sucessivas não trazem informação adicional já que o estado de inicialização será sempre o mesmo. É possível também "semear" a população inicial com indivíduos em posições no espaço de busca já conhecidos, obtidos pela aplicação de outros métodos de busca ou até de rodadas anteriores do algoritmo evolucionário.

3.1.5 Função de aptidão

A Função de Aptidão é uma expressão matemática utilizada para quantificar a proximidade das soluções à solução desejada. Alguns problemas de otimização procuram maximizar o valor da função, isto é, encontrar soluções que produzam o maior valor possível para a função de aptidão, como por exemplo: Definir o número máximo de caixas que podem ser colocadas dentro de um depósito. Outros problemas procuram minimizar o valor da função de aptidão, como por exemplo: Encontrar um caminho mais curto entre um conjunto de rotas. Existem ainda funções que procuram satisfazer mais de um objetivo, encontradas em problemas de otimização multiobjetivo.

Na Tabela 3.1 temos um exemplo do cálculo da aptidão de 7 indivíduos de uma população exemplo. Os cromossomos são representados por uma cadeia de valores binários. A função de aptidão escolhida somente retorna o valor correspondente em decimal do valor da cadeia binária. Na última coluna temos a aptidão relativa do invíduo utilizada para calcular o quanto apto é o indivíduo em relação a aptidão total da população.

Tabela 3.1: Valor de aptidão de indivíduos de uma população.

Indivíduo (i)	Cromossomo	Aptidão (f_i)
1	0000000	0
2	0000001	1
3	0000010	2
4	0001010	10
5	0001011	11
6	0000101	5
7	1010000	80

3.1.6 Seleção de pais

O objetivo do operador de seleção de pais é permitir de forma equilibrada que os indivíduos da população atual sejam escolhidos para participar na definição dos indivíduos da próxima geração. Como a seleção de pais é comumente probabilística, os indivíduos mais aptos possuem mais chances de serem escolhidos, o equilíbrio na escolha destes pais é necessário para que os indivíduos menos aptos também possam fazer parte do processo de geração de novos indivíduos. Esta necessidade se justifica pela manutenção da diversidade populacional, evitando uma convergência prematura do processo de busca (DORTMUND, 1995).

3.1.7 Operadores de variação

Os operadores de variação têm como papel gerar novos indivíduos a partir dos já existentes. Os operadores de variação são geralmente dividido em dois tipos de acordo com a aridade (número de entradas). Operadores de variação que recebem uma entrada são chamados de operadores de mutação. Operadores de variação que recebem mais de uma entrada são chamados de operadores de recombinação. Os operadores de recombinação que recebem duas entradas são tipicamente chamados de operadores de cruzamento (*Crossover*).

3.1.7.1 Cruzamento

Este operador simula o processo de reprodução sexuada como ocorre na natureza, em que os filhos são gerados a partir da combinação dos genes dos pais. As várias formas de se implementar o operador cruzamento exigem (em geral) como entrada dois indivíduos pais e uma constante P_{cruz}, definida no intervalo $[0,1]$, correspondente a probabilidade de ocorrência do cruzamento. A implementação desde operador depende do dialeto de AEs assim como a representação adotada dos indivíduos.

3.1.7.2 Mutação

Após a aplicação do operador de cruzamento, o processo segue com a aplicação do operador de mutação nos indivíduos filhos gerados no cruzamento. Na mutação é realizada uma pertubação dos valores que compõem a representação do indivíduo, afim de inserir variabilidade genética na população. A importância que é dada ao operador de mutação depende do dialeto de AEs, por exemplo, no caso dos GAs, o operador de mutação tem um papel secundário sendo responsável pela preservação e introdução de diversidade; já na Programação Evolucionária (*Evolutionary Programming*, EP) o operador de mutação é o único operador de variação utilizado; na Programação Genética (*Genetic Programming*, GP) a mutação é raramente utilizada. A implementação do operador de mutação depende também da representação adotada do indivíduo. Uma constante P_{mut}, definida no intervalo $[0,1]$, chamada de Taxa de Mutação determina a probabilidade de cada indivíduo sofrer mutação.

3.1.8 Seleção de sobreviventes

A fase de seleção de sobreviventes consiste na seleção dos indivíduos (entre os pais e filhos) que irão fazer parte da próxima geração. Muitos dialetos de AEs consideram que o tamanho da população seja constante, então é necessário uma estratégia para escolher indivíduos entre pais e filhos. Tipicamente temos a estratégia da substituição generalizada, onde todos os indivíduos filhos substituem completamente a população dos pais. Esta é estratégia adotada do dialeto GA canônico. Outra estratégia é a troca em estado

estacionário, onde em cada geração um filho é gerado, podendo substituir o pior indivíduo da população dos pais. Uma outra forma chamada de Elitismo consiste em selecionar os melhores indivíduos entre os pais e filhos.

3.2 Algoritmos Genéticos

Os Algoritmos Genéticos (*Genetic Algorithmns*, GA) (HOLLAND, 1975) constituem uma das mais conhecidas classes de Algoritmos Evolucionários. Os GA realizam um processo de busca inspirado no processo de seleção natural da Teoria da Evolução de Darwin e na genética de Mendel.

O GA opera através dos mesmos passos computacionais empregados pelos Algorítimos Evolucionários clássicos como a Inicialização, Seleção de pais, Cruzamento, Mutação e Seleção de Sobreviventes.

A formulação inicial do GA considera uma representação binária do indivíduo, uma seleção de pais proporcional à aptidão (Método da Roleta), valores baixos de P_{mut} (Taxa de Mutação), cruzamento de um ponto e na seleção de sobreviventes a prole sempre substituía os seus pais na próxima geração. O modelo inicial do GA tinha como objetivo definir um método de estudo ao comportamento adaptativo (HOLLAND, 1975) porém tem sido largamente aplicado como método de otimização, aprendizado de máquina e outras classes de problemas (EIBEN; SMITH, 2003; MITCHELL, 1998; WHITLEY, 1994).

3.2.1 Definição algorítmica

De acordo com o Algoritmo 2, o GA típico manipula a cada iteração G, um conjunto de pontos (população) de tamanho constante, gerando a partir destes uma nova população. A primeira atividade consiste na inicialização (Linha 2) da população $P_{indivíduos}(G)$ da geração G, sendo inicialmente $G = 1$. Em seguida a função de aptidão avalia (Linha 3) os indivíduos gerados na etapa anterior. O processo segue entrando em um laço evolucionário (Linha 4) que se repete até que a uma condição de parada definida tenha sido alcançada. Dentro do laço evolucionário uma população de pares de pais $P_{pais}(G)$ é

Algoritmo 2: Algoritmo Genético típico.

Algoritmo *Algoritmo Genético Típico*
Entrada
NP: tamanho da população
P_{cr}: taxa de cruzamento
P_m: taxa de mutação
Saída
população de indivíduos-soluções

1 $G \leftarrow 1$;
2 $inicializa(P_{indivíduos}(G), NP)$;
3 $avalia(P_{indivíduos}(G))$;
4 **enquanto** critério de parada não for satisfeito **faça**
5 $P_{pais}(G) \leftarrow P_{indivíduos}(G).SelecionaPais()$;
6 $P_{filhos}(G) \leftarrow cruzamento(P_{pais}(G), P_{cr})$;
7 $mutação(P_{filhos}(G), P_m)$;
8 $avalia(P_{filhos}(G))$;
9 $P_{indivíduos}(G+1) \leftarrow selecionaSobreviventesEntre(P_{filhos}(G), P_{indivíduos}(G))$;
10 $G \leftarrow G+1$;
11 **fim enquanto**
fim

gerada a partir da população atual $P_{indivíduos}(G)$ (Linha 5). O processo segue com a aplicação dos operadores de variação, onde na Linha 6 uma população de filhos $P_{filhos}(G)$ é gerada a partir do cruzamento entre os pares da população de pares de pais $P_{pais}(G)$ respeitando uma taxa de probabilidade de cruzamento P_{cr}. A aplicação dos operadores de variação termina com a Linha 7, onde o operador de mutação é aplicado sobre a população de filhos $P_{filhos}(G)$ respeitando uma probabilidade de mutação P_m. Agora é necessário realizar a avaliação da aptidão destes novos indivíduos que foram gerados, isto ocorre na Linha 8 e na Linha 9 são selecionados os sobreviventes para população da próxima geração $P_{indivíduos}(G+1)$ entre os indivíduos já existentes $P_{indivíduos}(G)$ e os indivíduos filhos $P_{filhos}(G)$. Cada etapa computacional de um GA será a seguir tratada em detalhes.

3.2.2 Representação

A representação a ser utilizada é inerente à modelagem do problema a ser explorado utilizando GA. O GA permite diversas formas de representação. O modelo inicial de GA utilizava representação binária (HOLLAND, 1975), porém é possível utilizar outras formas

lineares de dados como as representações numéricas (inteiros e reais), vetores de strings e até estruturas não lineares como árvores e grafos.

3.2.3 Inicialização

As considerações quanto à etapa de inicialização dos AE tratados na sessão 3.1.4 aplicam-se também ao passo computacional de inicialização implementado pelos GA.

3.2.4 Função de aptidão

As considerações quanto à etapa de Função de Avaliação dos AE tratados na sessão 3.1.5 aplicam-se também ao passo computacional correspondente no GA.

3.2.5 Seleção de pais

Existem vários métodos de seleção propostos. A maioria das propostas busca favorecer indivíduos mais aptos, como o Método da Roleta. No método da roleta os indivíduos são representados em uma roleta por uma fatia proporcional à sua aptidão. Assim, indivíduos com maiores valores de aptidão ocuparão maiores fatias na roleta, enquanto indivíduos com menores valores de aptidão ocuparão fatias menores. A área da fatia da roleta de cada indivíduo é definida através de sua aptidão relativa, segundo a Equação:

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j} \tag{3.1}$$

onde:

- p_i é a probabilidade de um indivíduo i ser selecionado.
- f_i corresponde à aptidão do indivíduo i, sendo $f_i \geq 0$, e
- m é o número de indivíduos da população.

Durante a fase de seleção de pais, a roleta é "girada", selecionando um indivíduo que irá seguir para a próxima fase, onde são aplicados os operadores de variação. A

Figura 3.2: Roleta.

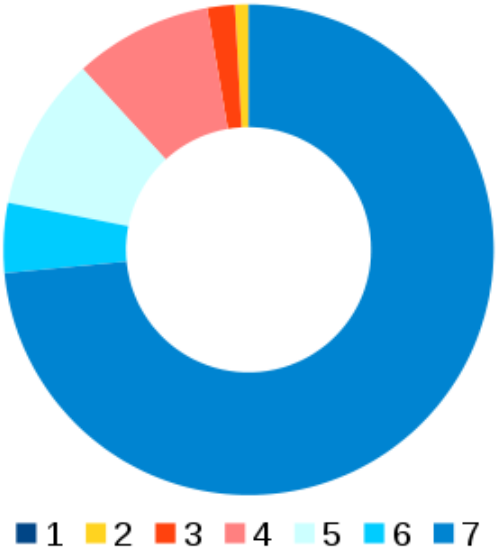


Figura 3.2 ilustra a definição de uma roleta para uma população exemplo disposta na Tabela 3.2. O método da roleta não funciona em caso de aptidões negativas e pode levar a uma convergência prematura do processo de busca.

Tabela 3.2: Probabilidade de seleção na roleta.

Indivíduo (i)	Cromossomo	Aptidão (f_i)	Aptidão Relativa (p_i)	Tamanho da Fatia
1	0000000	0	0.000	0%
2	0000001	1	0.009	9%
3	0000010	2	0.018	1.8%
4	0001010	10	0.091	9.1%
5	0001011	11	0.101	10.1%
6	0000101	5	0.046	4.6%
7	1010000	80	0.734	73.4%

Outra estratégia de seleção de pais é o método por torneio, onde um número k de indivíduos da população são escolhidos aleatoriamente, com a mesma probabilidade. O indivíduo com maior aptidão dentre estes k indivíduos será selecionado.

3.2.6 Cruzamento

A aplicação dos Operadores de Seleção de Pais retornam os dois indivíduos "pais" necessários para a realização do cruzamento. Em seguida é realizado um sorteio

Figura 3.3: Cruzamento com um ponto.

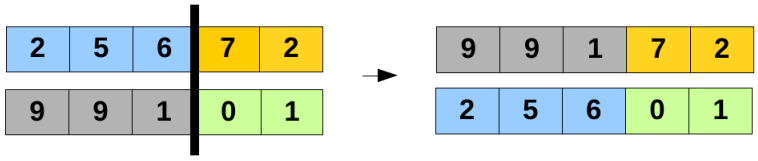
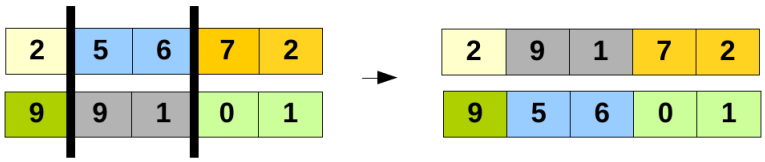


Figura 3.4: Cruzamento com vários pontos.



considerando a constante P_{cruz} já apresentada que irá definir se o cruzamento irá ocorrer ou não. Deste modo, de acordo com a taxa de cruzamento, pode ocorrer que os cromossomos "pais" sejam repassados sem modificação para a geração seguinte, criando "filhos" idênticos a eles.

Algumas formas de cruzamento típicas são: Cruzamento com um Ponto, Cruzamento com Vários Pontos e o Cruzamento Uniforme.

No Cruzamento com um Ponto o corte para a realização da operação é escolhido aleatoriamente, o exemplo da Figura 3.3 ilustra o processo. Quando são realizados cortes em mais de um ponto, caracteriza-se então o Cruzamento com Vários Pontos (*multi-point crossover*) (GOLDBERG, 1989) exemplificado na Figura 3.4

3.2.7 Mutação

Na Figura 3.5 temos um exemplo de mutação por inversão para uma representação binária usando um GA como dialeto. No exemplo é escolhido uma posição aleatória na representação do indivíduo e uma operação de inversão binária é realizada no valor da posição.

Uma outra forma de mutação é a troca aleatória, onde é escolhido uma posição

Figura 3.5: Mutação por inversão.

Figura 3.6: Mutação por troca aleatória.

aleatória na representação do indivíduo e uma operação de troca do valor da posição por um novo valor aleatório é realizada. Um exemplo dessa operação é visto na Figura 3.6.

3.2.8 Seleção de sobreviventes

As considerações quanto a etapa de Seleção de Sobreviventes dos AE tratados na sessão 3.1.8 aplicam-se também ao passo computacional correspondente no GA.

3.3 Evolução Diferencial

A Evolução Diferencial (*Differential Evolution*, DE) foi apresentada inicialmente por Storn e Price (STORN; PRICE, 1997) como uma heurística eficiente para otimização em espaços contínuos.

A DE opera através dos mesmos passos computacionais empregados pelos algoritimos evolucionários clássicos. Contudo, diferentemente dos demais algoritmos evolucionários como o GA, a DE emprega diferenças ponderadas entre vetores de parâmetros que compõem a população na realização do processo de busca. Algoritimos ancestrais a DE como o Algoritimo de Nelder-Mead (NELDER; MEAD, 1965) e Busca Randômica Controlada (*Controlled Random Search*, CRS) (PRICE, 1977), também realizavam diferença entre os vetores de parâmetros como forma de implementar variabilidade.

A DE é visto como um método de otimização atrativo pela sua facilidade de implementação comparada a maiorias dos outros AEs. A pesar de algoritmos como o PSO (*Particle swarm optimization*) também serem fáceis de implementar, o desempenho de DE em relação a PSO é superior em uma grande variedade de problemas como mencionam os estudos (DAS et al., 2009), (RAHNAMAYAN; TIZHOOSH; SALAMA, 2008) e (VESTERSTROM; THOMSEN, 2004). Como indicado em estudos sobre

DE (DAS et al., 2009), (RAHNAMAYAN; TIZHOOSH; SALAMA, 2008) e (ZHANG; SANDERSON, 2009), apesar de sua simplicidade a DE mostra uma performance superior a outros algoritmos como G3 com PCX, MA-S2, ALEP, CPSO-H. Outra característica marcante da DE é o pequeno número de parâmetros de controle (CR, F, e NP) na sua forma clássica que irão ser descritos em seguida. Esta característica permite que o algoritmo possa trabalhar em larga escala e em problemas de otimização custosos.

3.3.1 Definição algorítmica

A DE típica manipula a cada iteração G uma coleção de vetores de parâmetros D-dimensionais $x_{i,G}$ definidos na Equação 3.2. De acordo com o Algoritmo 3, o algoritmo recebe como entrada uma constante NP correspondente ao tamanho da coleção de indivíduos, uma taxa de cruzamento CR e o fator de ponderação F das diferenças vetoriais.

$$x_{i,G} = \left(x_{1i,G}, x_{2i,G}, ..., x_{Di,G}\right), \text{ sendo, } i = 1, ..., NP. \tag{3.2}$$

A primeira atividade (Linha 1) consiste na inicialização onde os vetores soluções $x_{i,1}$ são gerado aleatoriamente e devem cobrir todo o espaço de busca. Na ausência de qualquer conhecimento acerca do espaço de busca (regiões promissoras ou mesmo soluções parciais), utiliza-se uma distribuição uniforme para a população inicial. Na (Linha 2) a função de aptidão avalia os indivíduos gerados na etapa anterior. O processo segue entrando no laço evolucionário (Linha 3) que se repete até que a uma condição de parada definida tenha sido alcançada. Dentro do laço evolucionário, a primeira atividade executada é a aplicação dos operadores de variação (Linhas 5 e 6) para os vetor de parâmetros $x_{i,G}$. O operador de mutação (Linha 5) tem como entrada o vetor de parâmetros $x_{i,G}$ atual e o fator de ponderação F e retorna um vetor resultado $v_{i,G}$ chamado de vetor mutante. O operador de cruzamento (Linha 6) tem como entrada o vetor de parâmetros $x_{i,G}$ atual, o vetor mutante $v_{i,G+1}$ atual, a taxa de cruzamento CR e retonar um vetor resultado $u_{i,G+1}$ chamado de vetor ensaio. Os vetores mutantes e ensaio são definidos na Equação 3.3 e 3.4 respectivamentes.

Algoritmo 3: Algoritmo de Evolução Diferencial típico.

Algoritmo *Algortimo de Evolução Diferencial Típico*
Entrada
NP: tamanho da população
CR: taxa de cruzamento
F: fator de ponderação das diferenças vetoriais
Saída
população de indivíduos-soluções

```
1  G ← 1
2  inicializa(x_{i,1}, NP)
3  fit_x ← avalia(x_{i,1})
4  enquanto critério de parada não for satisfeito faça
5      para i = 1 ate NP faça
6          v_{i,G+1} ← mutation(x_{i,G}, F)
7          u_{i,G+1} ← crossover(x_{i,G}, v_{i,G+1}, CR)
8      fim para
9      fit_u ← avalia(u_{i,G+1})
10     para i = 1 ate NP faça
11         se fit_u(i) > fit_x(i) então
12             x_{i,G+1} ← u_{i,G+1}
13         senão
14             x_{i,G+1} ← x_{i,G}
15         fim se
16     fim para
17     G ← G + 1
18 fim enquanto
fim
```

$$v_{i,G+1} = \left(v_{1i,G+1}, v_{2i,G+1}, ..., v_{Di,G+1}\right), \text{ sendo, } i = 1, ..., NP, \tag{3.3}$$

$$u_{i,G+1} = \left(u_{1i,G+1}, u_{2i,G+1}, ..., u_{Di,G+1}\right), \text{ sendo, } i = 1, ..., NP, \tag{3.4}$$

Ao término da aplicação dos operadores de variação os vetores ensaios são avaliados pela função de aptidão (Linha 9). O processo segue com a aplicação do operador de seleção de sobreviventes, onde a aptidão do vetor solução $x_{i,G}$ é comparada com a aptidão do seu vetor ensaio $u_{i,G+1}$ correspondente, seguindo para a próxima geração os vetores mais aptos dentre os dois. Por fim o contador de iterações é incrementado (Linha 17) é o processo se repete.

3.3.2 Representação

Devido a natureza dos passos computacionais da DE, este dialeto de AE somente permite representações numéricas (inteiros e reais), já que as operações de mutação necessitam operar sobre parametros pertencentes ao conjunto dos números reais.

3.3.3 Inicialização

As considerações quanto a etapa de inicialização dos AE tratados na sessão 3.1.4 aplicam-se também ao passo computacional de inicialização implementado na DE.

3.3.4 Função de aptidão

As considerações quanto a etapa de Função de Avaliação dos AE tratados na sessão 3.1.5 aplicam-se também ao passo computacional correspondente na DE.

3.3.5 Mutação

Ao longo das iterações do processo evolucionário, a DE gera novos vetores de parâmetros $v_{i,G+1}$ chamados de vetores mutantes. Esta geração ocorre através da adição a um vetor solução da população atual a diferença ponderada entre outros vetores soluções. Esta operação é chamada de mutação. Existem várias abordagens para a produção do vetor mutante mas é sempre presente a diferença ponderada entre os vetores. Para cada vetor solução $x_{i,G}$ da população G, sendo $i = 1,2,3,4,5...,N_{pop}$, um vetor mutante pode ser gerados de acordo com as regras Rand/1/Bin (Equação 3.5), Rand/2/Bin (Equação 3.6), Best/1/Bin (Equação 3.7), Best/2/Bin (Equação 3.8) e Rand-To-Best/1/Bin (Equação 3.9). Os índices aleatórios $r_1, r_2, r_3, r_4 \in \{1,2,...,N_{pop}\}$, inteiros, mutualmente diferentes e F é um fator constante $\in [0,2]$ que controla a amplificação da diferença. O vetor $x_{best,G}$ corresponde ao melhor indivíduo da população na geração G.

$$v_{i,G+1} = x_{r_1,G} + F \cdot \left(x_{r_2,G} - x_{r_3,G}\right) \tag{3.5}$$

$$v_{i,G+1} = x_{r_1,G} + F \cdot \left(x_{r_2,G} - x_{r_3,G}\right) + F \cdot \left(x_{r_4,G} - x_{r_3,G}\right) \tag{3.6}$$

$$v_{i,G+1} = x_{best,G} + F \cdot \left(x_{r_1,G} - x_{r_2,G}\right) \tag{3.7}$$

$$v_{i,G+1} = x_{best,G} + F \cdot \left(x_{r_1,G} - x_{r_2,G}\right) + F \cdot \left(x_{r_3,G} - x_{r_4,G}\right) \tag{3.8}$$

$$v_{i,G+1} = x_{i,G} + F \cdot \left(x_{best,G} - x_{i,G}\right) + F \cdot \left(x_{r_1,G} - x_{r_2,G}\right) \tag{3.9}$$

3.3.6 Cruzamento

Para cada vetor solução $x_{i,G}$ da população G, sendo $i = 1,2,3,4,5...,N_{pop}$, além do vetor mutante $v_{i,G}$ é também gerado o vetor ensaio $u_{i,G}$, formado pela regra:

$$u_{ij,G+1} = \begin{cases} v_{j,i,G+1} & \text{se } rand_j(0,1) \leq \text{CR}, \\ x_{j,i,G} & \text{caso contrário.} \end{cases} \tag{3.10}$$

Na Equação 3.10, $CR \in [0,1]$ é uma constante predefinida e $rand_j(0,1)$ corresponde j-éssima geração de um número aleatório com distribuição normal $\in [0,1]$. Em Algoritmo 4 temos um pseudo-código padrão do processo de cruzamento onde a formação do vetor ensaio pode ser visto de forma mais detalhada.

Algoritmo 4: Algoritmo padrão de cruzamento para DE.

Algoritmo *cruzamento*
Entrada
$x_{i,G}$: vetor alvo i da geração G
$v_{i,G+1}$: vetor mutante i para a geração $G+1$
CR: taxa de cruzamento
Saída
$u_{i,G+1}$: vetor alvo

```
1 para j = 1 ate D faça
2     se randb(j) <= CR então
3         u_{j,i,G+1} ← u_{j,i,G+1}
4     senão
5         u_{j,i,G+1} ← x_{j,i,G}
6     fim se
7 fim para
```

fim

3.3.7 Seleção de sobreviventes

Os vetores que irão fazer parte da geração $G+1$ são definidos no processo de seleção de sobreviventes. O vetor solução $x_{i,G+1}$ e o seu vetor ensaio $u_{i,G+1}$ correspondente são dados como entrada neste processo, o vetor entre os dois que possuir melhor aptidão segue para compor a próxima geração.

4

METODOLOGIA

Este capítulo tem como objetivo descrever os materiais e métodos utilizados para atingir os objetivos do trabalhos.

4.1 Sequências utilizadas

Para formar a base de dados utilizada nos experimentos, foram construídos no total 50 grupos de 4 sequências retiradas do genoma do fungo Pyrenophora tritici-repentis da base pública do EMBL-EB. Foram utilizadas as sequências AAXI01000001 até AAXI01000040.

Para cada sequências AAXI010000* foram capturadas as primeiras 960 bases e em seguida distribuídas em linhas de comprimento igual a 60 bases. Na construção de cada um dos 50 conjuntos de sequências, quatro desses genomas foram escolhidos uniformemente ao acaso e de cada um, foi extraída a n-ésima linha onde n corresponde a um número aleatório $\in [1, 16]$ com distribuição uniforme. Estas quatro linhas irão compor grupos de quatro sequências que serão exclusivas para um ensaio.

4.2 Experimentos com GA

Nos experimentos com GA foi utilizado uma distribuição de probabilidade uniforme para a inicialização da população. A população inicial N_{pop} foi de 1500 indivíduos. A estratégia de seleção de país utilizada foi o torneio entre 2 indivíduos. A probabilidade de cruzamento P_{cruz} utilizada foi de 0.9. A probabilidade de mutação P_{mut} foi de 0.9. A técnica de cruzamento utilizada foi de um ponto. A técnica de mutação utilizada foi a de troca aleatória. Como estratégia de seleção de sobrevivente foi usado elitismo. Para a condição de parada do processo evolucionário foi definido um número máximo de 1500 iterações ou 150 iterações sem melhora na pontuação.

4.3 Experimentos com DE

Para as cinco abordagens de DE definidas (Best/1/Bin, Best/2/Bin, Rand/1/Bin, Rand/2/Bin e Rand-To-Best/1/Bin) foram utilizados os mesmos 50 grupos de sequências. Para cada grupo de sequência o processo evolucionário foi executado 100 vezes. Os parâmetros utilizados na execução de cada estratégia de DE estão na Tabela 4.1.

Tabela 4.1: Parâmetros das Estratégias de Evolução Diferencial.

Parâmetro	Descrição	Valor Utilizado
CR	Taxa de cruzamento	1
F	Amplificação da diferença	1
N_{pop}	Tamanho da População	1500
QntInteracoes	Nº de Iterações	1500

Foi utilizado uma distribuição de probabilidade uniforme para a inicialização da população. A estratégia de seleção de país utilizada foi o torneio entre 2 indivíduos. Para a condição de parada do processo evolucionário foi definido um número máximo de 1500 iterações ou 150 iterações sem melhora na pontuação.

4.4 Experimentos com Clustal

Para fins de comparação com o método de alinhamento progressivo foram obtidos os alinhamentos sugeridos pelo Clustal para cada um dos 50 grupos de sequências. Foi utilizado o Clustal versão W2 disponível no website do EMBL-EBI (`http://www.ebi.ac.uk/Tools/msa/clustalw2/`) utilizando os parâmetros padrões da ferramenta, conforme a Tabela 4.2.

A função objetivo foi aplicada para cada alinhamento sugerido pelo Clustal e os resultados obtidos pelas abordagens de GA, DE e Clustal foram comparados usando o teste não-paramétrico de Wilcoxon para amostras pareadas.

Tabela 4.2: Parâmetros do Clustal.

Parâmetro	Descrição	Valor Utilizado
dnamatrix	Matriz de pontuação utilizada	IUB
gapopen	Penalidade de criação de lacunas	-10
noendgaps	Penalizar separação de lacunas no fim do alinhamento?	no
gapext	Penalidade por extensão de lacunas	-0.2
gapdist	Penalidade por separação de lacunas	5
iteration	Estratégia de iteração	none
numiter	Numero máximo de iterações	1

4.5 Função Objetivo

A função objetivo adotada neste trabalho foi formulada inicialmente em (SOUZA; SANTOS; YARA, 2013). A função objetivo é definida na Equação 4.1 e busca atender os vários aspectos do problema de MSA. A ideia da função adotada é premiar aspectos desejáveis e penalizar os não-desejáveis. Para realizar isso a função objetivo é definida como uma combinação linear de outras quatro funções.

$$f(\mathbf{w}_i) = \omega_{\text{SP}} f_{\text{SP}}(\mathbf{w}_i) + \omega_{\text{Penal}} f_{\text{Penal}}(\mathbf{w}_i) + \omega_{\text{Corresp}} f_{\text{Corresp}}(\mathbf{w}_i) + \omega_{\text{Consec}} f_{\text{Consec}}(\mathbf{w}_i), \quad (4.1)$$

onde, ω_{SP}, ω_{Penal}, ω_{Corresp} e ω_{Consec} são os ponderadores das funções componentes, usados para configurar a influência de cada função de pontuação correspondente sobre o valor total da função objetivo. Os valores utilizados para os parâmetros estão na Tabela 4.3.

Tabela 4.3: Multiplicadores das Funções Componentes.

Parâmetro	Valor Utilizado
ω_{SP}	1,0
ω_{Penal}	1,0
ω_{Corresp}	2,0
ω_{Consec}	1,0

O primeiro termo da função objetivo é definido pela Equação 4.2. A matriz $w_{n,s}$ corresponde ao alinhamento resultante, sendo n o índice da sequência alinhada e s o índice da coluna do alinhanto. Em $f_{\text{SP}}(\mathbf{w})$ é realizado o somatório para cada coluna do alinhamento de todas as correspondências entre bases de uma mesma coluna, onde, o peso de cada correspondência seja determinado através de uma matriz de similaridades $M(i \times j)$.

$$f_{\text{SP}}(\mathbf{w}) = \sum_{s=1}^{S} \sum_{n=2}^{N} \text{IUB}(w_{1,s}, w_{n,s}), \tag{4.2}$$

A matriz de similaridade utilizada foi a matriz IUB, que corresponde à matriz padrão usada pelo Clustal em alinhamento de sequências de DNA. Na matriz IUB, cada elemento $m_{i,j}$ é um valor real representativo da similaridade entre a i-ésima e a j-ésima base. Todos os elementos da diagonal principal, sendo os representativo das correspondências (*matches*), são preenchidos com 1.9, enquanto que os elementos restantes, conhecidos como não-correspondentes (*mismatches*) são preenchidos com zeros. A componente f_{Penal} é a função para o cálculo das penalidades pela inserção das lacunas. O valor da penalidade atribuída para cada lacuna é a soma da penalidade relativa à sua posição dentro da sequência, chamada de penalidade sob posicionamento horizontal, e da penalidade relativa à quantidade de lacunas existentes em outras sequências que sofreram penalização horizontal na mesma posição, chamada de penalidade sob posicionamento vertical, sendo

definida como:

$$f_{\text{Penal}}(\mathbf{w}) = \omega_V \text{Penal}_V(\mathbf{w},S) + \omega_H \text{Penal}_H(\mathbf{w},S) \tag{4.3}$$

onde ω_H corresponde ao peso da componente de penalidade horizontal, $\text{Penal}_H(\mathbf{w},S)$, definida por

$$\text{Penal}_H(\mathbf{w},S) = \sum_{n=1}^{N}\sum_{s=1}^{S} \text{gap}_H(\mathbf{w},n,s), \tag{4.4}$$

onde $\text{gap}_H(\mathbf{w},n,s)$ é a penalidade horizontal para a s-ésima linha, definida como:

$$\text{gap}_H(\mathbf{w},n,s) = \begin{cases} 0, & s \in C_1 \\ \text{Penal}_{\text{gapExt}}, & s \in C_2 \\ \text{Penal}_{\text{gapOpen}}, & s \in C_3 \end{cases}, \tag{4.5}$$

$$C_1 = \{s = 1 \vee s < p_{\text{FB}}(\mathbf{w},n) \vee s > p_{\text{LB}}(\mathbf{w},n)\}, \tag{4.6}$$

$$C_2 = \{s > 1 \wedge \text{gap}_H(\mathbf{w},n,s-1) < 0\}, \tag{4.7}$$

$$C_3 = \{s > 1 \wedge \text{gap}_H(\mathbf{w},n,s-1) \geq 0\}, \tag{4.8}$$

e ω_V corresponde ao peso da componente de penalidade vertical, definida como:

$$\text{Penal}_V(\mathbf{w},S) = \sum_{s=1}^{S} \frac{\text{Penal}_{\text{gapOpen}}}{\text{ColumnGaps}(\mathbf{w},s)}, \tag{4.9}$$

onde $\text{ColumnGaps}(\mathbf{w},s)$ corresponde à quantidade de lacunas posicionadas na s-ésima posição do alinhamento representado por $\mathbf{w}$ e que sofreram penalização horizontal, com $0 \leq \text{ColumnGaps}(\mathbf{w},s) \leq N$; $p_{\text{FB}}(\mathbf{w},n)$ e $p_{\text{LB}}(\mathbf{w},n)$ são as posições da primeira e última bases posicionadas na n-ésima sequência do alinhamento $\mathbf{w}$, respectivamente. As constantes $\text{Penal}_{\text{gapOpen}}$ e $\text{Penal}_{\text{gapExt}}$ correspondem respectivamente às penalidades por abertura de lacuna e por extensão de lacuna. Nesta aplicação foram utilizados $\text{Penal}_{\text{gapOpen}} = -1$ e $\text{Penal}_{\text{gapExt}} = -0,2$.

A componente $f_{\text{Corresp}}(\mathbf{w})$ trata do número de correspondências (matches)

identificadas em cada coluna do alinhamento $\mathbf{w}$, sendo definida como:

$$f_{\text{Corresp}}(\mathbf{w}) = -\frac{N\text{ColumnGaps}(\mathbf{w},s)}{N - \sum_{s=2}^{S} \text{NumCorresp}(\mathbf{w},s)}, \tag{4.10}$$

onde $\text{NumCorresp}(\mathbf{w},s)$ representa o número de correspondências identificadas na s-ésima posição (coluna) do alinhamento $\mathbf{w}$, com $0 \leq \text{ColumnGaps}(\mathbf{w},s) \leq N-1$.

A componente $f_{\text{Consec}}(\mathbf{w})$ é a função que calcula a pontuação relativa à consecutividade de colunas de alta similaridade, definida por:

$$f_{\text{Consec}}(\mathbf{w}) = \sum_{s=1}^{S} \text{Consec}(\mathbf{w},s), \tag{4.11}$$

com

$$\text{Consec}(\mathbf{w},s) = \begin{cases} 0, & s = 1 \\ \frac{\text{NumCorresp}(\mathbf{w},s-1)}{(N-\text{NumCorresp}(\mathbf{w},s))^2}, & s \geq 1 \end{cases}, \tag{4.12}$$

onde s e n são respectivamente o número de colunas (posições) e o número de sequências (linhas) do alinhamento, com $1 \leq s \leq S$ e $1 \leq n \leq N$, sendo S o total de colunas e N o número de sequências do alinhamento.

5

RESULTADOS E DISCUSSÕES

5.1 Resultados

5.1.1 Testes de Wilcoxon

Os resultados dos testes de Wilcoxon comparando os valores obtidos pelo Clustal e as abordagens evolucionárias (DE e GA) estão na Tabela 5.1. Foi admitido um intervalo de confiança de 0.05. Os resultados foram favoráveis as estratégias de DE e sinalizam a equivalência do GA com Clustal.

Tabela 5.1: Evolução Diferencial superanado Clustal e Equivalência entre Algoritmos Genétivos e Clustal.

Abordagem	Hipótese Alternativa (H1)	Valor p	H1 Aceita?	Desfecho
Rand-1-Bin (R1B)	H1:R1B <Clustal H1:R1B $\neq$ Clustal	0,99999 0,00002	Não Sim	R1B foi superior significativamente
Rand-2-Bin (R2B)	H1:R2B <Clustal H1:R2B $\neq$ Clustal	0,99999 0,00001	Não Sim	R2B foi superior significativamente
Best-1-Bin (B1B)	H1:B1B <Clustal H1:B1B $\neq$ Clustal	0,99999 0,00003	Não Sim	B1B foi superior significativamente
Best-2-Bin (B2B)	H1:B2B <Clustal H1:B2B $\neq$ Clustal	0,99998 0,00005	Não Sim	B2B foi superior significativamente
Rand-to-Best-1-Bin (RB1B)	H1:RB1B <Clustal H1:RB1B $\neq$ Clustal	0,99999 0,00003	Não Sim	RB1B foi superior significativamente
GA	H1:GA <Clustal H1:GA $\neq$ Clustal	0.78300 0,78690	Não Não	GA foi equivalente

Os resultados dos testes de Wilcoxon comparando os valores obtidos entre as abordagens evolucionárias DE e GA estão na Tabela 5.2. Foi admitido um intervalo de confiança de 0.05. Os resultado foram favoráveis as estratégias de DE.

Tabela 5.2: Evolução Diferencial superando Algoritmos Genéticos.

Abordagem	Hipótese Alternativa (H1)	Valor p	H1 Aceita?	Desfecho
Rand-1-Bin (R1B)	H1:R1B <GA	0,99999	Não	R1B foi superior significativamente
	H1:R1B $\neq$ GA	0,00000	Sim	
Rand-2-Bin (R2B)	H1:R2B <GA	0,99999	Não	R2B foi superior significativamente
	H1:R2B $\neq$ GA	0,00000	Sim	
Best-1-Bin (B1B)	H1:B1B <GA	0,99999	Não	B1B foi superior significativamente
	H1:B1B $\neq$ GA	0,00000	Sim	
Best-2-Bin (B2B)	H1:B2B <GA	0,99998	Não	B2B foi superior significativamente
	H1:B2B $\neq$ GA	0,00000	Sim	
Rand-to-Best-1-Bin (RB1B)	H1:RB1B <GA	0,99999	Não	RB1B foi superior significativamente
	H1:RB1B $\neq$ GA	0,00000	Sim	

5.1.2 Gráficos

Na Figura 5.1 são apresentados os resultados de todas as abordagens. Seguindo a ordem da esquerda para direita temos os resultados da abordagem *Best/1/Bin* (B1B), *Best/2/Bin* (B2B), *Rand/1/Bin* (R1B), *Rand/2/Bin* (R2B), *Rand-To-Best/1/Bin* (RB1B), GA e finalmente Clustal. Os limites extremos correspondem aos valores de máximo e mínimo, enquanto que a marca central é a mediana. As posições abaixo e acima da mediana são os percentis 25 e 75.

Na Figura 5.2 temos em destaque uma linha mostrando que 75% dos resultados atingidos pelo Clustal estão abaixo da mediana das estratégias de DE (com exceção B2B que possui uma assimetria negativa). Os resultados do GA motram que mais de 75% dos valores atingidos estão abaixos dos valores médios de DE.

Na Figura 5.3 temos em destaque uma linha mostrando que 25% dos resultados atingidos pelo Clustal possuem valores inferiores aos valores mínimos atingidos pelas estratégias de DE.

Na Figura 5.4 temos em destaque mostrando que as estratégias DE possuem valores médios próximos, com exceção da B2B que possui uma assimetria negativa nos valores.

Na Figura 5.5 temos em destaque mostrando que os valores máximos e mínimos atingidos pelas estratégias DE estão mais próximas de seus respectivos valores médios quando comparada ao Clustal, sinalizando uma maior estabilidade das abordagens de DE.

Na Figura 5.6 temos em destaque mostrando que 50% dos valores próximos ao valores médios (25% dos valores acima e 25% dos valores abaixo da mediana) estão distribuídos em um intervalor menor nas estratégias de DE quando comparada ao Clustal, onde os valores estão mais dispersos. Este comportamente sinaliza uma maior precisão das estratégias de DE em relação ao Clustal.

Figura 5.1: Resultado Geral.

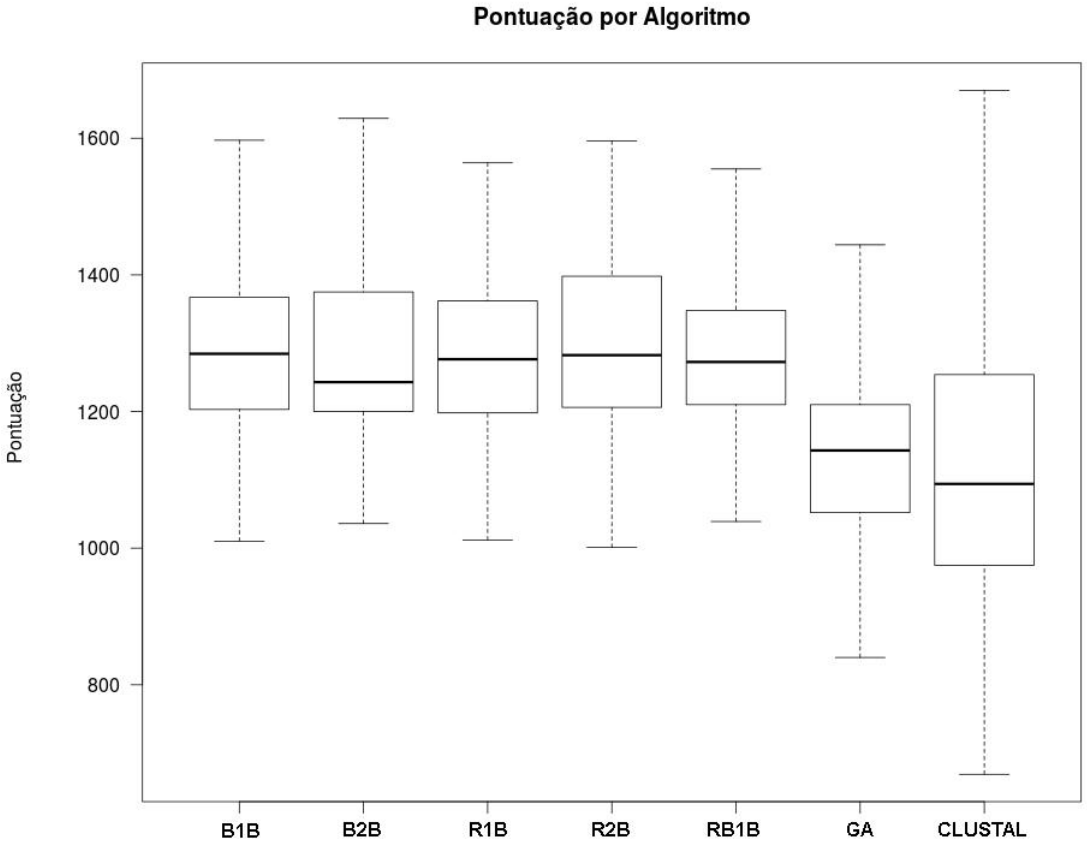


Figura 5.2: Resultado Geral - 75º Percentil do Clustal.

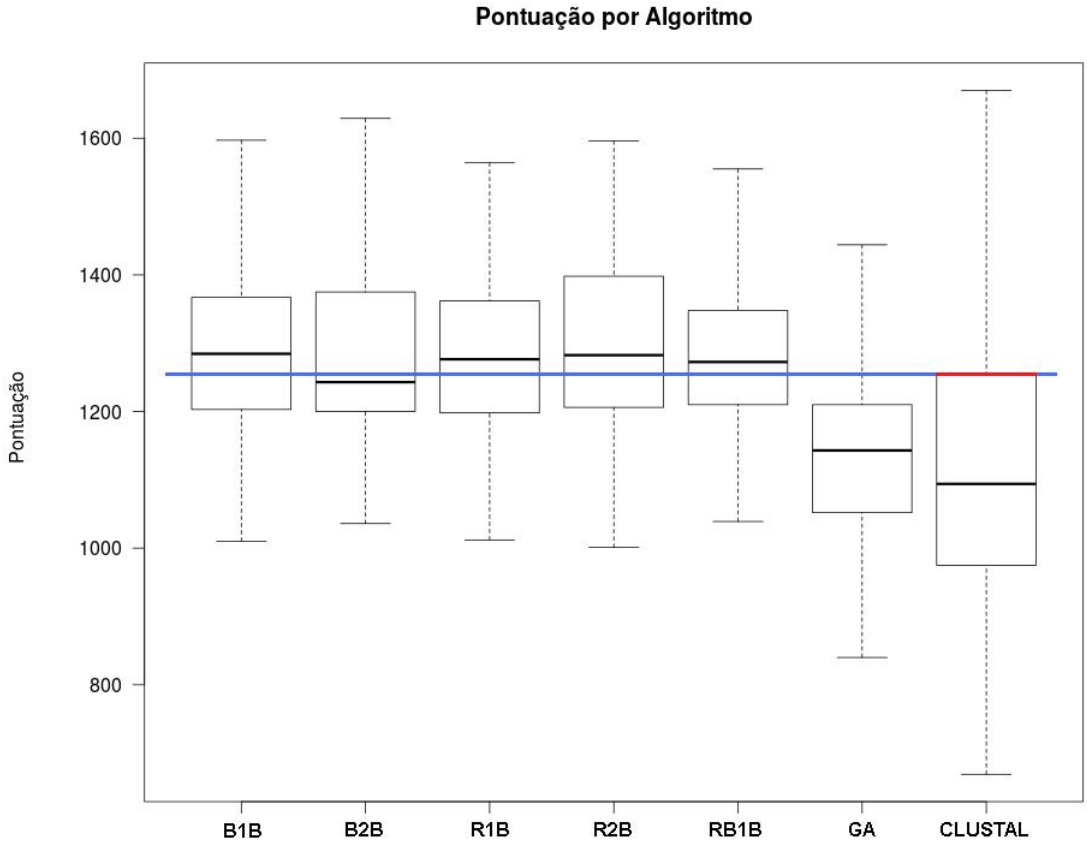


Figura 5.3: Resultado Geral - Valores Mínimos.

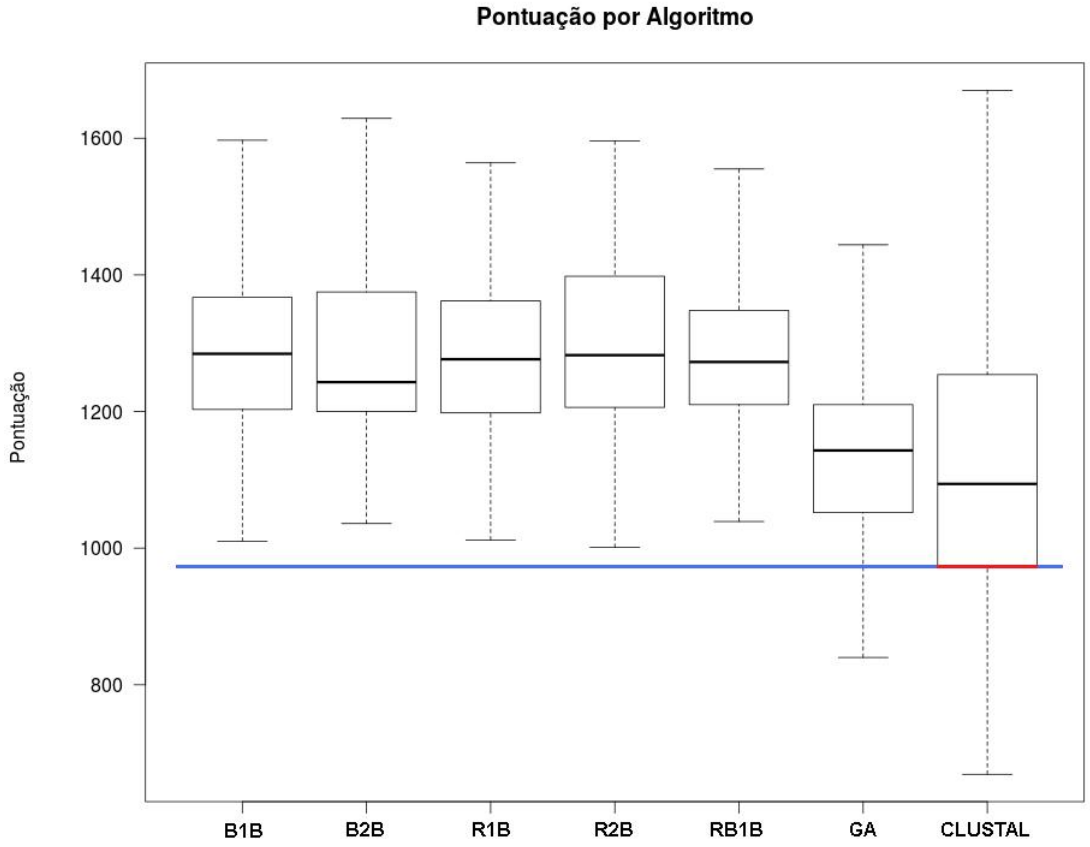


Figura 5.4: Resultado Geral - Medianas.

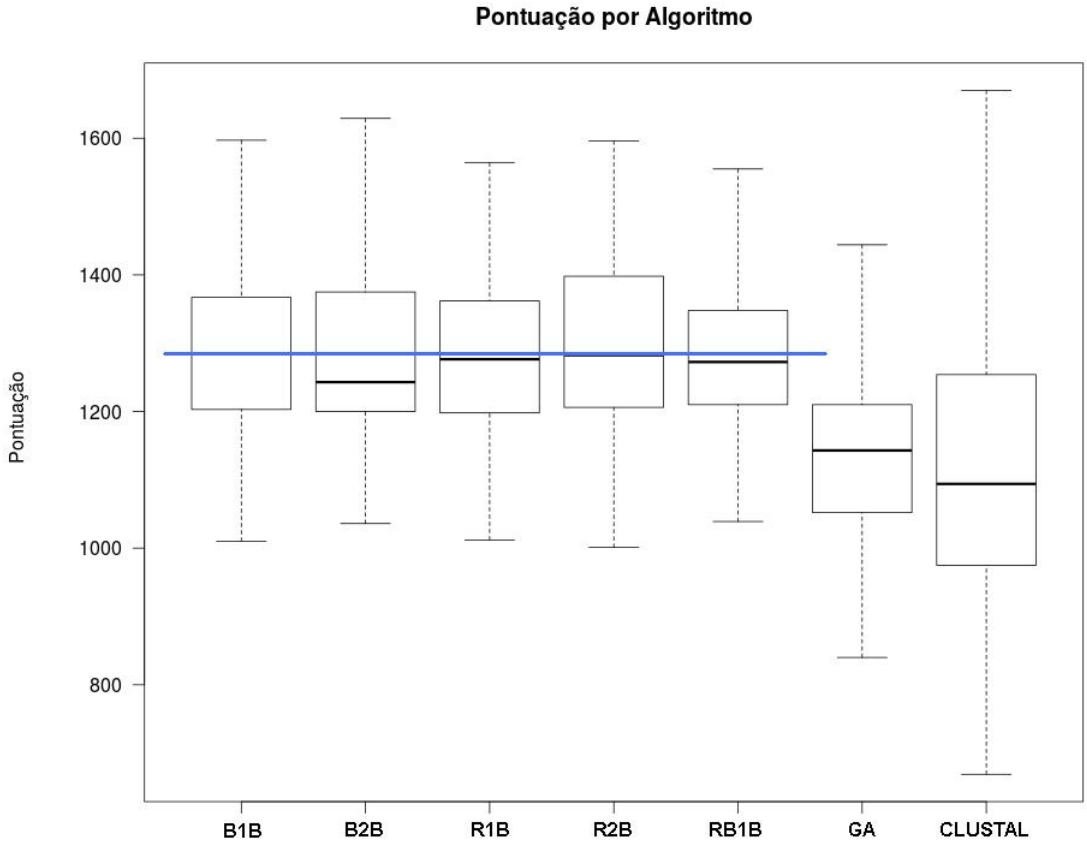


Figura 5.5: Resultado Geral - Estabilidade.

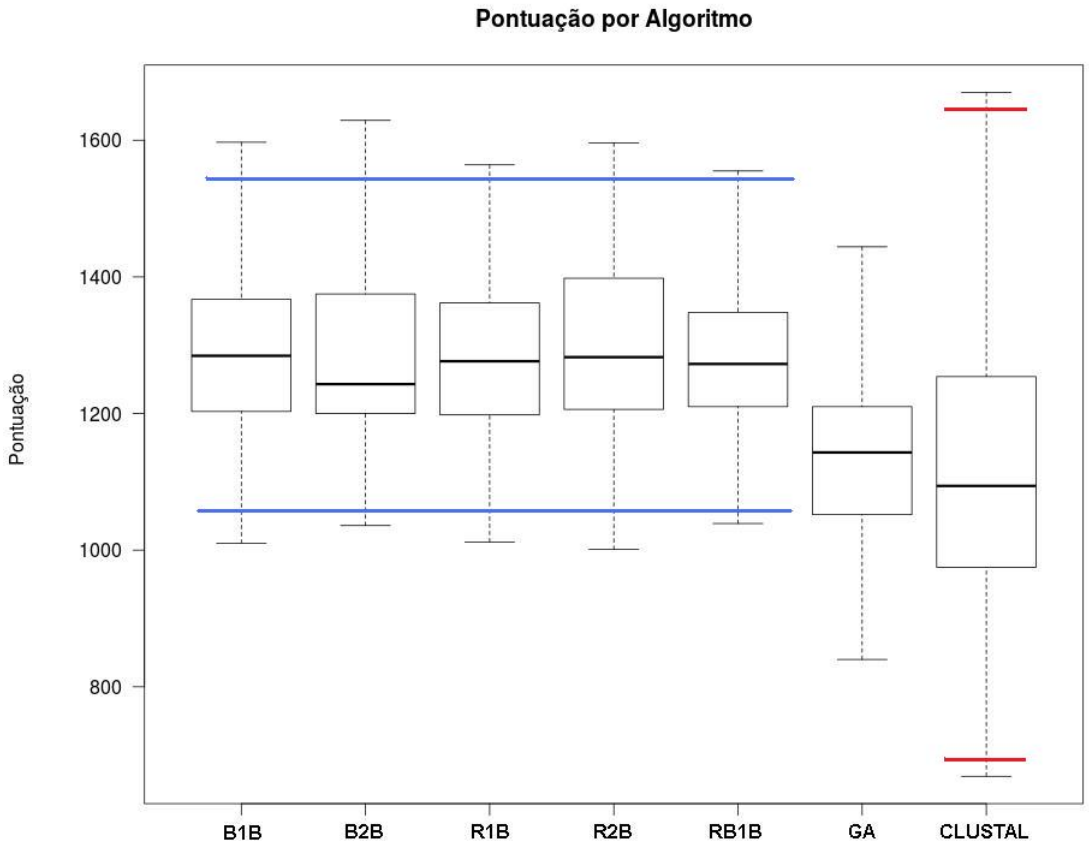
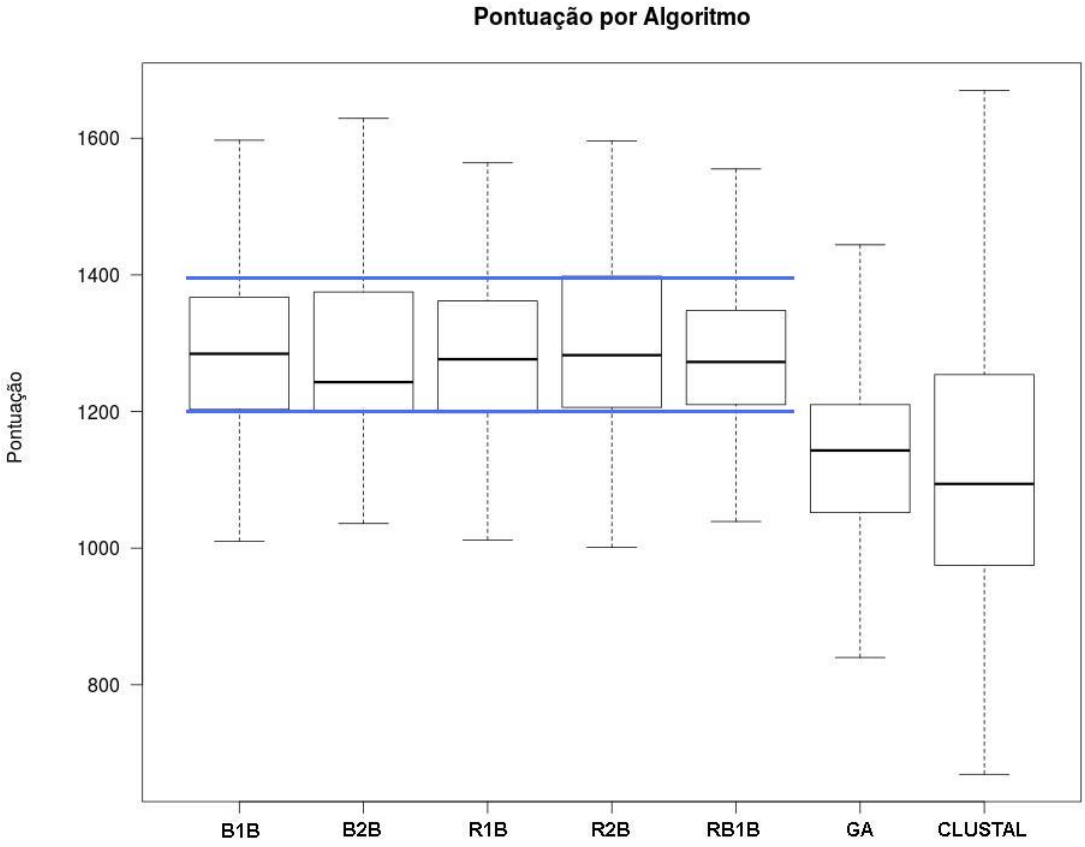


Figura 5.6: Resultado Geral - Precisão.



5.2 Discussões

Foi visto uma abordagem de otimização de uma função objetivo proposta para o problema de MSA, utilizando Evolução Diferencial. Foram utilizadas 5 estratégias de Evolução Diferencial: Best/1/Bin, Best/2/Bin, Rand/1/Bin, Rand/2/Bin e Rand-To-Best/1/Bin. As pontuações obtidas para cada estratégia foram comparadas com as pontuações obtidas com os alinhamentos realizados pelo método de alinhamento progressivo Clustal.

Para a maioria dos ensaios realizados em cada estratégia de DE testada, a meta-heurística da DE mostrou-se eficaz para o problema de alinhamento múltiplo de sequências, quando comparamos os seus resultados com os obtidos pelo método de alinhamento progressivo Clustal.

No geral, a abordagem com Evolução Diferencial conseguiu obter melhores resultados em até 80% dos ensaios. As menores porcentagem de superação ficaram com as estratégias Rand/1/Bin e Rand/2/Bin que superaram o Clustal em 76% dos ensaios. As maiores pontuações ficaram com as estratégias Rand-To-Best/1/Bin e Best/1/Bin que superaram o Clustal em 80% dos ensaios.

6

CONCLUSÃO

6.1 Conclusões gerais

Ao longo deste trabalho foi definido um estudo da aplicação de algoritmos evolucionários para resolver problemas de alinhamento múltiplo de sequências biológicas. Foram comparadas duas abordagens de algoritmos evolucionários (Evolução Diferencial e Algoritmos Genéticos) com a abordagem do estado da arte Clustal. Os resultados demonstraram que o uso de da Evolução Diferencial leva a resultados melhores do que com Clustal. Os Algoritmos genéticos demonstraram resultados equivalentes ao Clustal.

6.2 Publicações

A publicação (SILVA et al., 2014) foi realizada no Encontro Nacional de Inteligência Artificial e Computacional (ENIAC) em 2014 sob o título de *Uma Abordagem de Alinhamento Múltiplo de Sequências Utilizando Evolução Diferencial*; os autores foram Antônio L. V. da Silva, Rodrigo G. de Souza, Ricardo Yara e Wellington P. dos Santos.

A publicação (SOUZA et al., 2014) foi realizada no Encontro Nacional de Inteligência Artificial e Computacional (ENIAC) em 2014 sob o título de *Alinhamento múltiplo de sequências utilizando otimização dialética*; os autores foram Rodrigo G. de Souza, Antônio L. V. da Silva, Ricardo Yara e Wellington P. dos Santos.

6.3 Trabalhos futuros

Como forma de ampliar a investigação na obtenção de melhores alinhamentos, vem sendo estudada a utilização de alinhamentos já obtidos por métodos progressivos como dados de entrada no processo evolucionário, para constatar se os alinhamentos obtidos ao término do processo irão possuir uma melhor pontuação.

Referências

ALTSCHUL, S. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acids research, [S.l.], v.25, n.17, p.3389—3402, September 1997.

ALTSCHUL, S. F. et al. Basic local alignment search tool. Journal of molecular biology, [S.l.], v.215, n.3, p.403–410, 1990.

BAXEVANIS, A.; OUELLETTE, B. Bioinformatics: a practical guide to the analysis of genes and proteins. [S.l.]: John Wiley & Sons, 2001. (Methods of Biochemical Analysis).

CARRILLO, H.; LIPMAN, D. The Multiple Sequence Alignment Problem in Biology. SIAM J. Appl. Math., Philadelphia, PA, USA, v.48, n.5, p.1073–1082, Oct. 1988.

CORPET, F. Multiple sequence alignment with hierarchical clustering. Nucleic acids research, [S.l.], v.16, n.22, p.10881–10890, 1988.

CUTELLO, V. et al. Protein multiple sequence alignment by hybrid bio-inspired algorithms. Nucleic acids research, [S.l.], v.39, n.6, p.1980–1992, 2011.

DAS, S. et al. Differential Evolution Using a Neighborhood-Based Mutation Operator. Evolutionary Computation, IEEE Transactions on, [S.l.], v.13, n.3, p.526–553, June 2009.

DORIGO, M.; DI CARO, G.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. Artificial life, [S.l.], v.5, n.2, p.137–172, 1999.

DORTMUND, T. Evolutionary Algorithms in Theory and Practice : evolution strategies, evolutionary programming, genetic algorithms: evolution strategies, evolutionary programming, genetic algorithms. [S.l.]: Oxford University Press, USA, 1995.

EDGAR, R. C. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC bioinformatics, [S.l.], v.5, n.1, p.113, 2004.

EDGAR, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research, [S.l.], v.32, n.5, p.1792–1797, 2004.

EIBEN, A. E.; SMITH, J. E. Introduction to evolutionary computing. [S.l.]: Springer Science & Business Media, 2003.

FENG, D.-F.; DOOLITTLE, R. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. Journal of Molecular Evolution, [S.l.], v.25, n.4, p.351–360, 1987.

GOLDBERG, D. Genetic Algorithms in Search, Optimization, and Machine Learning. [S.l.]: Addison-Wesley, 1989. (Artificial Intelligence).

GONDRO, C.; KINGHORN, B. A simple genetic algorithm for multiple sequence alignment. Genet. Mol. Res, [S.l.], v.6, n.4, p.964–982, 2007.

GUSFIELD, D. Algorithms on Strings, Trees and Sequences: computer science and computational biology. [S.l.]: Cambridge University Press, 1997.

HENIKOFF, S.; HENIKOFF, J. G. Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences, [S.l.], v.89, n.22, p.10915–10919, 1992.

HIGGINS, D. G.; BLEASBY, A. J.; FUCHS, R. CLUSTAL V: improved software for multiple sequence alignment. Computer applications in the biosciences : CABIOS, [S.l.], v.8, n.2, p.189–191, 1992.

HIGGINS, D.; SHARP, P. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene, [S.l.], v.73, n.1, p.237—244, December 1988.

HOGEWEG, P.; HESPER, B. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. Journal of Molecular Evolution, [S.l.], v.20, n.2, p.175–186, 1984.

HOLLAND, J. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. [S.l.]: University of Michigan Press, 1975.

HU, G.-W. Chaos-differential Evolution for Multiple Sequence Alignment. In: INTELLIGENT INFORMATION TECHNOLOGY APPLICATION, 2009. IITA 2009. THIRD INTERNATIONAL SYMPOSIUM ON. Anais... [S.l.: s.n.], 2009. v.2, p.556–558.

JONES, D. T. Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology, [S.l.], v.292, n.2, p.195 – 202, 1999.

LAYEB, A.; MESHOUL, S.; BATOUCHE, M. Multiple sequence alignment by quantum genetic algorithm. In: PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, 2006. IPDPS 2006. 20TH INTERNATIONAL. Anais... [S.l.: s.n.], 2006. p.8 pp.–.

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

LIPMAN, D. J.; PEARSON, W. R. Rapid and sensitive protein similarity searches. Science, [S.l.], v.227, n.4693, p.1435–1441, 1985.

MASULLI, F.; PETERSON, L.; TAGLIAFERRI, R. Computational Intelligence Methods for Bioinformatics and Biostatistics: 6th international meeting, cibb 2009, genoa, italy, october 15-17, 2009, revised selected papers. [S.l.]: Springer, 2010. (LNCS sublibrary: Bioinformatics).

MESHOUL, S.; LAYEB, A.; BATOUCHE, M. A Quantum Evolutionary Algorithm for Effective Multiple Sequence Alignment. In: BENTO, C.; CARDOSO, A.; DIAS, G. (Ed.). Progress in Artificial Intelligence. [S.l.]: Springer Berlin Heidelberg, 2005. p.260–271. (Lecture Notes in Computer Science, v.3808).

MITCHELL, M. An introduction to genetic algorithms. [S.l.]: MIT press, 1998.

NCBI. Bethesda (MD): national center for biotechnology information (us). 2015.

NEEDLEMAN, S. B.; WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology, [S.l.], v.48, n.3, p.443–453, 1970.

NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. The Computer Journal, [S.l.], v.7, n.4, p.308–313, 1965.

NOTREDAME, C. Recent progress in multiple sequence alignment: a survey. Pharmacogenomics, [S.l.], v.3, n.1, p.131–144, 2002.

NOTREDAME, C.; HIGGINS, D. G. SAGA: sequence alignment by genetic algorithm. Nucleic acids research, [S.l.], v.24, n.8, p.1515–1524, 1996.

NOTREDAME, C.; HIGGINS, D. G.; HERINGA, J. T-Coffee: a novel method for fast and accurate multiple sequence alignment. Journal of molecular biology, [S.l.], v.302, n.1, p.205–217, 2000.

NOTREDAME, C.; HOLM, L.; HIGGINS, D. G. COFFEE: an objective function for multiple sequence alignments. Bioinformatics, [S.l.], v.14, n.5, p.407–422, 1998.

Plus and Minus Method (Sanger F et al 1975 J Mol Biol 94:441). In: Encyclopedia of Genetics, Genomics, Proteomics and Informatics. [S.l.]: Springer Netherlands, 2008. p.1514–1514.

PRICE, K.; STORN, R.; LAMPINEN, J. Differential Evolution: a practical approach to global optimization. [S.l.]: U.S. Government Printing Office, 2005. (Natural Computing Series).

PRICE, W. L. A controlled random search procedure for global optimisation. The Computer Journal, [S.l.], v.20, n.4, p.367–370, 1977.

RAHNAMAYAN, S.; TIZHOOSH, H.; SALAMA, M. Opposition-Based Differential Evolution. Evolutionary Computation, IEEE Transactions on, [S.l.], v.12, n.1, p.64–79, Feb 2008.

SETUBAL, J. C.; MEIDANIS, J. Introduction to Computational Molecular Biology. [S.l.]: PWS Pub., 1997. (Computer Science Series).

SIEVERS, F. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular systems biology, [S.l.], v.7, n.1, p.539, 2011.

SILVA, A. L. V. d. et al. Uma Abordagem de Alinhamento Múltiplo de Sequências Utilizando Evolução Diferencial. Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), São Carlos, 2014.

SIMOSSIS, V. A.; HERINGA, J. PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. Nucleic acids research, [S.l.], v.33, n.suppl 2, p.W289–W294, 2005.

SIMOSSIS, V.; KLEINJUNG, J.; HERINGA, J. An overview of multiple sequence alignment. Current protocols in bioinformatics, [S.l.], p.3–7, 2003.

SIPPL, M. J. Biological sequence analysis. Probabilistic models of proteins and nucleic acids, edited by R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. 1998. Cambridge: cambridge university press. 356 pp.. PRS, [S.l.], v.8, p.695–695, 3 1999.

SMITH, T. F.; WATERMAN, M. S. Identification of common molecular subsequences. Journal of molecular biology, [S.l.], v.147, n.1, p.195–197, 1981.

SOUZA, R. G. d. et al. Alinhamento múltiplo de sequências utilizando otimização dialética. Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), São Carlos, 2014.

SOUZA, R. G. d.; SANTOS, W. d.; YARA, R. Uma abordagem para alinhamento múltiplo de sequências de DNA usando Algoritmos Genéticos e número variável de lacunas. Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), Fortaleza, 2013.

STORN, R.; PRICE, K. Differential Evolution: a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization, Hingham, MA, USA, v.11, n.4, p.341–359, Dec. 1997.

TAYLOR, W. R. A flexible method to align large numbers of biological sequences. Journal of Molecular Evolution, [S.l.], v.28, n.1-2, p.161–169, 1988.

THOMPSON, J. D.; HIGGINS, D. G.; GIBSON, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research, [S.l.], v.22, n.22, p.4673–4680, 1994.

THOMSEN, R.; FOGEL, G.; KRINK, T. A Clustal alignment improver using evolutionary algorithms. In: EVOLUTIONARY COMPUTATION, 2002. CEC '02. PROCEEDINGS OF THE 2002 CONGRESS ON. Anais... [S.l.: s.n.], 2002. v.1, p.121–126.

VESTERSTROM, J.; THOMSEN, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: EVOLUTIONARY COMPUTATION, 2004. CEC2004. CONGRESS ON. Anais... [S.l.: s.n.], 2004. v.2, p.1980–1987 Vol.2.

WHITLEY, D. A genetic algorithm tutorial. Statistics and computing, [S.l.], v.4, n.2, p.65–85, 1994.

ZHANG, C.; WONG, A. K. A genetic algorithm for multiple molecular sequence alignment. Computer applications in the biosciences : CABIOS, [S.l.], v.13, n.6, p.565–581, 1997.

ZHANG, G.-Z.; HUANG, D.-S. Aligning multiple protein sequence by an improved genetic algorithm. In: NEURAL NETWORKS, 2004. PROCEEDINGS. 2004 IEEE INTERNATIONAL JOINT CONFERENCE ON. Anais... [S.l.: s.n.], 2004. v.2, p.1179–1183 vol.2.

ZHANG, J.; SANDERSON, A. JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, [S.l.], v.13, n.5, p.945–958, Oct 2009.