UNIVERSIDADE DE CAXIAS DO SUL

GUSTAVO PISTORE

Uso de Algoritmos Genéticos para o aumento da assertividade no reconhecimento e identificação de promotores no BACPP

Caxias do Sul
2017

Uso de Algoritmos Genéticos para o aumento da assertividade no reconhecimento e identificação de promotores no BACPP

por

Gustavo Pistore

Projeto de Diplomação submetido ao curso de Bacharelado em Ciência da Computação da Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul, como requisito obrigatório para graduação.

Projeto de Diplomação

Orientador: André Luis Martinotto
Banca examinadora:

Helena Graziottin Ribeiro
EXATAS/UCS
Scheila de Avila e Silva
EXATAS/UCS

Projeto de Diplomação apresentado em
Dezembro de 2017

Andre Gustavo Adami
Coordenador

" Aos meus pais Vilson e Janete, minha irmã Camila,
minha namorada Lisandra, meu padrinho Joanez
e a toda minha família que, com muito carinho e apoio,
não mediram esforços para que eu chegasse até esta etapa de minha vida.".

"A menos que modifiquemos a nossa maneira de pensar,
não seremos capazes de resolver os problemas causados
pela forma como nos acostumamos a ver o mundo".

Albert Einstein

RESUMO

A biologia molecular computacional é uma área de conhecimento que possui como um dos principais objetivos a análise dos dados resultantes do sequenciamento de genomas. Dentro deste contexto, foram desenvolvidos diversos softwares para a análise de sequências biológicas, como por exemplo o software BACPP, que foi desenvolvido pelo grupo de Bioinformática da UCS e que efetua a caracterização e a predição de promotores em bactérias Gram-negativas.

Para efetuar a caracterização e o reconhecimento de promotores dentro de um genoma, o BACPP utiliza-se de regras obtidas através do treinamento de Redes Neurais Artificiais (RNA). Os valores de saída da RNA são então ponderados de forma a obter-se uma melhor assertividade, sendo que os valores utilizados para essa ponderação foram escolhidos empiricamente após testes efetuados com valores inteiros entre -10 e 10.

O objetivo deste trabalho consistiu em aumentar a performance da predição de promotores, através utilização de Algoritmos Genéticos para obtenção de valores mais adequados para a ponderação das regras de saída das RNAs do BACPP. A partir da implementação desenvolvida, foram gerados novos valores de ponderação para a predição de promotores no BACPP. Estes valores de ponderação foram comparados aos valores anteriores existentes no BACPP, obtendo-se uma melhora melhoria da assertividade do método.

Palavras-chave: Algoritmos Genéticos, Genoma, Minimização.

BACPP Promoters identification and recognition Assertiveness increase through the use of Genetic Algorithms

ABSTRACT

Computational molecular biology is a knowledge area that have as one of its main objective, the genome sequencing data analysis. Within this context were developed several biological sequence analysis software, such as BACC software, developed by the UCS Bioinformatics group. BACPP perform the Gram-negative promoter prediction.

To perform the description and recognition of promoters inside a genome sequence, BACPP applies the rules obtained through Artificial Neural Network training. After, the rules of the Artificial Neural Network are weighted to get better assertiveness. These weighting values were chosen empirically after tests with integers between -10 and 10.

This academic work objective was to increase the assertiveness of the promoters prediction, applying genetic algorithms in order to achieve better weighting values to weight the artificial neural networks outputs on BACPP. From the developed implementation, new weighting values were generated to BACPP's promoter prediction. These weighting values were compared to the original ones on BACPP and showed an assertiveness increase for the method.

Keywords: Genetic Algorithms, Genome,Minimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AG	Algoritmos Genéticos
AP	Avaliação de Aptidão
BACPP	*Bacterial Promoter Prediction*
BLAST	*Basic Local Alignment Search Tool*
CE	Computação Evolutiva
DEAP	*Distributed Evolutionary Algorithms in Python*
DNA	Ácido Desoxirribonucleico
EO	*Evolving Objects*
GAJIT	*Genetic Algorithm Java Implementation Toolkit*
GA Toolbox	*Genetic Algorithms Toolbox*
GAUL	*Genetic Algorithm Utility Library*
GAlib	*Genetic Algorithm Library*
NNPP	*Neural Network Promoter Prediction*
pySTEP	*Python Strongly Typed gEnetic Programming*
RNA	Redes Neurais Artificiais
RNA	Ácido Ribonucleico
RNAp	Ácido Ribonucleico polimerase
UCS	Universidade de Caxias do Sul

SUMÁRIO

1 INTRODUÇÃO

A biologia molecular é a área responsável pelo estudo da estrutura celular e dos seus constituintes moleculares, bem como da localização e das interações entre esses constituintes (ZAHA, 2014). Entre esses constituintes destaca-se o genoma que consiste basicamente em um repositório da informação do DNA (Ácido Desoxirribonucleico) que constituiu uma célula.

O genoma é composto principalmente pelas regiões gênicas e intergênicas. As regiões gênicas são conhecidas como regiões codificadoras de proteínas, ou seja, essas regiões possuem as informações necessárias para sintetizar as proteínas ou o RNA estável. Já as regiões intergênicas encontram-se entre o final de uma região codificadora e o início de outra (ZAHA, 2014).

Nas regiões anteriores às regiões gênicas encontram-se mecanismos regulatórios que são conhecidos como promotores. A iniciação da transcrição dos genes inicia quando a enzima RNAp liga-se a um promotor. As regiões promotoras são importantes uma vez que essas interferem na transcrição das regiões codificadoras, atuando na ativação ou inibição de determinados genes (ZAHA, 2014). O reconhecimento dos promotores depende de uma proteína de transcrição que é denominada de fator sigma, uma vez que sem a presença do fator sigma a ligação à região do DNA não ocorre.

A biologia molecular computacional é uma área multidisciplinar, envolvendo as áreas da computação, estatística, biologia e matemática. Essa possui como um dos principais objetivos a análise dos dados resultantes do sequenciamento de genomas (ZAHA, 2014). Dentro deste contexto, foram desenvolvidos diversos softwares para a análise de sequências biológicas, como por exemplo, o BLAST (*de Basic Local Alignment Search Tool*) e o ClustalO, que são utilizados no estudo de genomas. Nesta área encontra-se também o software BACPP, que foi desenvolvida pelo grupo de Bioinformática da UCS e que efetua a caracterização e a predição de promotores em bactérias Gram-negativas (SILVA, 2011). O estudo das regiões promotoras é importante uma vez que auxilia na compreensão dos genes e do modo como um organismo interage com o meio.

O BACPP foi concebido pela aplicação de Redes Neurais Artificiais (RNAs) (SILVA, 2011) para o reconhecimento e a caracterização de promotores em um genoma. A concepção deste iniciou-se com o treinamento da RNA para a identificação de promotores, sendo que as regras obtidas foram ponderadas utilizando-se valores inteiros, que são apresentados na Tabela 1. Esses valores foram obtidos empiricamente, após testes realizados com valores inteiros entre -10 e 10, sendo que esses são utilizados igualmente para todos os fatores sigmas existentes (SILVA, 2011). A partir destes valores gera-se um histograma, onde são definidos os valores de corte para a identificação se uma sequência é ou não um promotor (SILVA, 2011).

Tabela 1.1: Valores de ponderação de acordo com o escore obtido através da regra da rede neural

Escore obtido da regra da Rede neural	Valor de ponderação
Maior que 0.6	+6
0.5 – 0.59	+4
0.4 – 0.49	+2
0.3 – 0.39	+1
0.2 – 0.29	0
0.1 – 0.19	-1
Menor que 0.1	-3

Fonte: (SILVA, 2011)

Como já mencionado, os valores de ponderação da ferramenta BACPP foram selecionados empiricamente. Desta forma, neste trabalho foram usados Algoritmos Genéticos (CARVALHO, 2011) para a busca de valores mais adequados para a predição de regiões promotoras de bactérias Gran-Negativas. Além disso, foram definidos valores específicos de ponderação para cada um dos fatores sigmas.

1.1 Objetivo

O objetivo geral deste trabalho consistiu na obtenção de valores de ponderação para as saídas da RNA do BACPP por meio da utilização de Algoritmos Genéticos. Para que esse objetivo fosse atingido, os seguintes objetivos específicos foram realizados: f

- Utilização de Algoritmos Genéticos para obtenção de valores mais adequados para a ponderação das saída da RNA no BACPP;
- Validação da implementação;
- Comparação dos resultados com a implementação já existente.

1.2 Estrutura do Trabalho

Este trabalho apresenta a seguinte estrutura:

- O Capítulo 2 traz uma breve introdução da área de Biologia Molecular de forma a facilitar a compreensão deste trabalho, bem como de um entendimento do contexto em que este trabalho está inserido. Este capítulo aborda de forma sucinta o genoma, a expressão gênica dos organismos, bem como as regiões promotoras. Ele traz ainda informações sobre o BACPP, que serviu como base para o desenvolvimento deste trabalho.
- O Capítulo 3 introduz o problema de otimização e o uso de Algoritmos Genéticos. Mais especificamente, neste capítulo são descritas as características e o funcionamento de Algoritmos Genéticos.
- O Capítulo 4 compreende a análise de bibliotecas. Neste capítulo, é feita a análise e escolha da biblioteca de AGs que será utilizada para a implementação do software.
- O Capítulo 5 compreende a implementação do software bem como as comparações com a implementação anterior.
- O capítulo 6 apresenta as conclusões finais do trabalho, bem como uma sugestão de trabalhos futuros.

2 BIOLOGIA MOLECULAR

A biologia molecular relaciona-se com o estudo das macromoléculas celulares essenciais, como por exemplo, o RNA, o DNA e as proteínas. Os estudos nesta área iniciaram-se na segunda metade do século XX, porém apenas recentemente atingiram um grau de maturidade mais elevado (LIPAY, 2015). Ao longo do tempo, a biologia molecular acabou sendo associada à regulação, a função e a estrutura das vias de informação no nível molecular, ou seja, aos processos relativos à passagem da informação genética de uma geração para outra. Apesar de sua história relativamente breve, a biologia molecular apresenta um grande impacto no conhecimento humano, sendo essencial para as áreas de medicina, agricultura, ciência forense, etc (COX, 2012). Neste capítulo, serão abordados os conceitos básicos desta área, de modo a facilitar a compreensão deste trabalho.

2.1 Genoma

Todas as células vivas conhecidas, armazenam as suas informações hereditárias na forma de moléculas de DNA (Ácido Desoxirribonucleico) de fita dupla. Estas moléculas se apresentam como longas cadeias poliméricas pareadas e não-ramificadas, formadas por quatro tipos de nucleotídeos, também conhecidos como monômeros. Os monômeros são formados por duas partes que são: um açúcar (desoxirribose), que possui um grupo fosfato ligado a ele; e uma base, que pode ser Adenina (A), Tinina (T), Citosina (C) e Guanina (G). Na Figura 2.1(a), tem-se um exemplo de um monômero onde um açúcar encontra-se ligado a uma base de Guanina. Por sua vez o açúcar está ligado ao próximo por meio do grupo fosfato, formando uma cadeia polimérica composta de uma longa sequência linear, com as bases projetando-se dela. De fato, o polímero de DNA se estende através da adição de monômeros em uma das extremidades. Em uma fita simples, como visto na Figura 2.1(b), essas bases podem ser adicionadas em qualquer ordem, pois essas se ligam à próxima de forma independente da base (ALBERTS, 2011).

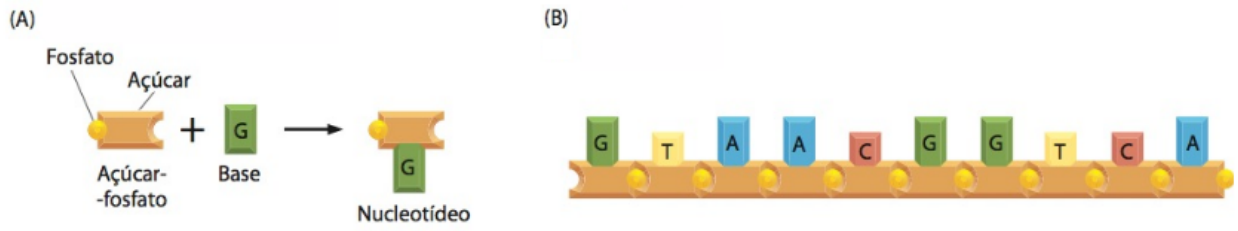


Figura 2.1: (a)Bloco de Construção de DNA.(b)Fita de DNA
Fonte: (ALBERTS, 2011)

Já na etapa em que é gerada a dupla fita, conhecida como Polimerização, as bases que se projetam da fita ligam-se com bases específicas da fita que está sendo polimerizada seguindo uma regra rigorosa, que é definida como: A liga-se com T, e C liga-se com G, como pode ser visto na Figura 2.2(a). Deste modo, uma estrutura de fita dupla é criada, composta de duas sequências exatamente complementares de bases, conforme visto na Figura 2.2(b) (ALBERTS, 2011).

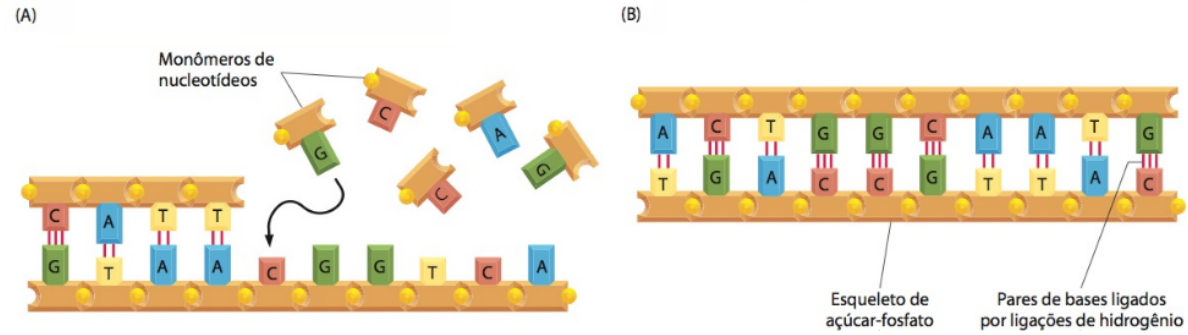


Figura 2.2: (a)Polimetização de uma nova fit a partir de um molde. (b)DNA de fita dupla.
Fonte: (ALBERTS, 2011)

Essas duas fitas torcidas entre si formam uma dupla-hélice, que pode ser observada na Figura 2.3. Esta dupla-hélice, que contém toda a informação hereditária de um organismo, é conhecida como genoma. A menor unidade do genoma que é capaz de representar um produto biológico e que possua alguma função no organismo é conhecida como gene. A quantidade de genes presentes no genoma de um organismo é variável, podendo ser de 480 genes no caso de organismos simples, até mais de 24000 genes no caso do Homo Sapiens (ALBERTS, 2011).

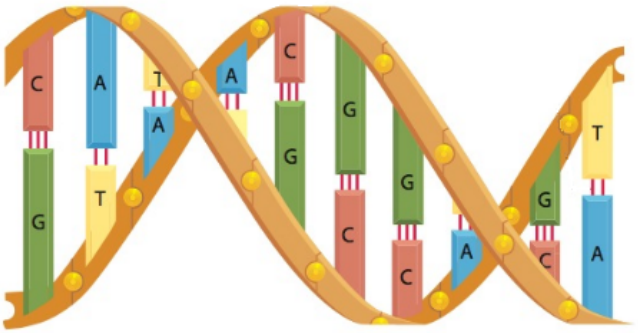


Figura 2.3: Dupla-hélice de DNA
Fonte: (ALBERTS, 2011)

2.2 Expressão Gênica

O genoma propriamente dito não realiza nenhum tipo de função ativa no organismo. Deste modo, para a manutenção dos mecanismos biológicos vitais é necessário que os genes sejam expressos, ou seja, traduzidos em um produto biológico. Conforme pode ser observado na Figura 2.4, a expressão de um gene ocorre quando o mesmo é transcrito em RNA que, posteriormente, é traduzido em um produto biológico com alguma função específica para o organismo, como por exemplo, uma proteína (COX, 2012).

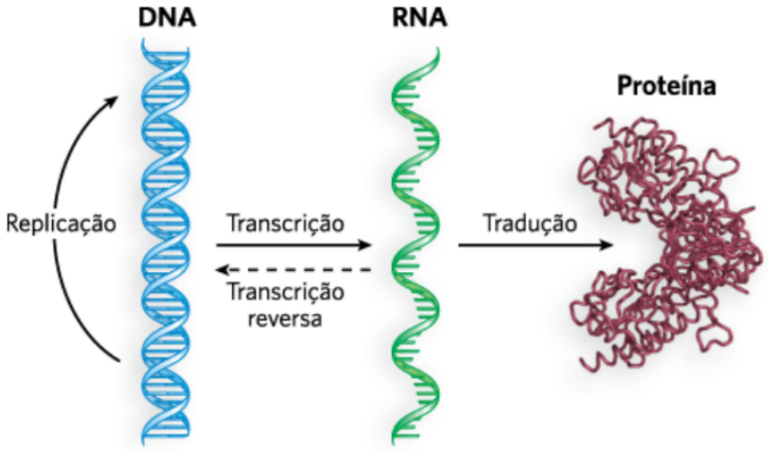


Figura 2.4: Transcrição do DNA.
Fonte: (COX, 2012)

A transcrição é o processo através do qual o RNA é sintetizado, ou seja, uma parte da dupla hélice de DNA é transcrita em RNA e posteriormente transformada em proteínas. Deste modo, a transcrição possui uma posição de destaque na expressão gênica, tratando-se do primeiro passo da transformação do DNA em proteínas (ZAHA, 2014). Durante a transcrição, também ocorre a regulação da expressão gênica. A regulação é feita através das regiões promotoras que definem quais as partes do DNA que serão sintetizados e qual a quantidade de proteína que será produzida (ZAHA, 2014). Desta forma, a transcrição do RNA é iniciada quando a enzima RNA polimerase reconhece uma região promotora (MADIGAN, 2011).

2.3 Regiões Promotoras

A composição do genoma se dá, principalmente, por dois tipos de regiões, que são as regiões gênicas e intergênicas. As primeiras também são conhecidas como regiões codificadoras de proteínas e se tratam de regiões que possuem as informações necessárias para sintetizar as proteínas ou o RNA estável. Já as regiões intergênicas, conhecidas como não-codificantes, são a maior parte do genoma e não são capazes de codificar proteínas, porém possuem funções importantes. De fato, são essas regiões do DNA que são responsáveis por regular a expressão dos genes adjacentes. Ou seja, é através destas regiões, que as células controlam quando e onde um gene será ativado (ALBERTS, 2011).

As regiões específicas no DNA, que determinam o início da transcrição, são chamadas de promotores. Para iniciar a síntese de RNA, se faz necessário que a RNA polimerase reconheça uma região promotora através do fator sigma (Figura 2.5). Após a síntese de um pequeno segmento de RNA, o fator sigma é dissociado (MADIGAN, 2011). Os números que nomeiam cada fator sigma referem-se ao tamanho da proteína em quilodáltons[1], sendo que os valores mais comuns são: 24, 28, 32, 38, 54 e 70. Estes fatores sigmas correspondem a diferentes reações da célula em relação ao meio, como por exemplo, o fator sigma 32 corresponde ao choque térmico e o fator sigma 70 ao crescimento normal da célula (MADIGAN, 2011).

[1]Unidade de Massa Atômica, 1 quilodálton equivale a 1,66 x 10^{-21} gramas.

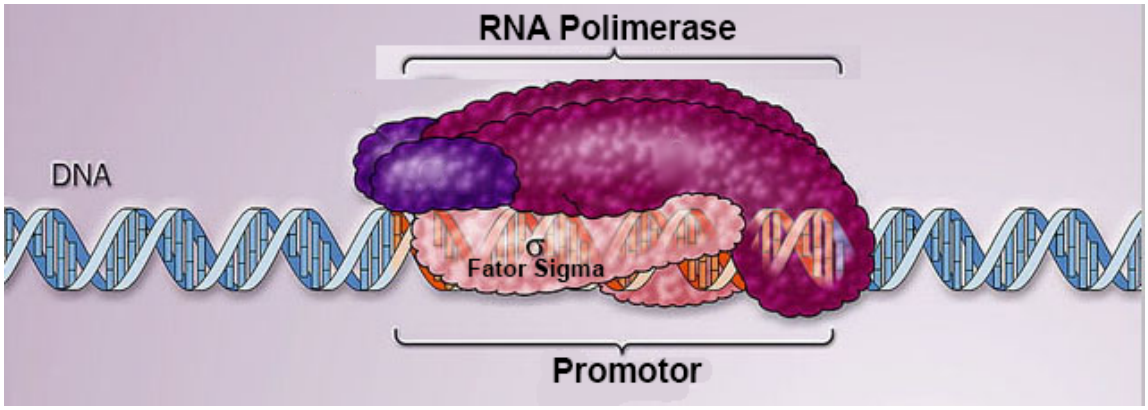


Figura 2.5: Identificação da região promotora.
Fonte: (LIMA, 2017)

A finalização da transcrição do RNA ocorre por sequências de bases específicas presentes no DNA que são denominados de terminadores transcricionais. Após o reconhecimento do terminador transcricional, o RNA recém sintetizado separa-se do DNA e a fita de DNA que estava aberta para permitir a transcrição volta a fechar-se retornando à forma de dupla-hélice (MADIGAN, 2011).

2.4 Bioinformática

O avanço das técnicas de biologia molecular tem sido cada vez maior, sendo que a junção desta área com as ciências exatas causou uma nova área de conhecimento que é chamada de bioinformática. A bioinformática faz a junção de algoritmos matemáticos, da física e da estatística, e tem como principal objetivo utilizar técnicas dessas áreas de forma a obter informações com valor biológico (LIPAY, 2015).

Um dos maiores objetos de estudo da bioinformática é a genômica, que é a área que estuda o mapeamento, o sequenciamento, a análise e a comparação de genomas (MADIGAN, 2011). O primeiro caso de genoma sequenciado ocorreu em 1976 e foi o genoma de RNA do vírus MS2. No ano seguinte, foi sequenciado o primeiro genoma de DNA, do vírus X174 (MADIGAN, 2011).

O sequenciamento de genomas traz informações valiosas para a biologia, porém é apenas uma parte do que é necessário para o entendimento das implicações biológicas de cada parte de um genoma. De fato, para um completo entendimento do funcionamento das funções gênicas, é necessário localizar e determinar a função de cada uma das partes de um genoma. O processo de análise e interpretação do genoma para extração do significado biológico é chamado de anotação genômica (STEIN, 2001).

A bioinformática é muito utilizada em estudos que envolvam a anotação genômica, devido principalmente à grande quantidade de dados encontrados em um genoma. Um exemplo do uso da bioinformática são os bancos de dados de genes, que podem ser utilizados para a predição da função de um gene em um outro genoma, sem que seja necessário uma análise biológica (STEIN, 2001). Outros exemplos que podem ser citados, são os algoritmos de aprendizado de máquina, como as RNAs, que são utilizados para a identificação das estruturas que compõem um genoma (STEIN, 2001).

2.5 BACPP

O BACPP (SILVA, 2011) é um software *web* que faz uso de algoritmos de aprendizado de máquina para a predição de promotores bacterianos. Diferente dos softwares comumente utilizados, como por exemplo o NNPP (PROJECT, 2017), o BACPP não é limitado exclusivamente ao uso do valor de sigma 70. De fato, esse permite a utilização dos valores de sigma 24, 28, 32, 38, 54 e 70. Na Figura 2.6 podemos ver a página inicial do BACPP.

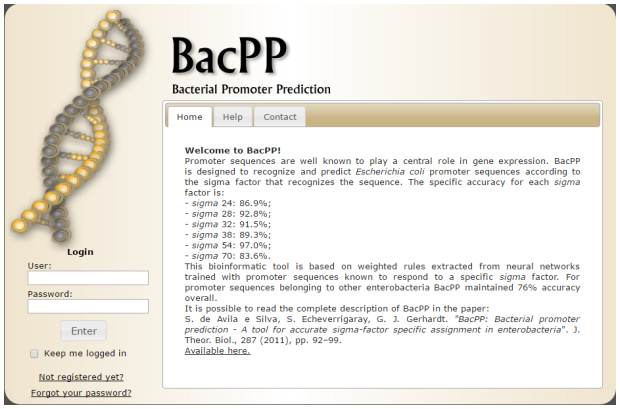


Figura 2.6: Pagina inicial ferramenta BACPP
Fonte: (SILVA, 2016)

O BACPP foi desenvolvido na tese de doutorado de Sheila de Ávila e Silva e tem como objetivo a aplicação de Redes Neurais Artificiais no reconhecimento de regiões promotoras em bactérias Gram-negativas (SILVA, 2011). O BACPP é baseado no uso das regras derivadas do processo de aprendizado de RNAs, sendo que os valores obtidos através da RNA, são ponderados de forma a maximizar a predição dos promotores e classificá-los de acordo com o fator sigma que foi reconhecido (SILVA, 2011). Os valores de ponderação (ver Tabela 2.1) utilizados pelo BACPP foram selecionados empiricamente, após testes com valores inteiros entre -10 e +10, e são iguais para todos os valores de sigma reconhecidos (SILVA, 2011).

Tabela 2.1: Valores de ponderação de acordo com o escore obtido através da regra da rede neural

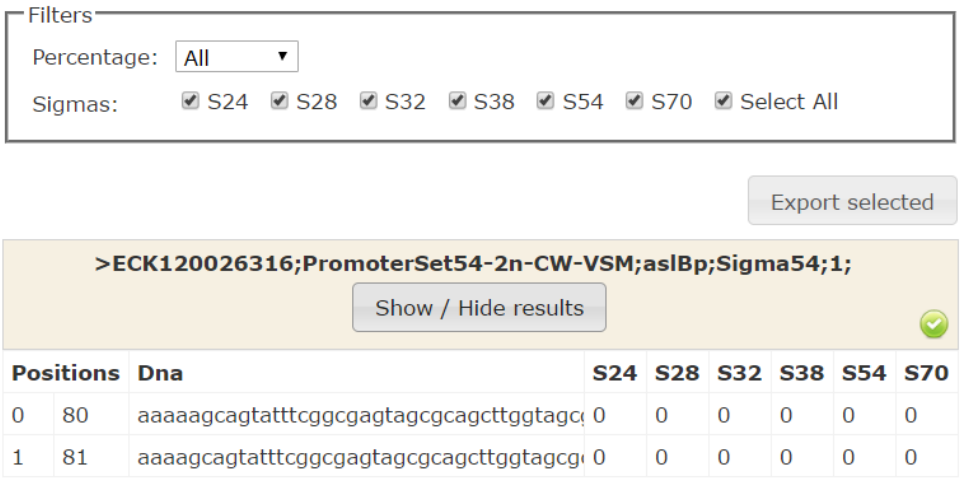
Escore obtido da regra da Rede neural	Valor de ponderação
Maior que 0.6	+6
0.5 – 0.59	+4
0.4 – 0.49	+2
0.3 – 0.39	+1
0.2 – 0.29	0
0.1 – 0.19	-1
Menor que 0.1	-3

Fonte: (SILVA, 2011)

O BACPP apresenta como entrada, arquivos de sequências genéticas no formato FASTA (SILVA, 2016). Este formato é oriundo do programa FastA, que verifica a similaridade entre sequências de DNA (VIRGINIA, 2017). A primeira linha de um arquivo FASTA (Figura 2.7) traz o cabeçalho, que é iniciado pelo caractere »", e traz a identificação da sequência e comentários. Após a linha de cabeçalho, tem-se a sequência, onde cada linha deve possuir ao menos 80 caracteres que representam cada um dos ácidos nucleicos (Adenina (A), Tinina (T), Citosina (C) e Guanina (G)).

Figura 2.7: Dados no formato FASTA
Fonte: (DENMARK, 2017)

Após o usuário deve selecionar quais os fatores sigma que serão considerados na análise, sendo que ao final da análise tem-se um arquivo com a identificação do arquivo de entrada, a posição do nucleotídeo inicial da análise, a posição do nucleotídeo final da análise e a probabilidade da sequência ser ou não um promotor. Além disso, a saída poderá ser apresentada em tela (Figura 2.8), onde tem-se o cabeçalho de identificação do arquivo FASTA, além de uma tabela com as posições analisadas, o DNA que foi analisado e o resultado da análise em cada um dos fatores sigmas selecionados.



Positions		Dna	S24	S28	S32	S38	S54	S70
0	80	aaaaagcagtatttcggcgagtagcgcagcttggtagc	0	0	0	0	0	0
1	81	aaaagcagtatttcggcgagtagcgcagcttggtagcg	0	0	0	0	0	0

Figura 2.8: Saída de dados do Software BACPP
Fonte: (SILVA, 2016)

Na Figura 2.9 tem-se a arquitetura do BACPP. Como pode ser observado, a arquitetura do BACPP é dividida em duas partes, que são a parte funcional e a parte administrativa. Na parte funcional, é permitido ao usuário efetuar a análise das sequências gênicas, bem como visualizar os resultados. A parte administrativa, é o local onde o usuário (com permissões de administrador) pode efetuar as alterações e atualizações do sistema, além da gerência de usuários.

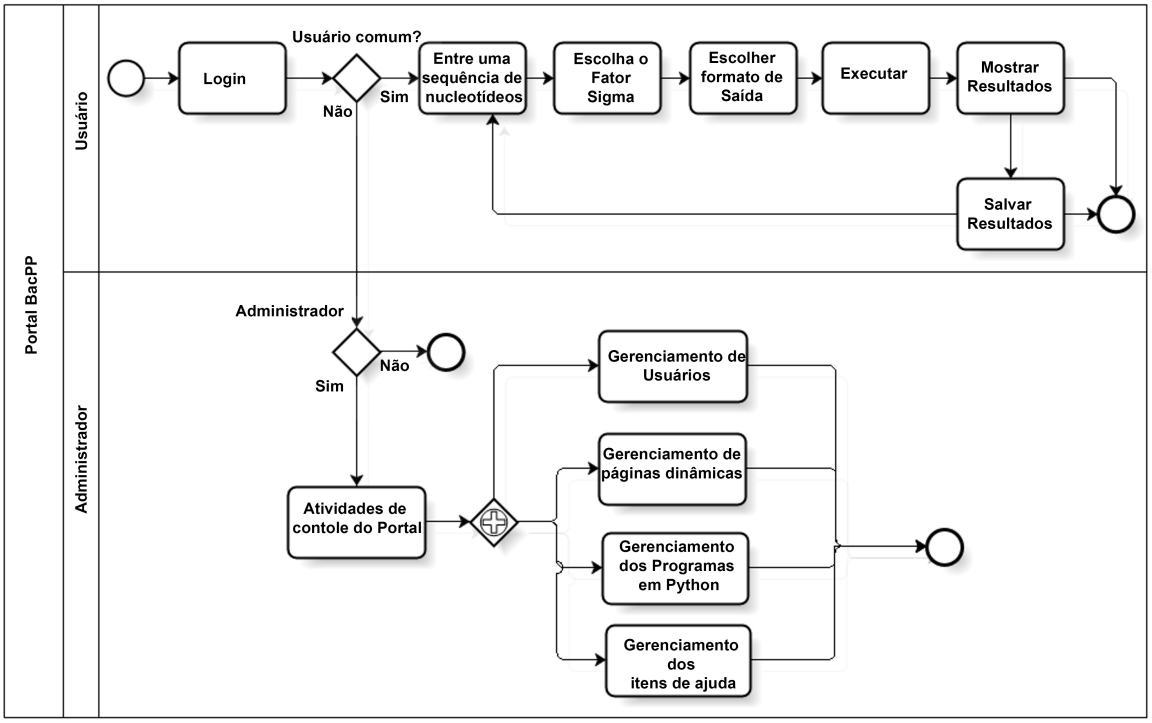


Figura 2.9: Estrutura Software BACPP
Fonte: (SILVA, 2016)

3 ALGORITMOS GENÉTICOS

A busca por algoritmos inspirados nos seres vivos para a resolução de problemas de otimização iniciou-se na década de 1960. Um termo comum para referir-se a estas técnicas de minimização é a Computação Evolutiva (CE) (GOLDBERG, 1989). A CE engloba um conjunto de técnicas de busca e otimização baseados na evolução natural das espécies. Ou seja, é criada uma população de indivíduos que irão se reproduzir e competir pela sobrevivência, sendo que os indivíduos mais aptos sobrevivem e transferem seus atributos às novas gerações. Dentre os algoritmos desta classe, destacam-se os Algoritmos Genéticos, que foram introduzidos por John Holland e sua equipe em 1975 (HOLLAND, 1992). Esses são muito conhecidos devido ao seu potencial e sua ampla aplicação em técnicas de busca e otimização (GOLDBERG, 1989).

3.1 Problemas de Otimização

Muitas vezes é necessário descobrir o maior ou menor valor para a solução de um problema. Como por exemplo, os engenheiros automotivos querem construir um automóvel que utilize a menor quantidade de combustível possível e os cientistas querem identificar o comprimento de onda que carrega a maior radiação possível em uma determinada temperatura. Estes problemas pertencem a um campo da matemática chamado de otimização (HUGHES-HALLET, 2003).

Em casos onde é possível definir um modelo matemático representativo do sistema, pode-se aplicar as técnicas de otimização de forma a obter os valores mínimos ou máximos de uma função previamente definida. Os valores mínimos e máximos são divididos em dois tipos, que são: os locais e os globais. O menor (ou maior) valor de uma função em um determinado domínio é conhecido como mínimo (ou máximo) global. Os mínimos e máximos globais também são conhecidos como valores extremos ou valores ótimos. Além dos mínimos e máximos globais, existem os mínimos e máximos locais, que correspondem aos menores ou maiores valores de uma função em um determinado intervalo, sendo que mínimos e máximos locais podem coincidir ou não com os mínimos ou máximos globais. Na Figura 3.1, tem-se exemplos de mínimos e máximos globais e locais (HUGHES-HALLET, 2003).

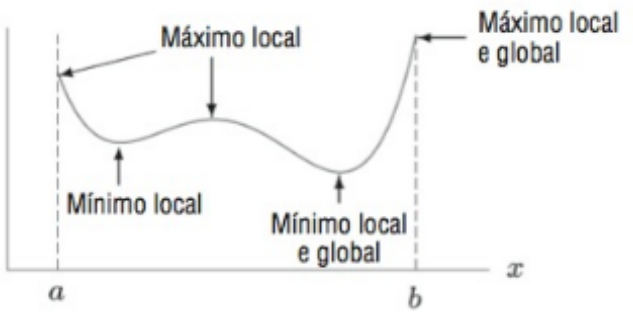


Figura 3.1: Máximos e mínimos globais e locais
Fonte: (HUGHES-HALLET, 2003)

3.2 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são técnicas que buscam encontrar a solução para problemas de busca e otimização. Estes são métodos iterativos que combinam a sobrevivência do melhor indivíduo e a troca de informações para formar um algoritmo de busca baseado na evolução dos seres vivos. De fato, os AGs seguem os princípios da seleção natural, onde quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será a sua chance de sobreviver e gerar novos descendentes (RICH, 2009).

Os AGs são algoritmos iterativos que procuram reproduzir de modo simplificado, a evolução das espécies, tratando as possiveis soluções do problema como "indivíduos" de uma "população". Deste modo, esta população irá "evoluir" a cada iteração, que é tratada como uma "geração".

Para (COPPIN, 2010), um AG pode ser descrito em 5 passos, que são apresentados na Figura 3.2. Primeiramente, é gerada uma população aleatória com as possíveis soluções. A aptidão de cada indivíduo é avaliada através de uma função de custo, sendo que os mais aptos são selecionados e então sofrem operações genéticas para a geração de novas soluções. Caso as novas soluções atinjam o critério de parada definido, o melhor indivíduo é retornado. Caso contrário, o ciclo se repete até que o critério de parada seja atingido.

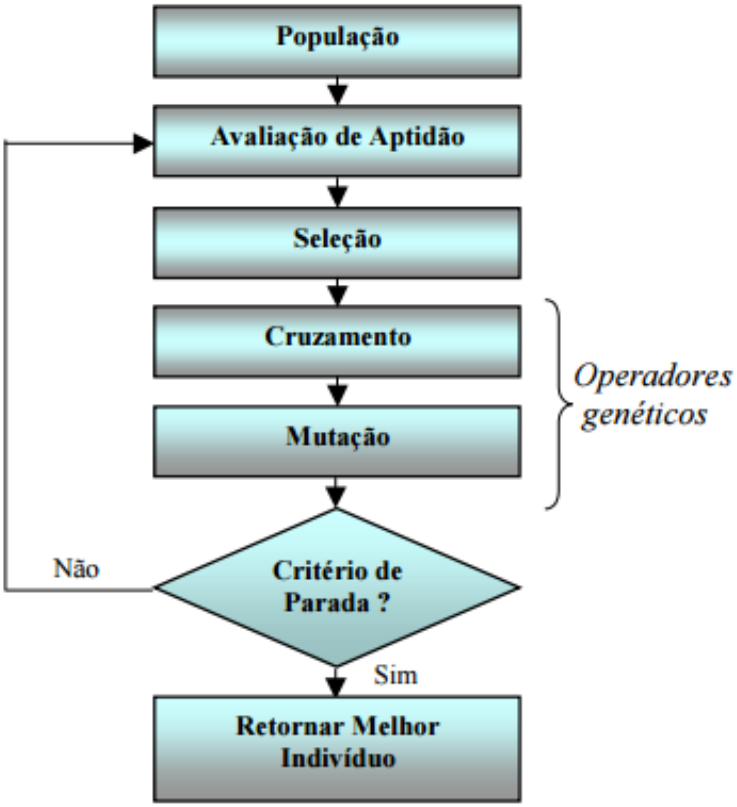


Figura 3.2: Estrutura Básica de Algoritmo Genético
Fonte: (POZO et al., 2017)

3.2.1 Geração dos Indivíduos

Dentro de um AG, cada indivíduo representa uma solução do problema proposto. A cada nova geração, estes indivíduos são cruzados ou sofrem mutações, de forma a criar novas soluções. Os indivíduos são formados por palavras genéticas (genótipo), sendo que cada grupo de caracteres (genes) que compõem essa palavra representa um atributo (cromossomo). A tradução deste genótipo para valores reais é conhecida como fenótipo (MITCHELL, 1998).

Um dos principais modos de representação de um indivíduo é como uma concatenação de *bits*. Nesta representação, cada *bit* representa um gene, o conjunto destes genes representa um cromossomo do indivíduo e o conjunto destes cromossomos representa um indivíduo. Existem outras variações de codificações binárias, além de diversas outras formas possíveis para representação de indivíduos em AG (HOLLAND, 1992). Na Figura 3.3 temos o exemplo de um indivíduo hipotético. Neste exemplo, é representado cada um dos genes, como os mesmos formam um cromossomo e o genótipo formado pela junção destes cromossomos, além da tradução do genótipo para valores reais, ou fenótipo.

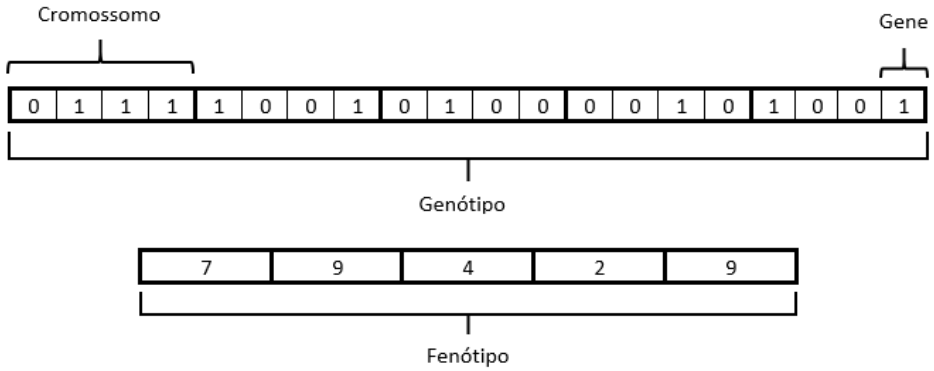


Figura 3.3: Representação de um indivíduo

3.2.2 Criação da População

A população de um AG corresponde a um conjunto de indivíduos que representam possíveis soluções e que serão usados para criar um novo conjunto de indivíduos. O tamanho desta população pode variar de acordo com o problema explorado, porém é necessário um certo cuidado com o tamanho da população, pois populações pequenas dificultam uma diversidade que possibilite uma convergência para a solução do problema, enquanto grandes populações, necessitam de um alto poder de processamento (MITCHELL, 1998).

A população inicial de um AG deve ser criada aleatoriamente, sendo que a partir dessa população inicial novas soluções serão desenvolvidas. Deste modo, é importante que a população inicial possua uma diversividade suficiente de modo a permitir que sejam geradas novas soluções que possam cobrir todo o domínio do problema.

Uma população pode ser construída de diversos modos, sendo que o modo mais simples é a utilização de um vetor de indivíduos, conforme pode ser visto na Figura 3.4. Nela possuímos a população de um Algoritmo Genético hipotético, onde cada indivíduo dentro desta população possui um número de identificação, um genótipo, um fenótipo e o resultado de uma função custo, ou seja, a função que será utilizada para a avaliação da aptidão do mesmo.

Identificador	Dados		
	Genótipo	Fenótipo	F(X)
1	01111001010000101001	79429	81
2	00100101011010100001	25701	92
3	01100111010010110101	67515	130
4	01000100011010100011	44703	78
5	01010101011110110111	55817	49
n	00010011010010000001	13481	60

Figura 3.4: Representação de uma população hipotética

3.2.3 Avaliação de Aptidão

A Avaliação de Aptidão (AP) é realizada a partir de uma função custo e determina a aptidão de cada indivíduo dentro de uma população. Se comparada com a teoria da evolução de soluções de Darwin, a função custo permite verificar quão adaptado um indivíduo é ao seu habitat (GEN, 2000). Esta avaliação é imprescindível para o correto funcionamento de um algoritmo genético. É através desta função que se mede quão próximo um indivíduo está da solução desejada (POZO et al., 2017).

A função custo deve ser a que melhor representa o problema, permitindo diferenciar as boas e as más soluções em uma proporção correta. De fato, se a precisão da avaliação for baixa, pode ocorrer o descarte de uma solução boa, além de um processamento desnecessário em soluções pouco promissoras.

3.2.4 Seleção dos Indivíduos

Dentro dos AGs, o processo de seleção é encarregado de estabelecer quais indivíduos irão sobreviver, para posteriormente, serem utilizados em uma nova população. Essa seleção é baseada no valor de aptidão, sendo que indivíduos com uma aptidão maior, possuem maiores chances de sobreviver. Existem diversos métodos para seleção dos indivíduos, entra os quais, destacam-se os métodos de: Seleção por Roleta, Seleção por Torneio e Elitismo (MITCHELL, 1998).

O método de seleção por roleta foi proposto por (GOLDBERG, 1989), onde cada indivíduo de uma população possui um espaço em uma roleta. Este espaço deve ser proporcional à sua avaliação de aptidão, ou seja, quanto maior a probabilidade de sobrevivência, maior será esse espaço. Deste modo, indivíduos com maior aptidão possuem uma chance maior de serem escolhidos. Cada vez que a roleta é girada, um novo indivíduo é escolhido para a população. Este processo deve ser repetido até que a população esteja completa. A Figura 3.5 ilustra um exemplo de roleta utilizada para a seleção de indivíduos. Apesar de os melhores indivíduos possuírem uma maior probabilidade de escolha, este método não possui garantias que os melhores indivíduos serão selecionados, podendo até mesmo selecionar apenas os piores resultados dentro de uma população.

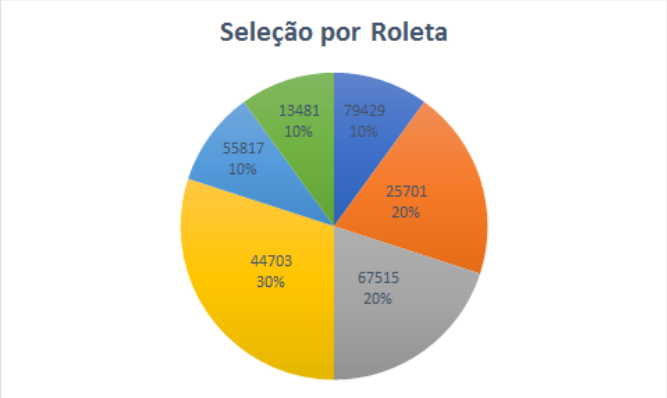


Figura 3.5: Representação de uma seleção por roleta hipotética

A Seleção por Torneio é um método onde N indivíduos são selecionados na população através de uma roleta. Então, dentre os N selecionados, o melhor indivíduo é adicionado à nova população. O processo é então repetido até que a nova população esteja completa. A Seleção por Torneio garante que o pior indivíduo será descartado da população, uma vez que esse nunca será o indivíduo escolhido.

O elitismo é um operador adicional presente em quase todas as implementações de AG. Neste os N melhores indivíduos são adicionados diretamente na nova geração (MITCHELL, 1998). Deste modo, os melhores indivíduos não são perdidos ou destruídos pelos operadores genéticos. Assim, o elitismo aumenta significativamente a performance dos AGs.

3.2.5 Operadores Genéticos

Após a seleção de uma população, é necessário evoluir esta população para a obtenção de novas soluções. A evolução dos AGs ocorre a partir de três operadores, que são: a reprodução, o cruzamento e a mutação.

A reprodução é a cópia de um indivíduo para a próxima geração, sem que ocorra nenhum tipo de mudança no mesmo. Já o cruzamento *(crossover)*, é o principal operador genético de um AG (MITCHELL, 1998) e é análogo à reprodução sexuada nos seres vivos. Neste, dois indivíduos são selecionados e recombinados de forma a gerar dois novos indivíduos com características de ambos os pais. Nos casos de codificação binária, os indivíduos podem ser gerados pela combinação dos *bits* dos pais.

O cruzamento de um ponto é a forma mais simples de cruzamento, onde um ponto aleatório n é escolhido e os atributos dos pais após este ponto são trocados. Na Figura 3.6 tem-se um exemplo de cruzamento de um ponto entre dois indivíduos hipotéticos. Como pode ser observado, o ponto do cruzamento encontra-se no quarto gene, sendo que todos genes após este ponto são trocados entre os pais. Este tipo de cruzamento possui limitações uma vez que não permite todas as combinações possíveis entre os genes dos pais.

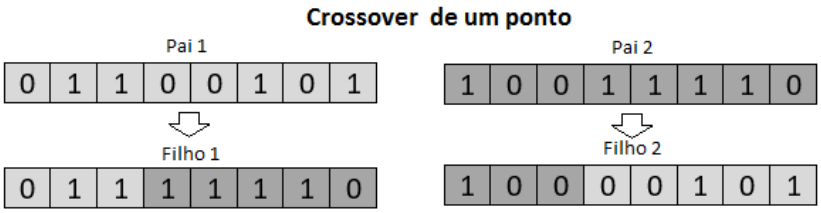


Figura 3.6: Cruzamento de um ponto

De forma a reduzir essa limitação, tem-se o cruzamento de dois pontos, onde dois pontos aleatórios são selecionados e os genes entre estes pontos são trocados. Na Figura 3.7, tem-se um exemplo de um cruzamento de dois pontos entre dois indivíduos hipotéticos. Neste exemplo, os pontos de cruzamento se encontram no quarto e no quinto gene, sendo que apenas os genes quatro e cinco são trocados para a geração dos filhos.

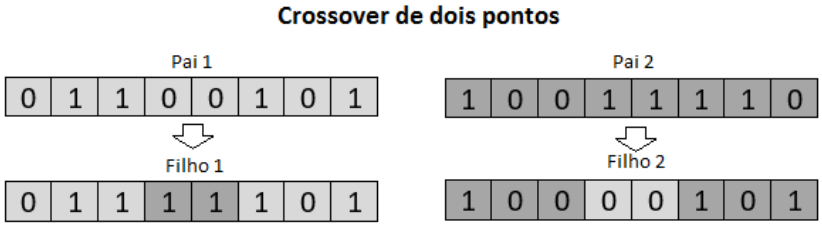


Figura 3.7: Cruzamento de dois pontos

Além do cruzamento de um ponto e de dois pontos, existem diversos outros modos de cruzamentos, sendo que esses devem ser escolhidos de acordo com o AG que será utilizado. Mais exemplos de cruzamentos podem ser encontrados em (GOLDBERG, 1989).

O operador da mutação é um operador que tem como objetivo percorrer regiões que ainda não foram exploradas pelo AG. A operação de mutação é importante para garantir a diversidade da população através das gerações. Na Figura 3.8 tem-se um exemplo de mutação, onde um indivíduo hipotético sofre uma mutação. De fato, o indivíduo tem o quarto gene alterado antes de ser adicionado à nova população.

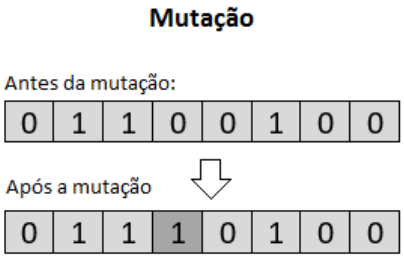


Figura 3.8: Mutação de indivíduo

3.2.6 Gerações de um Algoritmos Genéticos

Em um AG, cada geração corresponde a uma nova iteração, onde um novo conjunto de indivíduos são gerados a partir da evolução da geração anterior através do uso dos operadores genéticos. É através da criação de sucessivas gerações, que são obtidos os indivíduos mais aptos em um AG.

A Figura 3.9 representa a criação de uma nova geração hipotética dentro de um AG. Os indivíduos 1 e 2 da nova geração, foram obtidos através do cruzamento de um ponto, iniciado a partir do décimo primeiro gene, dos indivíduos 1 e 2 da geração anterior. Já o indivíduo 3, por ser o indivíduo com maior aptidão, foi reproduzido na nova geração através do elitismo. O indivíduo 4 da geração anterior sofreu uma mutação no seu décimo segundo gene ao ser adicionado à nova geração. Os indivíduos 5 e 6, sofreram um cruzamento de dois pontos, entre o décimo e décimo terceiro genes, gerando assim os indivíduos 5 e 6 da nova geração.

Criação de uma nova Geração

Identificador	Dados		
	Genótipo	Fenótipo	F (X)
1	01111001010000101001	79429	81
2	00100101011010100001	25701	92
3	01100111010010110101	67515	130
4	01000100011010100011	44703	78
5	01010101100000010111	55817	49
6	00010011010010000001	13481	60

Mutação
Crossover
Reprodução

Identificador	Dados		
	Genótipo	Fenótipo	F(X)
1	01111001011010100001	79701	74
2	00100101010000101001	25429	95
3	01100111010010110101	67515	130
4	01000100011110100011	44803	73
5	01010101110010010111	56297	55
6	00010011000000000001	13001	48

Figura 3.9: Criação de uma nova geração em um AG

4 BIBLIOTECAS PARA ALGORITMOS GENÉTICOS

O objetivo principal deste trabalho consistiu no aumento da assertividade na predição de promotores no BACPP através da aplicação de técnicas de minimização com o uso de AGs. O desenvolvimento de uma solução de minimização baseada em AGs exige a implementação de diversas estruturas específicas como, por exemplo, os operadores genéticos. Entretanto, existem diversas ferramentas que facilitam e auxiliam a implementação destas estruturas. Neste capítulo são apresentadas algumas das ferramentas disponíveis para esta função, bem como uma breve descrição dessas. Ao final será apresentada a ferramenta que foi utilizada para o desenvolvimento deste trabalho.

4.1 *Genetic Algorithms Toolbox*

O *Genetic Algorithms Toolbox (GA Toolbox)* é um módulo disponível para o software *MATLAB* e que disponibiliza rotinas para implementação de AG e outras técnicas de computação evolutiva (SHEFFIELD, 2017). Esse módulo caracteriza-se pelo uso das funções matriciais presentes no *MATLAB* e o suporte à representações binárias, para representação dos indivíduos e implementação das operações genéticas. O GA Toolbox trabalha com uma grande diversidade de operadores genéticos e é capaz de suportar múltiplas sub-populações. O GA Toolbox é um software de código livre, sendo que o código-fonte e a documentação do mesmo podem ser obtidos em `http://codem.group.shef.ac.uk/index.php/ga-toolbox` (SHEFFIELD, 2017).

4.2 GAlib

A *GAlib* é uma biblioteca desenvolvida na linguagem de programação *C++* e que possui um conjunto de objetos para a implementação de algoritmos genéticos (WALL, 2017). A *GAlib* suporta diferentes representações de cromossomos, como *bit-string*, *array*, lista, árvore, ou ainda, pode-se derivar um cromossomo baseado em um objeto da *GAlib*. Além disso, essa biblioteca suporta diversas variações de operadores genéticos e métodos de seleção. O *GAlib* é um software de código aberto que pode ser executado nas plataformas UNIX, MAC e MS-DOS. O código-fonte e a documentação do *GAlib* podem ser obtidos em `http://lancet.mit.edu/ga` (WALL, 2017).

4.3 *Distributed Evolutionary Algorithms in Python*

O *Distributed Evolutionary Algorithms in Python (DEAP)* é um *framework* de Computação Evolutiva desenvolvido na linguagem *Python*. Esse *framework* permite o desenvolvimento de AGs utilizando diferentes estruturas para a representação de cromossomos como, por exemplo, *strings*, inteiros e *arrays*. Além disso, o DEAP disponibiliza recursos para a personalização dos indivíduos e dos operadores genéticos de um AG. Por fim, o DEAP também permite o uso de recursos de paralelismo (LAVAL, 2017).O código-fonte e a documentação do DEAP podem ser obtidos no endereço `https://github.com/DEAP/deap` (LAVAL, 2017). Esse é um software de código aberto, que pode ser executado nas plataformas UNIX, MS-DOS e MAC.

4.4 Evolver

O *Evolver* é um software comercial desenvolvido nas linguagens de programação *C/C++* para a resolução de problemas de minimização através de algoritmos genéticos. Esse é um *add-in* de otimização desenvolvido para o software *Microsoft Excel* (PALISADE, 2017).

O *Evolver* é um software proprietário, sendo que o *download* de uma versão de testes, documentação e informações podem ser obtidos no site oficial, que pode ser acessado em `http://www.palisade.com/evolver` (PALISADE, 2017).

4.5 *Python Strongly Typed gEnetic Programming*

A *pySTEP* é uma biblioteca de AG desenvolvida na linguagem de programação *Python*. Essa biblioteca permite a definição de regras para a evolução dos indivíduos, (KHOURY, 2017). A biblioteca é de código aberto, sendo que essa pode ser obtida, juntamente à documentação, em `http://pystep.sourceforge.net` (KHOURY, 2017).

4.6 *Genetic Algorithm Utility Library*

O *Genetic Algorithm Utility Library (GAUL)* é uma biblioteca de computação evolutiva desenvolvida nas linguagens de programação *C/C++* e projetada para auxiliar no desenvolvimento de aplicações que utilizem algoritmos genéticos e evolucionários. De fato, esta biblioteca disponibiliza estruturas para a representação de indivíduos, populações,etc., além de suportar o processamento paralelo de AGs (ADCOCK, 2017). O *GAUL* é uma biblioteca de código aberto, sendo que a sua documentação e o código-fonte podem ser obtidos em `http://gaul.sourceforge.net` (ADCOCK, 2017).

4.7 *Genetic Algorithm Java Implementation Toolkit*

A *Genetic Algorithm Java Implementation Toolkit (GAJIT)* é uma biblioteca para implementação de algoritmos genéticos baseada na biblioteca GAGS e desenvolvida na linguagem de programação *Java* (MICROPRAXIS, 2017). Na GAJIT, os indivíduos são representados por *strings* de *bits*, com tamanho variável. Além disso, ela suporta nove tipos de operadores genéticos e a população é representada como uma lista. A GAJIT é uma biblioteca de código aberto, sendo que o código-fonte, com a documentação inclusa, pode ser obtido em `http://www.micropraxis.com/gajit` (MICROPRAXIS, 2017).

4.8 Pyevolve

O *Pyevolve* é um *framework* completo de AG, desenvolvido na linguagem *Python*. Esse *framework* possui estruturas padrão para facilitar a criação de um AG, permitindo ainda a personalização de representações e operadores genéticos. Além disso, o usuário pode monitorar, através de gráficos e estatísticas, a evolução das gerações de um AG. O *Pyevolve* é um *framework* de código aberto, sendo que a sua documentação e código-fonte podem ser obtidos em `http://pyevolve.sourceforge.net` (PERONE, 2017).

4.9 *Evolving Objects*

O *Evolving Objects (EO)* é um *framework* de Computação Evolutiva, implementado na linguagem C++, que permite a criação de algoritmos específicos para a solução de problemas de otimização (GENEURA, 2017). Dentre os métodos disponíveis destacam-se os AGs, sendo que a EO disponibiliza diversas estruturas para a representação dos indivíduos, além de permitir a personalização destes. O *EO* permite ainda o uso de recursos de computação paralela. O EO é um *framework* de código aberto, sendo que o *download* do seu código fonte e da sua documentação pode ser efetuado no site oficial, que pode ser acessado em `http://eodev.sourceforge.net`. (GENEURA, 2017).

4.10 Definição da biblioteca de AG

Para o desenvolvimento deste trabalho optou-se pela utilização de uma biblioteca que implementasse as estruturas e os métodos necessários para o desenvolvimento de uma solução baseada em AGs. Para a definição dessa biblioteca, os seguintes critérios foram utilizados:

- Linguagem: deverá ser implementada na linguagem *Python*, de forma a facilitar a integração com o BACPP, que foi implementado nesta linguagem.
- Licença: deverá ser de código aberto, permitindo assim a realização de alterações no código fonte.
- Plataforma: deverá ser compatível com a plataforma UNIX (*Linux*).

Na Tabela 4.1, tem-se um comparativo entre as bibliotecas e ferramentas que foram descritas neste capítulo. Como pode ser observado somente as bibliotecas DEAP, PYSTEP e pyEVOLVE possuem suporte para a linguagem *Python*. Dentre estas 3 optou-se pela biblioteca *DEAP*, uma vez que essa é a mais utilizada e citada na literatura. Alguns exemplos de projetos que utilizam o DEAP são: *Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science* (OLSON et al., 2016), *Construction cost and energy performance of single family houses: From integrated design to automated optimization*(CHARDON et al., 2016) e *Automating biomedical data science through tree-based pipeline optimization*(ANDREWS et al., 2016).

Tabela 4.1: Seleção da ferramenta a ser utilizada

Ferramenta	**Linguagem Python**	**Software Livre**	**Plataforma Unix**
GAT	Não	Sim	Não
GAlib	Não	Sim	Sim
DEAP	Sim	Sim	Sim
Evolver	Não	Não	Não
pySTEP	Sim	Sim	Sim
GAUL	Não	Sim	Sim
GAJIT	Não	Sim	Sim
Pyevolve	Sim	Sim	Sim
EO	Não	Sim	Sim

5 IMPLEMENTAÇÃO E RESULTADOS OBTIDOS

Na implementação de um algoritmo genético algumas coisas devem ser consideradas, sendo que dentre essas destacam-se: a forma como os indivíduos serão representados; a função custo que será utilizada para a classificação dos indivíduos; e os tipos de cruzamento, mutação e seleção utilizados. Além disto, existem ainda outros parâmetros a serem definidos como, por exemplo, o tamanho da população e os critérios de convergência do método.

Neste capítulo serão abordados os aspectos relacionados à implementação, bem como o detalhamento dos parâmetros utilizados. A linguagem de programação utilizada para o desenvolvimento deste trabalho foi a linguagem *Python*, utilizando a biblioteca *DEAP*, sendo executada de forma paralela entre os núcleos do processador, através do uso da biblioteca *SCOOP (Scalable COncurrent Operations in Python)* (HOLD-GEOFFROY; GAGNON; PARIZEAU, 2014). O código-fonte se encontra no CD em anexo.

5.0.1 Definição de Indivíduos

Primeiramente foi defininida a codificação dos indivíduos, onde o genótipo do indivíduo é composto por uma sequência de sete cromossomos, referentes a cada um dos valores de ponderação no BACPP. Cada um destes atributo é representado por uma variável do tipo *float*, com valores entre -10 e 10. Na Figura 5.1 podemos ver a representação de um indivíduo aleatório. Conforme (HERRERA; LOZANO; VERDEGAY, 1998), este tipo de codificação apresenta vantagens quando usada em problemas de otimização com variáveis sobre o domínio contínuo, ou sejam em domínios onde ocorre uma dificuldade de representar os indivíduos através de valores binários.

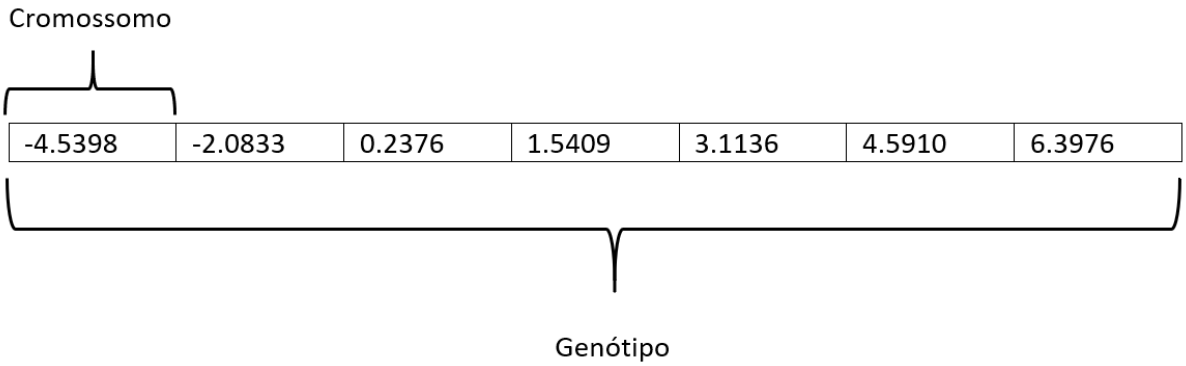


Figura 5.1: Representação de Indivíduo

A população foi definida incialmente com 200 indivíduos, que foram gerados a partir de valores aleatórios. Durante o desenvolvimento deste trabalho foram realizados testes com diferentes tamanhos de população, mais especificamente foram realizados testes com 200, 500 e 1000 indivíduos. Nestes testes verificou-se que o *fitness* [1] médio não sofreu variações significativas com o aumento da população. Na Tabela 5.1, tem-se a avaliação do *fitness médio* até a 100ª geração para populações de 200;500 e 1000 indivíduos. Conforme pode ser observado, o aumento da população não provocou melhorias significativas no *fitness médio*. Desta forma, o tamanho da população foi mantida com 200 indivíduos pelo fato de apresentar um custo computacional inferior.

Tabela 5.1: População e fitness médio

Tamanho da População	**200**	**500**	**1000**
1ª Geração	49.06	49.43	49.60
5ª Geração	39.23	40.29	40.20
10ª Geração	35.69	35.90	37.44
50ª Geração	34.58	35.33	35.61
100ª Geração	35.28	35.41	35.31

5.1 Função Custo

Depois de formada a população inicial ou ao final de cada geração é necessário avaliar o grau de aptidão, ou *fitness*, de cada indivíduo. Esta avaliação é feita através de uma função, que é chamada de função custo. A função custo é extremamente importante para o bom funcionamento do algoritmo, pois é através dela que será qualificada cada solução.

Neste trabalho, a função custo consistiu em substituir os valores de ponderação

[1] Fitness representa o grau de aptidão do indivíduo

do BACPP (SILVA, 2011) pelos valores do indivíduo avaliado, sendo que o BACPP retorna a probabilidade de uma sequência ser uma região promotora. Sabendo-se previamente quais regiões são promotoras, é possível calcular o percentual de acerto para cada indivíduo. Desta forma, primeiramente, a função custo é avaliada para todas as sequênciass que são promotoras, resultando em um percentual de acerto (PV). Após, a função custo é avaliada para todas as sequências que não são promotoras. Neste caso, é retornado o percentual de sequências não promotoras que foram reconhecidas de forma incorreta (PF). Por fim, a função custo de um indivíduo é calculada pela Equação 5.1.

$$F = \frac{(99 - PV) + PF}{2} \tag{5.1}$$

Destaca-se que a função custo é executada utilizando-se sempre o mesmo conjunto de dados de entrada para cada valor de *sigma*. Este conjunto é composto por 100 sequências de DNA que devem ser reconhecidas como uma região promotora e por 100 sequências de DNA que não devem ser reconhecidas como regiões promotoras. Destaca-se que para cada diferente valor de sigma tem-se um conjunto de 100 sequências de DNA verdadeiras (promotores). Já o conjunto de sequência de DNA falsos (não promotores) é o mesmo para todos os valores de sigma.

5.2 Seleção de indivíduos

Para a seleção de indivíduos foi utilizada a seleção por torneio, onde são selecionados 2 indivíduos da população através de uma roleta. Dentre os 2 indivíduos selecionados é escolhido o indivíduo que apresentar o menor valor custo. Este procedimento é repetido até que a nova população esteja completa. Além disto, utilizou-se o elitismo, onde os dois melhores indivíduos de cada geração foram automaticamente selecionados para a próxima geração.

5.3 Operadores Genéticos

O principal operador genético utilizado foi o operador de *crossover*. Por possuir indivíduos com cromossomos formados por valores reais, o *crossover* consistiu na troca de cromossomos completos entre os indivíduos. Foram definidos dois pontos de cortes aleatórios para cada operação efetuada, permitindo a troca de 1 a 6 cromossomos entre os pais. Na Figura 5.2 verificamos um exemplo de um cruzamento entre dois indivíduos hipotéticos, onde os cromossomos 2, 3 e 4 foram trocados entre os pais.

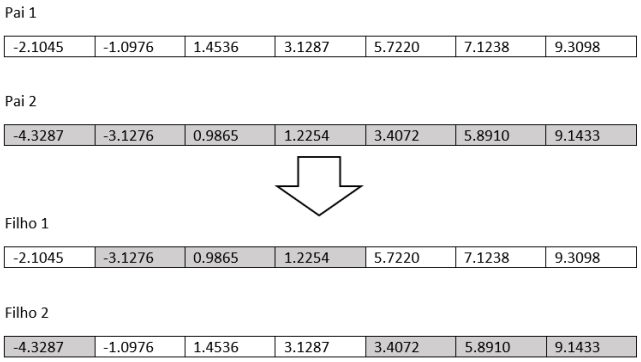


Figura 5.2: Exemplo de Crossover

Para definição da taxa de *crossover*, foram efetuados testes com taxas de 60%, 70%, 80% e 90%, que são os valores que comumente trazem os melhores resultados (LACERDA E. G. M.; CARVALHO, 1999). Na Tabela 5.2 tem-se os valores de *fitness médio* obtidos. Verifica-se que não houve uma variação significativa entre os valores de *fitness médio*. Deste modo, optou-se por utilizar uma taxa de *crossover* de 60% por apresentar um menor custo computacional.

Tabela 5.2: Taxa de Crossover e fitness médio

Taxa de Crossover	**60%**	**70%**	**80%**	**90%**
1ª Geração	49.10	48.71	48.87	49.66
5ª Geração	40.29	34.37	35.59	36.92
10ª Geração	33.95	31.13	30.70	32.02
50ª Geração	30.98	30.92	30.86	31.56
100ª Geração	31.18	31.00	30.86	32.08

O segundo operador genético utilizado foi a operação de mutação, que efetua a alteração de um cromossomo do indivíduo. Ou seja, um cromossomo do indivíuo é selecionado aleatoriamente e então é substituído por um novo cromossomo. O valor do novo cromossomo foi gerado aleatoriamente com valores entre -10 e 10. A Figura 5.3 ilustra uma operação de mutação de um indivíduo hipotético, onde o cromossomo 3 foi substituído por um novo cromossomo gerado aleatoriamente.

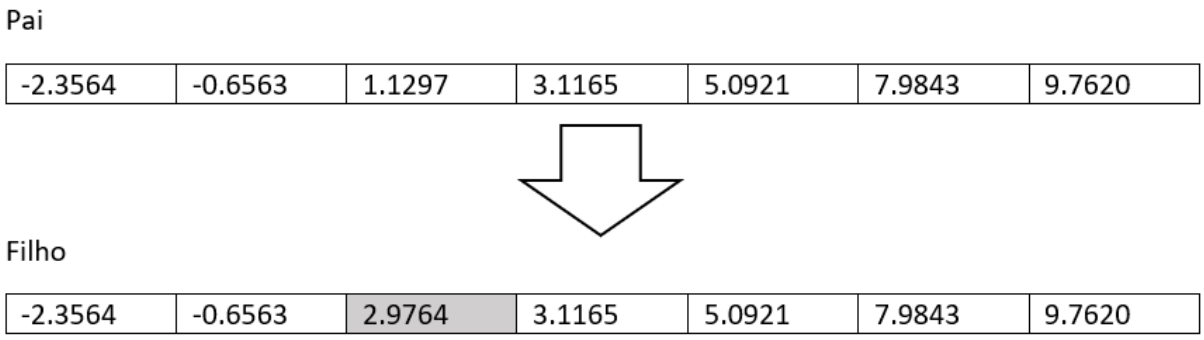


Figura 5.3: Exemplo de Mutação

Para a definição da taxa de mutação, foram realizados testes com taxas de 0,1%, 1%, 3% e 5%, que são as taxas comumente utilizadas, e que trazem os melhores resultado (LACERDA E. G. M.; CARVALHO, 1999). Na Tabela 5.3 tem-se o *fitness médio* obtido para cada uma dessas taxas. Observa-se que não houve diferenças significativas entre esses. Neste caso optou-se por uma taxa de mutação de 0,1% devido ao menor custo computacional.

Tabela 5.3: Taxa de Mutação e fitness médio

Taxa de Mutação	**0,1%**	**1%**	**3%**	**5%**
1ª Geração	49.62	48.93	49.09	48.98
5ª Geração	45.19	42.87	43.79	42.91
10ª Geração	39.35	38.62	39.29	38.76
50ª Geração	30.00	30.00	30.10	30.47
100ª Geração	30.09	30.12	30.41	30.69

5.4 Critério de Parada

Como critério de parada, foi utilizado o número máximo de gerações. Neste caso, o AG foi finalizado após atingir 1000 gerações para cada execução. Testes realizados com todos os valores de sigma mostraram que a convergência foi obtida com um número inferior a 200 gerações. Na Figura 5.4, temos como exemplo a evolução do AG para o sigma 70 durante as 1000 gerações. Na Figura 5.5 é possível verificar, em detalhe, a convergência do AG em cerca de 200 gerações.

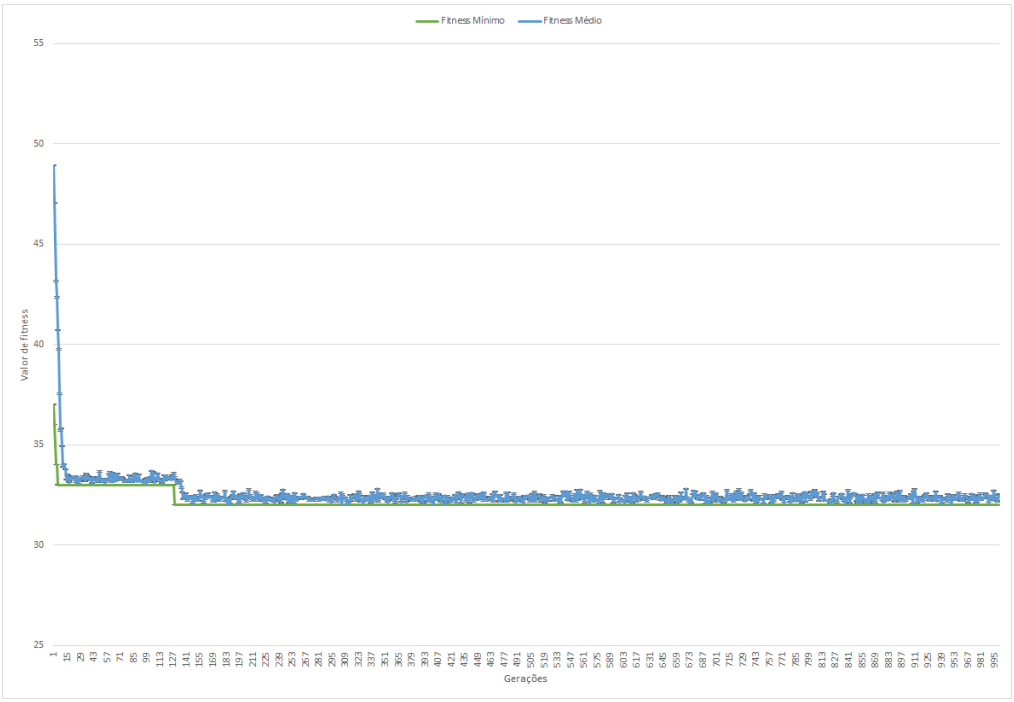


Figura 5.4: Evolução do Algoritmo Genético

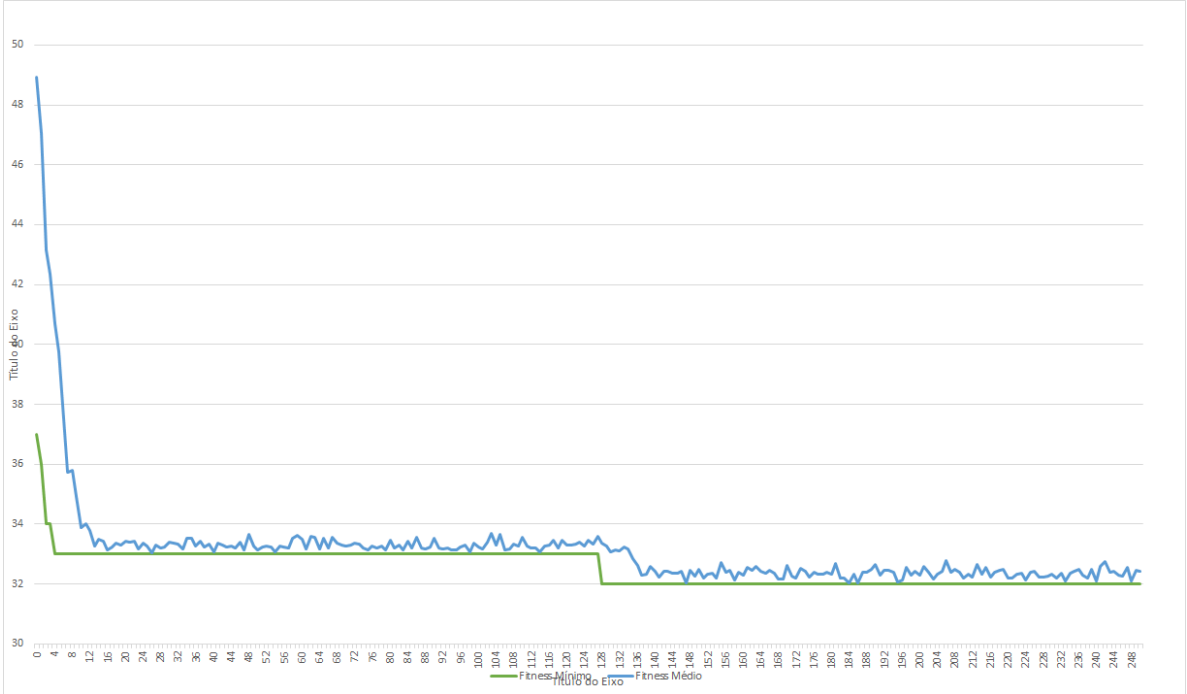


Figura 5.5: Convergência do Algoritmo Genético

5.5 Resultados Obtidos

O AG foi executado individualmente para cada um dos valores de *sigma*. Na Tabela 5.4 podemos visualizar os melhores valores de ponderação que foram obtidos para cada um dos valores de *sigma*. Os valores de ponderação presentes nesta tabela serão utilizados para a comparação com os valores de ponderação que são utilizados no BACPP (ver Tabela 2.1)

Tabela 5.4: Melhores valores de ponderação que foram obtidos para cada valor de sigma

Sigma	Melhor Indivíduo						
24	-8.3279	-7.1062	4.2224	-1.3843	4.9743	8.0845	8.1717
28	-9.9431	3.7106	-1.3737	2.2363	-2.3180	2.3141	9.8984
32	-3.5346	6.192	-3.6645	-1.3807	9.5517	2.0012	8.7469
38	-9.2144	-6.6309	-1.0613	6.8984	6.7771	6.7869	-6.3646
54	-9.6626	-7.7369	6.8202	0.8367	0.6725	0.4171	8.5434
70	-6.8458	-4.0414	-3.1295	7.6268	8.3240	5.6543	2.5226

Para comparação foram selecionados 100 sequências promotoras e 100 sequências que não são promotoras. Para cada valor de sigma foram utilizadas 100 sequências promotoras específicas, enquanto a sequência de não promotores é única para todos os valores de sigma. A análise foi efetuada primeiramente para a implementação original do *BACPP*, para os valores de *sigma*: 24, 28, 32, 38, 54 e 70. Após a mesma análise foi efetuada substituindo-se so valores de ponderação originais do *BACPP* pelos valores de ponderação obtidos através do AG (ver Tabela 5.4).

O AG foi executado em um computador equipado com um processador *Intel core i7-7500U*, com 16 *gigabytes* de memória *RAM*. Nesta configuração de computador, a execução do AG para cada um dos valores de sigma, em uma única instância do processador, necessitou em média de 140 minutos para o processamento. Como forma de diminuir o tempo necessário, foi feita a paralelização do processamento entre os quatro núcleos disponíveis no processador. Para isto, utilizou-se a bibliotececa de paralelização suportada pelo *DEAP*, chamada *SCOOP*. Com o uso do processamento paralelo entre os núcleos, foi possível diminuir o tempo médio de processamento utilizado por cada valor de sigma para cerca de 105 minutos.

Uma primeira abordagem para os resultados foi gerada utilizando-se a probabilidade média obtida na saída do BACPP. Na figura 5.6 tem-se uma comparação de probabilidade média obtida utilizando-se os valores de ponderação antigos do BACPP e os valores calculados neste trabalho utilizando as 100 sequências promotoras. Já na figura 5.7 tem-se uma comparação de probabilidade média obtida utilizando-se os valores de ponderação antigos do BACPP e os valores calculados neste trabalho utilizando as 100 sequências não promotoras.

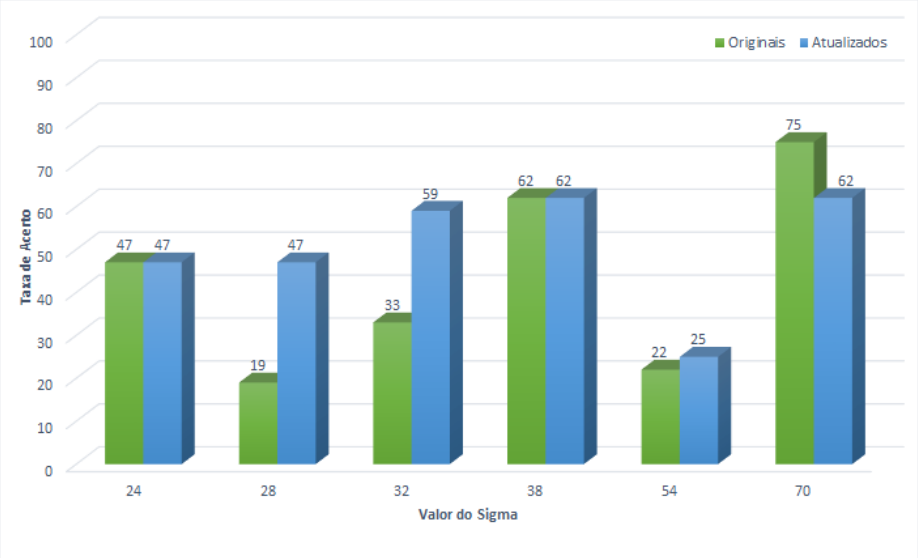


Figura 5.6: Probabilidade média nas Sequências Promotoras

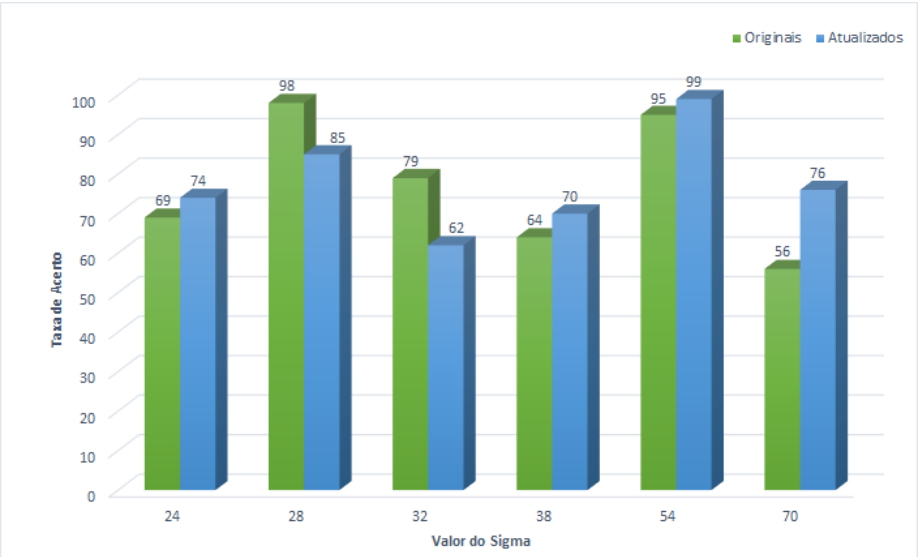


Figura 5.7: Probabilidade média nas Sequências não Promotoras

Na Figura 5.8 tem-se a média das probilidades das sequências de promotores e não promotores. Observa-se que neste caso, os resultados são superiores aos que foram obtidos utilizando os valores antigos do BACPP, com aumento de aproximadamente 4% nas taxas de acerto.

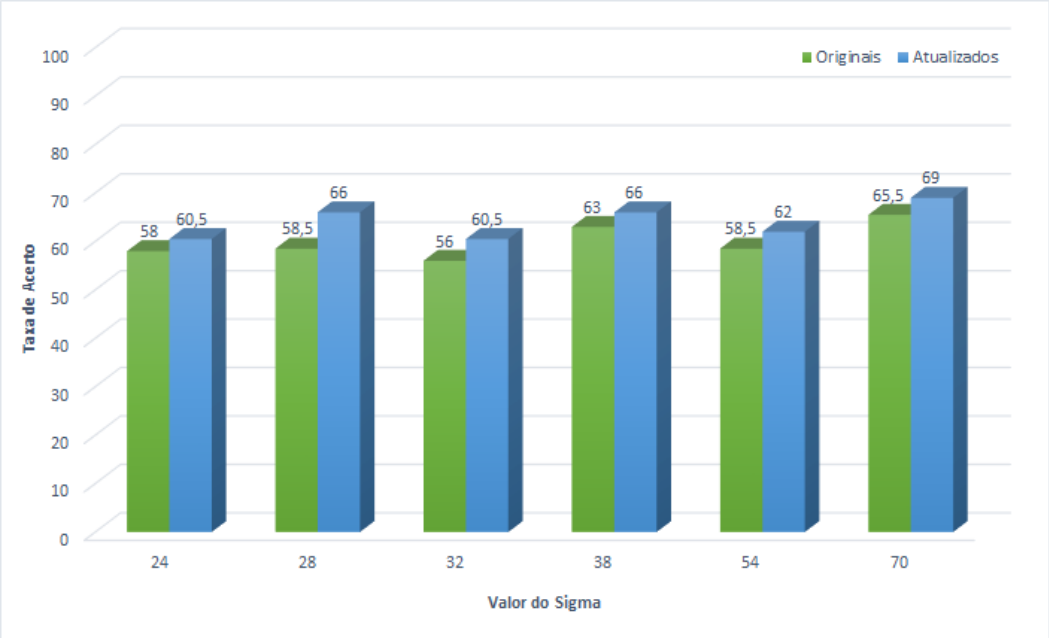


Figura 5.8: Probabilidade média Geral

Outra abordagem foi criada para os resultados através o uso de uma matriz de confusão. Pela matriz de confusão é possível classificar todos os casos possíveis, determinando se o valor real obtido correspondeu ao previsto. Neste caso, foram contabilizados os testes com as 100 sequências promotoras e as 100 não promotoras e os totais foram adicionados à matriz. Para definição de quais sequências foram definidas pelo sistema como promotoras ou não promotoras, foi mantido o ponto de corte utilizado atualmente pelo BACPP.

A matriz de confusão, possui 4 diferentes classificações: os promotores classificados como promotores pelo sistema, são chamados de verdadeiros positivos (VP); Os promotores classificados como não promotores, são conhecidos como falsos negativos (FN); Os não promotores classificados como produtores, são chamados falsos positivos (FP); e os não promotores classificados como não promotores, são chamados de verdadeiros negativos (VN). A Tabela 5.5 nos traz o exemplo da matriz confusão criada para o sigma 70 dos valores de ponderação atualizados. Nela é possíveis verificar que houveram 69 Verdadeiros positivos, 31 falsos negativos, 29 falsos positivos e 71 verdadeiros negativos.

Tabela 5.5: Matriz confusão sigma 70 atualizado

	Promotor	Não Promotor
Promotor	69	31
Não Promotor	29	71

A matriz de confusão foi gerada para todos os sigmas dos valores originais do BACPP, bem como dos valores atualizados gerados através do AG. Os resultados obtidos podem ser vistos na Tabela 5.6.

Tabela 5.6: Resultado da matriz confusão

Sigma	Versão	VP	FP	VN	FN
24	Original	86	76	24	14
24	Atualizado	66	45	55	34
28	Original	27	4	96	73
28	Atualizado	55	29	71	45
32	Original	52	50	50	48
32	Atualizado	72	65	35	28
38	Original	89	71	29	11
38	Atualizado	79	38	62	21
54	Original	38	8	92	62
54	Atualizado	38	3	97	62
70	Original	94	72	28	6
70	Atualizado	69	29	71	31

A partir dos valores obtidos pela matriz de confusão, foram extraídos alguns parâmetros de avaliação, que permitem comparar os novos valores de ponderação gerados, com os valores existentes no BACPP. O primeiro parâmetro é a sensibilidade, responsável por medir a capacidade do sistema em predizer corretamente a condição para casos que realmente a têm. A sensibilidade é calculada pela Equação 5.2.

$$F = \frac{VP}{(VP + FN)} \tag{5.2}$$

O segundo parâmetro é a especificidade, responsável por medir a capacidade do sistema em predizer corretamente a ausência da condição para casos que realmente não a têm. A especificidade é calculada pela Equação 5.3.

$$F = \frac{VN}{(VN + FP)} \tag{5.3}$$

O terceiro parâmetro utilizado é a eficiência, que calcula a média aritmética entre a sensibilidade e a especificidade. a eficiência é calculada pela Equação 5.4.

$$F = \frac{(Sensibilidade + Especificidade)}{2} \tag{5.4}$$

A Figura 5.9 traz um comparativo entre valores de sensibilidade obtidos com os valores de ponderação originais e os valores atualizados para cada um dos sigmas. Verifica-se na imagem, que há uma variação de resultados bastantes grande entre os diferentes valores de sigma, havendo ganho em alguns casos e perda em outros. O mesmo tipo de resultado é observado para os valores de especificidade, vistos na Figura 5.10.

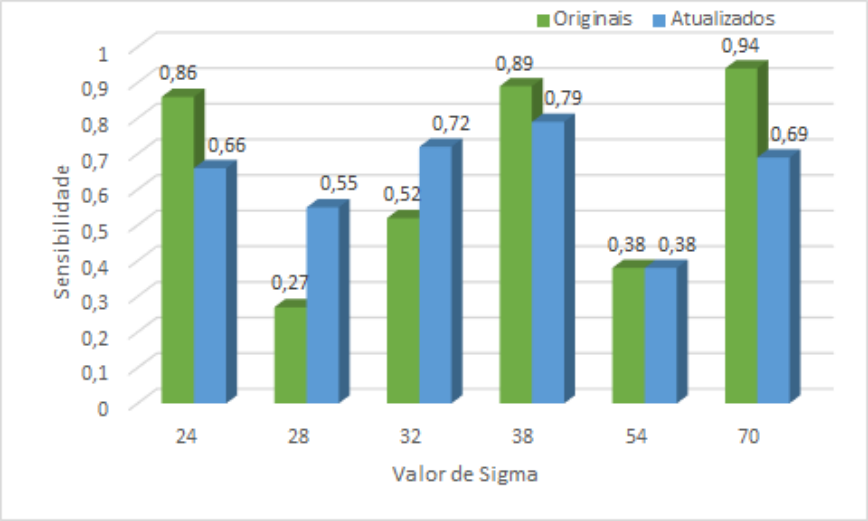


Figura 5.9: Comparativo de sensibilidade

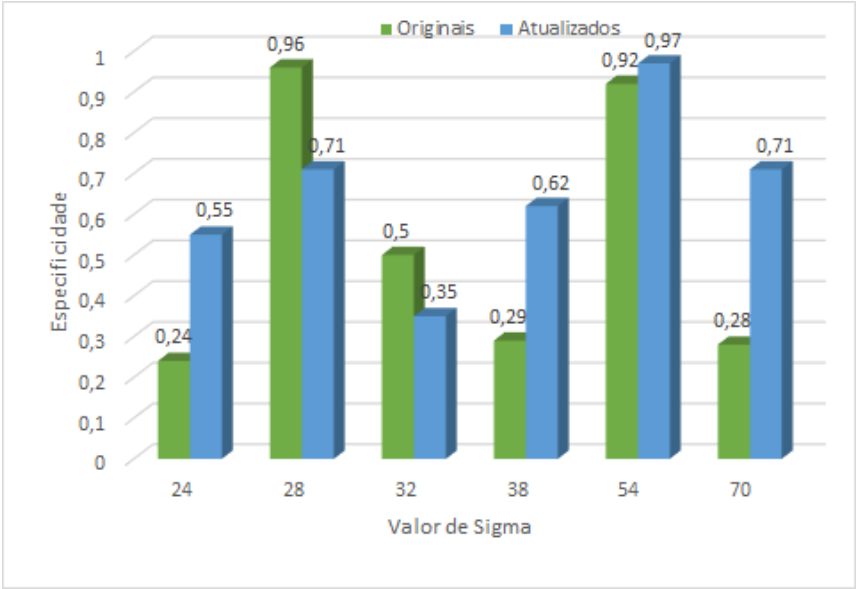


Figura 5.10: Comparativo de especificidade

As variações de resultados presentes na sensibilidade e especificidade, porém, não se repetem para a eficiência. Conforme pode ser visto na Figura 5.11, para todos os valores de sigma observados, quando somadas as variações de sensibilidade e especificidade, há um resultado superior nos valores atualizados, com um aumento médio de 4% em relação aos valores originais.

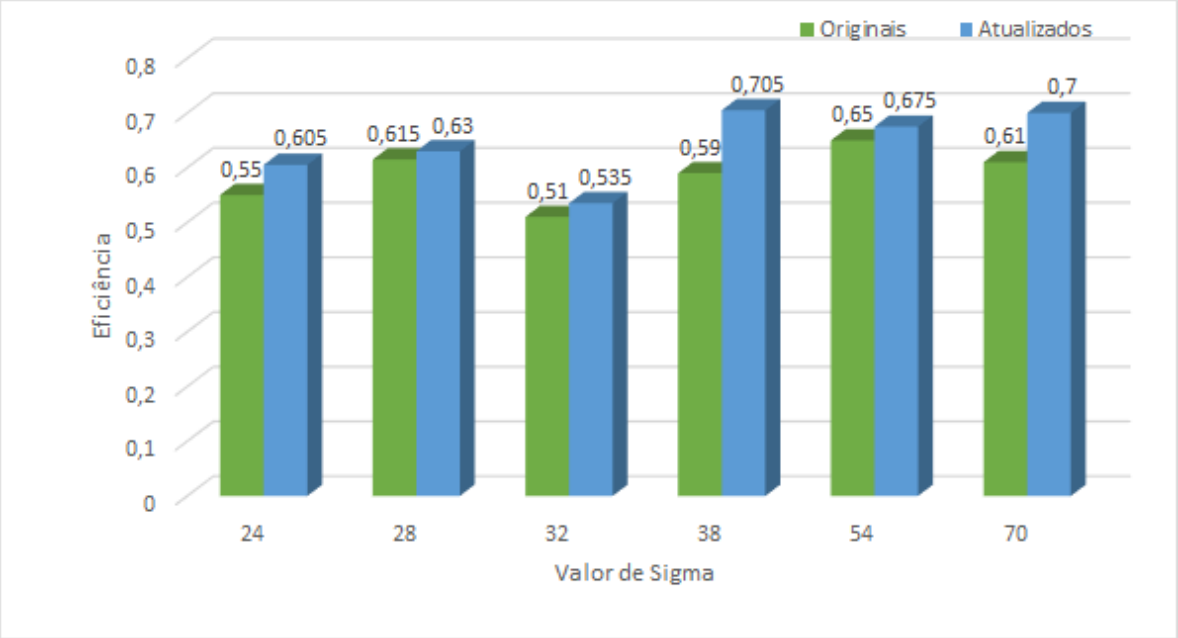


Figura 5.11: Comparativo de eficiência

6 CONSIDERAÇÕES FINAIS

Ao longo do desenvolvimento deste trabalho, realizou-se um estudo sobre o uso de técnicas de minimização, mais especificamente sobre o uso de algoritmos genéticos com o objetivo de aumentar a assertividade no software BACPP.

A codificação dos indivíduos como variáveis do tipo *float*, mostrou-se adequada para o desenvolvimento deste trabalho, facilitando o desenvolvimento do mesmo. Na seleção por torneio, verificou-se que a seleção de uma quantidade maior de indivíduos para o torneio tornava o algoritmo sucetível a mínimos locais. Deste modo, optou-se por utilizar somente dois indivíduos na seleção por torneio. Observou-se ainda que o método convergiu para todos os valores de sigma em aproximadamente 200 gerações. Utilizou-se como critério de convergência um número maior de gerações a fim de evitar futuros problemas de convergência.

Os operadores genético simples (*crossover*, mutação, elitismo) utilizados neste trabalho, atenderam perfeitamente aos objetivos. Porém, como uma futura melhoria é possível a implementação de outros operadores mais complexos e específicos para AGs de representação real, como por exemplo, o *Crossover* Aritmético e a Mutação *CREEP*. Verificou-se ainda que as taxas de *crossover* e mutação utilizadas comumente utilizandas em AGs, trouxeram resultados satisfatórios.

A biblioteca *DEAP* mostrou-se adequada para o desenvolvimento deste trabalho uma vez que facilitou consideravelmente o desenvolvimento do mesmo, devido ao fato de implementar diversos recursos para o desenvolvimento de AGs, além de possibilitar o uso de processamento paralelo, através da biblioteca *SCOOP*. Com a paralelização do processamento, foi possível diminuir o tempo para a execução do AG em cerca de 25%.

Os resultados obtidos com a implementação do AG foram razoáveis, uma vez que houve um aumento de aproximadamente 4 % na probabilidade média, bem como na eficiência, para todos os valores de sigma. Esperava-se que os resultados obtidos fossem maiores, porém para que melhores resultados sejam atingidos, acredita-se ser necessário um número maior de sequência de promotores e não promotores para avaliação da função custo, bem como a utilização de operadores genéticos específicos

para uso com indivíduos de representação real. Estas melhorias foram elencadas como possíveis trabalhos futuros.

6.1 Trabalhos Futuros

Como trabalhos futuros, sugerem-se as seguintes melhorias e modificações:

- Revisão dos valores de corte utilizados pelo BACPP para classificar uma sequência como promotor ou não promotor;
- Implementação de outros operadores genéticos para a busca de melhores indivíduos;
- Utilização de um número maior de sequências de promotores e não promotores na função custo;
- Utilização de outras técnicas de minimização para validação da implementação;

REFERÊNCIAS

ADCOCK, S. **GAUL**: genetic algorithm utility library. <Disponível em: `http://gaul.sourceforge.net/`>. Acesso em 18 de junho de 2017.

ALBERTS, B. **Biologia Molecular da Celula**. [S.l.]: Artmed, 2011.

ANDREWS, P. C. et al. Construction cost and energy performance of single family houses : from integrated design to automated optimization. **Automation in Construction**, [S.l.], v.70, p.1–13, 2016.

CARVALHO, A. C. P. de Leon de. **Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina**. [S.l.]: LTC, 2011.

CHARDON, S. et al. Automating biomedical data science through tree-based pipeline optimization. **Applications of Evolutionary Computation**, [S.l.], p.123–137, 2016.

COPPIN, B. **Inteligência Artificial**. [S.l.]: LTC, 2010.

COX, M. M. **Biologia Molecular**: princípios e técnicas. [S.l.]: ArtMed, 2012.

DENMARK, T. U. of. **Center of Biological Sequence Analysis**. <Disponível em: `http://www.cbs.dtu.dk/services/NetGene2/fasta.php`>. Acesso em 17 de abril de 2017.

GEN, M. **Genetic algorithms and engineering optimization**. [S.l.]: Wiley, 2000.

GENEURA. **Evolving Objects (EO)**: evolutionary computation framework. <Disponível em: `http://eodev.sourceforge.net`>. Acesso em 18 de junho de 2017.

GOLDBERG, D. **Genetic algorithms in search, optimization, and machine learning**. [S.l.]: Addison-Wesley, 1989.

HERRERA, F.; LOZANO, M.; VERDEGAY, J. Tackling Real-Coded Genetic Algorithms: operators and tools for behavioural analysis. **Artificial Intelligence Review**, [S.l.], v.12, n.4, p.265–319, 1998.

HOLD-GEOFFROY, Y.; GAGNON, O.; PARIZEAU, M. Once you SCOOP, no need to fork. In: ANNUAL CONFERENCE ON EXTREME SCIENCE AND ENGINEERING DISCOVERY ENVIRONMENT, 2014. **Proceedings...** [S.l.: s.n.], 2014. p.60.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. [S.l.]: M.I.T.P., 1992.

HUGHES-HALLET, D. **Cálculo de uma Variável**. 3.ed. [S.l.]: LTC, 2003.

KHOURY, M. **Python Strongly Typed Genetic Programming**. <Disponível em: `http://pystep.sourceforge.net/`>. Acesso em 18 de junho de 2017.

LACERDA E. G. M.; CARVALHO, A. C. **Sistemas inteligentes**: aplicações a recursos hídricos e ciências ambientais. [S.l.]: UFRGS, 1999.

LAVAL, U. **Distributed Evolutionary Algorithms in Python**. <Disponível em: `http://deap.gel.ulaval.ca`>. Acesso em 18 de junho de 2017.

LIMA, M. A. C. **GENETICA ON LINE**. <Disponível em: `http://aprendendogenetica.blogspot.com.br/2011/03/genetica-molecular-aula-3-transcricao.html`>. Acesso em 17 de abril de 2017.

LIPAY, M. N. **Biologia Molecular - Métodos e Interpretação - Série Análises Clínicas e Toxicológicas**. [S.l.]: Roca, 2015.

MADIGAN, M. T. **Microbiologia de Brock**. [S.l.]: Artmed, 2011.

MICROPRAXIS. **GAJIT - A Simple Java Genetic Algorithms Package**. <Disponível em: `http://www.micropraxis.com/gajit/index.html`>. Acesso em 18 de junho de 2017.

MITCHELL, M. **An Introduction to Genetic Algorithms**. 5.ed. [S.l.]: A Bradford Book The MIT Press, 1998.

OLSON, R. S. et al. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. **Proceedings of GECCO 2016**, [S.l.], p.485–492, 2016.

PALISADE. **Evolver**: sophisticated optimization for spreadsheets. <Disponível em: `http://www.palisade.com/evolver`>. Acesso em 18 de junho de 2017.

PERONE, C. S. **Pyevolve**. <Disponível em: http://pyevolve.sourceforge.net/>. Acesso em 18 de junho de 2017.

POZO, A. et al. **Computação evolutiva**. <Disponível em: http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf>. Acesso em 17 de abril de 2017.

PROJECT, B. D. G. **Neural Network Promoter Prediction**. <Disponível em: http://www.fruitfly.org/seq_tools/promoter.html>. Acesso em 17 de abril de 2017.

RICH, E. **Artificial intelligence**. 3.ed. [S.l.]: Tata McGraw-Hill, 2009.

SHEFFIELD, U. of. **Genetic Algorithms Toolbox**. <Disponível em: http://codem.group.shef.ac.uk/index.php/ga-toolbox>. Acesso em 18 de junho de 2017.

SILVA, S. Ávila e. **Redes neurais artificiais aplicadas no reconhecimento de regiões promotoras em bactérias Gram-negativas**. 2011. Dissertação (Mestrado em Ciência da Computação) — Universidade de Caxias do Sul.

SILVA, S. Ávila e. **BACPP Bacterial Promoter Prediction**. <Disponível em: http://www.bacpp.bioinfoucs.com/home>. Acesso em 10 de outubro de 2016.

STEIN, L. Genome annotation: from sequence to biology. **Nature reviews genetics**, [S.l.], v.2, n.7, p.493–503, 2001.

VIRGINIA, U. of. **FASTA Sequence Comparison**. <Disponível em: http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml>. Acesso em 17 de abril de 2017.

WALL, M. **GAlib A C++ Library of Genetic Algorithm Components**. <Disponível em: http://lancet.mit.edu/ga>. Acesso em 18 de junho de 2017.

ZAHA, A. **Biologia Molecular Básica**. 5.ed. [S.l.]: Artmed, 2014.