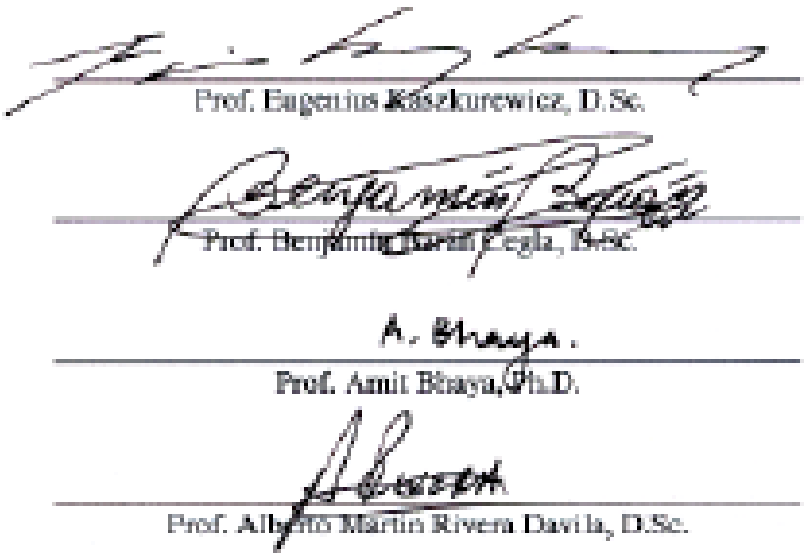
ALGORITMOS EVOLUCIONARIOS MULTIOBJETIVO PARA ALINHAMENTO MÚLTIPLO DE SEQÛËNCIAS BIOLÓGICAS

Margarita Ramona Ruiz Olazar

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:



RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2007

OLAZAR, MARGARITA RAMONA RUIZ

Algoritmos Evolucionários Multiobjetivo para Alinhamento Múltiplo de Seqüências Biológicas [Rio de Janeiro] 2007

XV, 116 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Algoritmos Evolucionários Multi-objetivo.

2. Alinhamento Múltiplo de Seqüências.

I. COPPE/UFRJ II. Título (série)

A minha mãe e a meu pai^{+} por ter me concientizado da importância pelo estudo e por suas incomensuráveis ensinanças.

AGRADECIMENTOS

A Deus, por todas as oportunidades e bênçãos na minha vida.

A minha família que sempre acreditou no meu potencial e me deu seu apoio incondicional em todos os momentos de minha existência. A minha mãe, meus irmãos e irmãs, meus sobrinhos e sobrinhas, por me dar sempre a força e o alento necessários nestes dois anos longe de casa.

Ao professor Eugenius Kaszkurewicz, quem me acolheu como sua orientada e possibilitou que a minha experiência de viver dois anos em função do mestrado fosse a mais proveitosa possível. Agradeço-te pelo inestimável apoio e pela dedicação do seu tempo determinantes para o sucesso deste trabalho.

Ao professor Benjamín Barán, quem com sua forma tão simples e humilde sempre foi o principal modelo a seguir desde os tempos da graduação e foi o motivador pelo qual tive a ousadia de tentar este mestrado. Graças por seu constante apoio e motivação e pelos ensinamentos como mestre e como amigo.

Ao professor Amit Bhaya, pela ajuda e apoio nestes dois anos de pesquisa.

A minha grande amiga Júnia, quem com sua incondicional amizade esteve a meu lado nos maus e bons momentos e tornou minha estada no Rio de Janeiro muito mais agradável. Amizade para toda a vida.

A meus amigos, do NACAD - Núcleo de Atendimento de Computação de Alto Desempenho, do Laboratório de Controle e do Laboratório de Potência, por sua cordial acolhida, amizade e ajuda nos momentos difíceis desta pesquisa.

À Universidade Federal do Rio de Janeiro, e em especial ao Programa de Engenharia Elétrica da COPPE por ter contribuído para minha formação e ter recebido e dado oportunidade a uma estudante paraguaia.

À CAPES pelo fundamental apoio financeiro para o cumprimento desta pesquisa.

Meus sinceros agradecimentos!

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

ALGORITMOS EVOLUCIONARIOS MULTIOBJETIVO PARA ALINHAMENTO MÚLTIPLO DE SEQÜÊNCIAS BIOLÓGICAS

Margarita Ramona Ruiz Olazar

Abril/2007

Orientadores: Eugenius Kaszcurewicz
Benjamín Barán Cegla

Programa: Engenharia Elétrica

Estudamos uma metodologia para Alinhamento Múltiplo de Seqüências biológicas (MSA) usando Algoritmos Evolucionários Multi-objetivo (MOEAs). Este método evolui uma dada população de alinhamentos gradualmente, melhorando o "fitness" da população medida por dois critérios; a qualidade do alinhamento calculada com a função "Soma de pares" utilizando a matriz de substituição de resíduos BLOSUM62 e a qualidade do alinhamento calculada com a função "Soma de pares" utilizando a matriz de substituição de resíduos PAM250.

Este problema, em geral, demanda tempo elevado de processamento, e a implementação proposta pretende tirar proveito da computação de alto desempenho uma vez que o programa é executado em paralelo por vários processadores.

As vantagens da metodologia proposta é que ela pode ser usada tanto para seqüências de proteínas como de DNA, além de apresentar a possibilidade de otimizar diferentes funções objetivo, qualquer sejam estas. Os resultados obtidos demostram que os MOEAs são métodos efetivos e eficientes de otimização e podem ser utilizados em problemas de MSA, quando o domínio do problema é determinado.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MULTIOBJECTIVE EVOLUCIONARY ALGORITHMS FOR BIOLOGICAL MULTIPLE SEQUENCES ALIGNMENT

Margarita Ruiz

Abril/2007

Advisors: Eugenius Kaszcurewicz
Benjamín Barán Cegla

Department: Electrical Engineering

We studied a methodology for the Biological Multiple Sequences Alignment (MSA) using Multi-Objective Evolutionary Algorithms (MOEAs). This method evolves gradually a given population of alignments, improving the "fitness" of the population measured by two criteria; the quality of the alignment calculated with the "SP score" function using the matrix of substitution BLOSUM62 and the quality of the alignment calculated with "SP score" function using the matrix of substitution PAM250.

In general, this problem demands a huge processing time; therefore, the proposed implementation tries to benefit from high performance computation using several processors to run a parallel program.

The main advantage of the proposed methodology is that it can be used for protein and DNA sequences as well. In addition, it gives the possibility of optimizing different objective functions at a time. Experimental results show that MOEAs are efficient methods of optimization and can be used in MSA problems, when the problem domain is determined.

SUMÁRIO

5. Experimentos e Resultados

LISTA DE FIGURAS

LISTA DE TABELAS

SÍMBOLOS E ABREVIATURAS

DNA *Acido Desoxirribonucléico*

FO *Função Objetivo*

AG *Algoritmos Geneticos*

MSA *Multiple Sequence Alignment*

MOEA *Multiobjective Optimization Evolutionary Algorithm*

MOP *Multiobjective Optimization Problem*

NSGA-II *Non-dominated Sorting Genetic Algorithm II*

pMOEA *Parallel Multiobjective Optimization Evolutionary Algorithm*

SPEA *Strength Pareto Evolutionary Algorithm*

EMO *Evolucionary Multiobjective Optimization*

SP score *Sum of Pairs score*

PAM *Point Accepted Mutation*

BLOSUM *BLOck SUbstitution Matrix*

SPS *Sum-of-pairs Score*

CS *Column Score*

Símbolo	**Significado**
$\in$	Símbolo pertence a
max{}	Máximo de os valores contidos em {}
$\cup$	Símbolo união
$\mathcal{A}$	Alfabeto Base de as Seqüências
$\mathcal{A}_{DNA}$	Alfabeto Base das Seqüências de DNA
$\mathcal{A}_{Proteínas}$	Alfabeto Base das Seqüências de Proteínas
-	Símbolo de gap
$\mathcal{A}'$	Alfabeto estendido
X^i	Seqüência i não alinhada
S^i	Seqüência alinhada i
x_j^i	A unidade j da seqüência i
#	Operador de alinhamento
s_j^i	A subunidade j da seqüência alinhada i

$\sum$	Símbolo somatória
$\mathbf{P}_{s_k^i s_k^j}$	Pontuação dada às subunidades alinhadas na posição *k* das seqüências S^i e S^j
S^i e S^j	Seqüência i alinhada
$\boldsymbol{A}$	Alinhamento
|	Correspondência de subunidades
G	Penalidade de gap
$\boldsymbol{F(A)}$	Função mede a qualidade de um alinhamento
$\boldsymbol{C}(S^i, S^j)$	Função que calcula a qualidade do alinhamento par S^i e S^j
$F_{SP}(A)$	Função Soma de pares de A
$F(x)$	Vetor de funções objetivo a otimizar
$f_i(x)$	Função objetivo *i*
$g_j(x)$	Vetor de restrições
$h_k(x)$	Vetor de restrições
u,v	Vetores
E	Espaço de busca ou região factível
Z	Espaço de objetivos
D	Espaço de decisão ou espaço de decisão
$\preceq$	Operador de dominância
$\forall$	Símbolo que significa: Para todo
$\wedge$	Símbolo que significa: e
$\exists$	Símbolo que significa: Existe
x^*	Solução que pertence ao espaço D
P^*	Conjunto de ótimo pareto
FP^*	Frente de Pareto
F_{si}	Valor de aptidão do individuo i
$\phi(\boldsymbol{d}_{ij})$	Sharing function
σ_{share}	Parâmetro que determina a proximidade permitida entre indivíduos
w_m	Peso dado à função objetivo *m*
$i_{distance}$	Estimação de densidade
Pop	População genética
P_{nd}	População de elementos não dominados

N_{max}	Número máximo de indivíduos de Pop
N_{nd}	Número de indivíduos de P_{nd}
I[*i*]	Solução *i* em I
I[*i*].*m*	Valor de I da *m-ésima* FO do *i-ésimo* individuo no conjunto I
f_m^{max}, f_m^{min}	Máximo e mínimo valor da *m-ésima FO*
$\prec_n$	Operador crowded-comparison
i_{rank}	Hierarquia de não dominação
$\mathbf{g_a}$	Gap de abertura
$\mathbf{g_e}$	Gap de extensão
$C_{PAM}(S^i,S^j)$	Função que calcula a pontuação de duas subunidades de S^i *e* S^j com a tabela PAM 250
$C_{BLOSUM}(S^i,S^j)$	Função que calcula a pontuação de duas subunidades de S^i *e* S^j com a tabela BLOSUM62
$\boldsymbol{F_{Q1}(A)}$	Função de qualidade do alinhamento usando matriz PAM250
$\boldsymbol{F_{Q2}(A)}$	Função de qualidade do alinhamento usando a matriz BLOSUM62
P1, P2	Pesos dados às funções objetivo no algoritmo Soma Ponderada.
w^i	Valor que indica a ponderação da seqüência *i*

CAPÍTULO

1

INTRODUÇÃO

1.1 Contexto

As técnicas de resolução de problemas baseados em computadores são hoje em dia ferramentas fundamentais de tomada de decisões. Existe uma ampla classe de problemas e alguns deles muito complexos para os quais não se dispõe ferramentas que os resolvam com precisão utilizando uma quantidade moderada de recursos.

Um dos campos onde este tipo de problemas complexos aparece em grande número é da Biologia Molecular, que apresentou avanços muito significativos nas últimas décadas. No caso dos biólogos, estes freqüentemente trabalham com uma grande quantidade de informação, geradas a partir de experimentos em laboratório. Dada a necessidade de manipular essa informação, surgiu a Bioinformática, que aplica técnicas computacionais, matemáticas e estatísticas para a análise de dados de interesse biológico.

Um dos principais problemas dentro desta área é a comparação de seqüências moleculares de DNA ou Proteínas. A técnica mais freqüentemente usada para comparar duas ou mais seqüências de DNA ou proteínas consiste em sobrepor uma cadeia sobre outra e procurar semelhanças e diferenças, esta operação é formalmente denominada alinhamento [SETUBAL & MEIDANIS, 1997].

Naturalmente, é desejável que para cada problema possamos encontrar a solução ótima através de algoritmos exatos, entretanto as modelagens associadas à grande parte dos problemas em Bioinformática, dão lugar a problemas NP-Completos [WANG & JIANG, 1994; DELLA VEDOVA & BONIZZONI, 2001] e conseqüente-mente, devem ser abordados através de técnicas heurísticas. Não obstante, esta área é especialmente atraente já que a resolução destes problemas e a obtenção de novos mecanismos de solução têm impacto em, principalmente, duas áreas: as Ciências da Computação e a Biologia.

1.2 Motivação

As ferramentas existentes para alinhamentos múltiplos de seqüências de DNA e proteínas apresentam em geral boas respostas, mas ainda podem ser melhoradas. Muitas vezes os biólogos precisam fazer intervenções manuais sobre os alinhamentos para obter uma solução que tenha sentido biológico. Também, às vezes, é necessário que os biólogos realimentem as ferramentas com diferentes parâmetros até que se obtenha uma boa solução. É necessário então aperfeiçoar os métodos existentes para alinhamentos múltiplos de forma que diminua o esforço humano para resolvê-los.

Tradicionalmente, o Problema de Alinhamento de Seqüências Biológicas é formulado como um problema de otimização de objetivo simples [ZHANG, C, 1997]. A Função Objetivo (FO) mais popularmente utilizada é aquela que avalia a qualidade do alinhamento estabelecendo uma pontuação em cada coluna do alinhamento e então, procurando a melhor pontuação possível. Se a Função Objetivo é bem escolhida ou é uma medida precisa dessa qualidade, então esta aproximação correspondente indica que o alinhamento resultante é o melhor por algum critério. No entanto, não existe uma Função Objetivo que possa ser considerada mais adequada para indicar que um alinhamento é preferido a outro, ou indicar que se trata do melhor alinhamento possível [NOTREDAME C., 2002].

Consideramos o problema de Alinhamento Múltiplo de Seqüências (MSA) como um problema que usualmente requer uma busca de soluções que satisfaçam de forma simultânea múltiplos critérios de desempenho e objetivos, que podem ser conflitantes. A Otimização Multi-objetivo aborda problemas deste tipo, afim de encontrar um conjunto de soluções que representem um "compromisso" entre esses objetivos. [DEB K., 2001]

O uso de algoritmos eficientes reflete-se diretamente na qualidade das soluções que podem ser encontradas e na velocidade de execução. O que pretendemos é ampliar o número de soluções válidas considerando funções que respondem bem ao cálculo de alinhamentos locais assim como globais. Para isto propomos aplicar Algoritmos Evolucionários Multi-objetivo (MOEA) de forma a encontrar um conjunto de alinhamentos ótimos atendendo a diferentes FOs.

1.3 Revisão Bibliográfica

A comparação de seqüências é uma operação muito importante para a análise de seqüências biológicas. Comparar seqüências significa encontrar quais partes destas são parecidas e quais são diferentes. Estas diferenças são determinadas fazendo o alinhamento das mesmas.

O método de comparação visual utilizado nas primeiras pesquisas era muito tedioso e a determinação da significância dos resultados usualmente era intuitiva, até que algumas aproximações baseadas em computação e estatística surgiram na década de 60, [FITCH, 1966; NEEDLEMAN & BLAIR, 1969], mas foi em 1970 que Needleman e Wunsh [NEEDLEMAN & WUNSH, 1970] desenvolveram um algoritmo muito eficiente para comparar duas seqüências baseando-se na programação dinâmica, posteriormente outros algoritmos de muito sucesso e precisão matemática para alinhamento de duas seqüências foram desenvolvidos [SANKOFF, 1975; SMITH & WATERMAN, 1981].

Estes algoritmos baseados na programação dinâmica garantem um alinhamento matemático ótimo, mas o método é limitado a um pequeno número de seqüências curtas já que o poder computacional requerido para grandes alinhamentos é muito grande. Para superar este problema, várias aproximações heurísticas foram desenvolvidas conduzindo a uma enorme quantidade de programas usando diversos algoritmos.

As maiorias das heurísticas para MSA usam uma aproximação progressiva, onde o alinhamento múltiplo é construído gradualmente alinhando primeiro as seqüências mais proximamente relacionadas e sucessivamente agregando as mais distantes, as distâncias são proporcionadas pela construção de uma árvore guia. Os programas, CLUSTALW [THOMPSON JD, HIGGINS DG AND GIBSON TJ, 1994] e MULTAL [TAYLOR, 1988] utilizam o algoritmo Neighbour-Joining para construir a árvore guia, MULTIALIGN [BARTON & STERNBERG, 1987] e PILEUP(Wisconsin Package v.8; Genetics Computer Group, Madison, WI) constroem a árvore guia usando o método UPGMA [SNEATH & SNOKAL, 1973]. Aproximações progressivas têm como vantagem a rapidez e simplicidade, mas sua principal desvantagem é que não obtém bons alinhamentos locais e, portanto não são muito consistentes.

Outras alternativas de solução foram propostas, como o uso de modelos de Cadeia de Markov (HMMs) [KROGH et al., 1994], que tenta encontrar simultaneamente um modelo de alinhamento e probabilidade de substituições, inserções e deleções que são auto consistentes tal como o programa de computador "*MSA"* que propõe uma solução limitada para 7 u 8 seqüências como máximo[LIPMAN, ALTSCHUL, KECECIOGLU, 1989] [GUPTA, KECECIOGLU AND SCHAFFER, 1996].

Também o uso de métodos estocásticos como Simulated Annealing [ISHIKAWA ET AL., 1993] [AART & VAN LAARHOVEN, 1987], Gibbs sampling [LAWRENCE et al., 1993] tentam encontrar alinhamentos ótimos ou quase ótimos. O uso de Gibbs sampling foi ótimo para encontrar o melhor bloco de alinhamento múltiplo local sem *gaps*, mas sua aplicação com *gaps* não é trivial.

Também muitas tentativas foram feitas usando estratégias iterativas, como aplicações desenvolvidas usando Algoritmos Genéticos (AG), alguns sem muito sucesso por ter uma velocidade de cálculo muito lenta em comparação com as heurísticas progressivas, mas capazes de encontrar um alinhamento múltiplo ótimo global em tempo razoável. Esta aproximação tem a vantagem que pode ser usada para otimizar qualquer função objetivo. [ZHANG, 1997; NOTREDAME E HIGGINS, 1996; ZHANG E WONG, 1997; HANADA, 2000; HORNG et al, 2004].

Lamentavelmente nenhuma destas aproximações provê soluções completamente satisfatórias, razão pela qual novos métodos de solução são propostos por pesquisadores. Existem modelos matemáticos que se ajustam melhor a natureza destes problemas, estes são conhecidos como otimização com objetivos múltiplos o multi-objetivo. Considera-se que grande parte dos problemas do mundo real implicam na otimização simultânea de vários objetivos que geralmente apresentam conflitos entre si; o seja, a melhora de um deles conduz à deterioração de outro.

Estes modelos originam-se na "Programação Evolutiva" (PE), idéias apresentadas por Lawrence J. Fogel, que concebeu o uso da evolução simulada na solução de problemas na década de 60 [FOGEL, 1964], e Peter Bienert, Ingo Rechenberg e Hans-Paul Schwefel que desenvolveram um método de ajustes discretos aleatórios inspirados no mecanismo de mutação que ocorre na natureza, esta técnica foi denominada "Estratégias Evolutivas". [RECHENBERG I., 1973] [BREMERMANN H.

J., 1962] [SCHWEFEL H. P., 1965]. Hoje em dia existem variantes em diversos domínios de aplicação, um aporte muito importante foi feito por John H. Holland que introduz os Algoritmos Genéticos (chamados originalmente planos reprodutivos) e são a técnica evolutiva mais popular da atualidade. [HOLLAND, 1962, 1975].

Os Algoritmos Evolutivos (AE) são especialmente adequados para otimização Multi-objetivo. Desde a década de 80s, com o aumento do poder de computação, foram desenvolvidos vários AE, capazes de buscar múltiplas soluções "*ótimas"* de forma simultânea. Na década de 90 estabeleceram-se como alternativas práticas, sendo utilizados com sucesso na resolução de uma ampla variedade de problemas de otimização e busca de grande complexidade. No transcorrer dos anos, a Otimização Evolutiva Multi-objetivo tornou-se uma área importante de pesquisa e recebe cada vez maior atenção. [COELLO, 1999, 2001]

Com o uso cada vez mais estendido destes algoritmos em problemas reais de otimização e de complexidade crescente, foi necessário melhorar a efetividade deles. Uma alternativa foi a incorporação de conceitos de paralelismo ao projeto destes algoritmos, vários modelos de paralelização foram desenvolvidos nesta área demonstrando ser uma técnica efetiva [VAN VELDHUIZEN & LAMONT, 2000] [BARÁN & DUARTE, 2001] [ZITZLER & THIELE, 1999] [AZARM, REYNOLDS & NARAYANAN, 1999] [VAN VELDHUIZEN, 1999] [ZITZLER, DEB & THIELE, 1999] [ANG & LI, 2001].

Recentemente alguns estudos aplicando Otimização Multi-objetivo (MOP) foram desenvolvidos para encontrar um conjunto de alinhamentos ótimos, usando objetivos e critérios diferentes [CANCINO W., 2003; CHELLAPILLA & FOGEL, 1999; ZWIR, 2002; SEELUANGSAWAT E CHONGSTITVATANA, 2005; CHENG et al., 2005]. Esta heurística está em constante crescimento, pois é uma boa alternativa de otimização já que proporciona eficácia e robustez na busca de um conjunto de soluções viáveis.

1.4 Objetivos

Existe um grande número de algoritmos evolutivos paralelos para otimização multi-objetivo com muito bom desempenho em problemas reais de otimização. O objetivo deste trabalho é apresentar três modelos de Algoritmos Evolutivos Multi-

objetivo paralelos para o alinhamento de seqüências protéicas. Os resultados fornecidos pelos algoritmos serão avaliados usando métricas de desempenho usuais e comparando-os com outros softwares popularmente utilizados na resolução de MSA.

As aplicações para resolver problemas de MSA em geral demandam um grande tempo de processamento, nossa implementação pretende tirar proveito da supercomputação ao permitir que o programa seja executado em paralelo por vários processadores.

1.5 Estrutura do trabalho

O presente trabalho apresenta no segundo capítulo os conceitos básicos da comparação de seqüências biológicas e as heurísticas mais utilizadas para sua resolução. Após, no capitulo três apresentamos o problema de Otimização Multi-objetivo e a descrição de alguns algoritmos. O método proposto para a resolução do problema de MSA é descrito no capítulo quatro. Finalmente, os experimentos e resultados serão mostrados no capítulo 5, assim como também as conclusões e trabalhos futuros.

CAPÍTULO

2

ALINHAMENTO DE SEQÜÊNCIAS DE BIOMOLÉCULAS

No capítulo anterior apresentamos uma introdução geral deste trabalho de pesquisa, os objetivos procurados e a organização deste documento. Neste capítulo será abordado o problema do alinhamento de seqüências, descrevendo alguns conceitos importantes para o entendimento deste trabalho.

2.1 O problema da comparação de Seqüências

Este é atualmente um dos problemas mais importantes da biologia computacional, dado o número e diversidade de seqüências existentes e dada a freqüência com que ele precisa ser resolvido diariamente. A comparação de seqüências de genomas de organismos é uma das aplicações computacionais mais utilizadas por biólogos moleculares. A descoberta de homología entre seqüências de uma proteína ou famílias de proteínas freqüentemente traz as primeiras pistas a respeito da função de um novo gene seqüênciado.

O alinhamento de seqüências biológicas é uma operação básica em Bioinformática, por servir de suporte pra outros processos como, por exemplo, a determinação da estrutura tridimensional das proteínas e análises filogenéticas. O processo de evolução continuamente re-utiliza (ou duplica) estruturas que dão certo. "Biologia é baseada em enorme redundância" (R. Doolittle).

2.1.1 Alinhamento de seqüências

Uma forma de fazer a comparação de seqüências é mediante o alinhamento das mesmas. Um alinhamento de duas seqüências é obtido inserindo espaços (dentro ou nos extremos da seqüência) e então localizando as duas seqüências resultantes uma acima

da outra, tal que cada caráter ou espaço na seqüência oposta seja único na outra seqüência, a Figura 2.1 mostra dois alinhamentos. O espaço ou caráter nulo inserido "-" é conhecido como **gap** ou buraco. As seqüências devem ter o mesmo comprimento.

```
SEQ 1   a  c  g  t  g  c      SEQ 1   a  c  g  t  g  c
        |        |  |  |              |        |  |  |
SEQ 2   a  a  -  t  g  c      SEQ 2   a  -  a  t  g  c
```

Figura 2.1: Alinhamento de duas seqüências

2.1.2 Tipos de Alinhamento

- *Alinhamento Global*: As seqüências são alinhadas de um extremo ao outro, sobre seu comprimento total, dando origem a apenas um resultado.
- *Alinhamento Local*: procura-se alinhar apenas as regiões mais conservadas entre seqüências, independente da localização relativa de cada região em sua seqüência. Dessa forma, o alinhamento resulta em uma ou mais regiões conservadas entre as seqüências.

2.2 Descrição da representação utilizada neste trabalho.

Uma ***molécula*** de DNA pode ser descrita completamente se a seqüência de nucleotídeos que a compõem for conhecida, e no caso das proteínas se for conhecida a seqüência de aminoácidos que a compõem. A determinação da seqüência exata chama-se **s*eqüenciamento*** desta molécula.

Cada ***nucleotídeo*** pertencente a seqüências de DNA é representado por uma das quatro letras de um alfabeto que chamamos ***alfabeto base das seqüências de DNA***, definido como $\mathcal{A}_{DNA}$= {A, C, G, T}.

Cada ***aminoácido*** pertencente à seqüência de Proteínas é representado por uma das vinte letras do alfabeto que chamamos ***alfabeto base das seqüências de proteínas***, definido como: $\mathcal{A}_{Proteinas}$ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}.

Chamamos de ***base*** cada letra (nucleotídeo ou aminoácido) do alfabeto de seqüências de DNA ou Proteínas que compõem $\mathcal{A}_{\text{DNA}}$ ou $\mathcal{A}_{\text{Proteínas}}$ respectivamente.

Um alinhamento de seqüências de DNA ou proteínas é obtido introduzindo espaços dentro ou nos extremos das seqüências a serem alinhadas, de forma que a ordem das bases em cada seqüência seja preservada e as seqüências tenham o mesmo comprimento.

Exemplo 2.1: Sejam duas seqüências GLISVT e GIVT um possível alinhamento é:

G	L	I	S	V	T
G	I	V	-	-	T

Ao espaço introduzido '-' chamaremos ***gap*** e a cada caracter que compõe as seqüências alinhadas chamaremos ***subunidade***. Então cada subunidade é uma base (letra do $\mathcal{A}_{\text{DNA}}$ ou $\mathcal{A}_{\text{Proteínas}}$) ou um gap e estes pertencem a um alfabeto que chamaremos ***Alfabeto estendido*** $\mathcal{A}$', que é definido como $\mathcal{A}' = \mathcal{A}_{\text{DNA}} \cup \{\text{'-'}\}$ ou $\mathcal{A}' = \mathcal{A}_{\text{Proteínas}} \cup \{\text{'-'}\}$.

Formalmente, dado um conjunto de *n* seqüências, $\boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^n$

onde:

$$\boldsymbol{X}^1 = \boldsymbol{x}_1^1 \boldsymbol{x}_2^1 ... \boldsymbol{x}_{\boldsymbol{m}_1}^1,$$
$$\boldsymbol{X}^2 = \boldsymbol{x}_1^2 \boldsymbol{x}_2^2 ... \boldsymbol{x}_{\boldsymbol{m}_2}^2,$$
$$...$$
$$\boldsymbol{X}^n = \boldsymbol{x}_1^n \boldsymbol{x}_2^n ... \boldsymbol{x}_{\boldsymbol{m}_n}^n$$

Denotamos $\boldsymbol{x}_j^i \in \mathcal{A}_{\text{Proteínas}}$ ou $\boldsymbol{x}_j^i \in \mathcal{A}_{\text{ADN}}$

onde $1 \le \boldsymbol{i} \le \boldsymbol{n}$ **e** $1 \le \boldsymbol{j} \le \boldsymbol{m}_i$ e $\boldsymbol{m}_i$ é o comprimento da seqüência i.

Definição 2.1: Alinhamento de *n* seqüências.

O alinhamento de *n* seqüências $\boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^n$ denotado como $\boldsymbol{S}^1 \# \boldsymbol{S}^2 \# ... \# \boldsymbol{S}^n$, onde # representa a operação de alinhamento, satisfaz as seguintes condições:

1. Todas as seqüências têm o mesmo comprimento $\boldsymbol{m}$, onde $\boldsymbol{m} \geq \max_{i}\{\boldsymbol{m}_i\}$.
2. A ordem das bases na seqüência é preservada no alinhamento.
3. Não existe, no alinhamento uma coluna na qual todas as subunidades são gaps.

Então o alinhamento resultante pode ser visto como uma matriz **A** de *n* linhas por *m* colunas.

$$
\begin{array}{rcccccc}
\boldsymbol{S}^1 & = & \boldsymbol{s}_1^1 & \boldsymbol{s}_2^1 & .... & \boldsymbol{s}_m^1 \\
 & & | & | & | & | \\
\boldsymbol{S}^2 & = & \boldsymbol{s}_1^2 & \boldsymbol{s}_2^2 & .... & \boldsymbol{s}_m^2 \\
 & & | & | & | & | \\
 & & .... & .... & .... & .... \\
 & & | & | & | & | \\
\boldsymbol{S}^n & = & \boldsymbol{s}_1^n & \boldsymbol{s}_2^n & .... & \boldsymbol{s}_m^n
\end{array}
$$

onde $\boldsymbol{s}_j^i \in \mathcal{A}'$ é chamada subunidade (incluindo os gaps) e $1 \leq i \leq n$ y $1 \leq j \leq m$

e | indica a correspondência de subunidades na mesma coluna.

2.3 Esquema de Valoração para um Alinhamento de duas Seqüências.

Considerando o modelo descrito anteriormente para obter um alinhamento, podemos definir um esquema de valoração como método de medida de sua ótimalidade, esta valoração é um número real. Um alinhamento de duas seqüências pode ser visto como uma matriz $\mathbf{A} = [S^i\ S^j]^T$.

$$
\begin{array}{rcccccc}
 & & \boldsymbol{s}_1^i & \boldsymbol{s}_2^i & ... & \boldsymbol{s}_m^i \\
\boldsymbol{A} & = & | & | & & | \\
 & & \boldsymbol{s}_1^j & \boldsymbol{s}_2^j & ... & \boldsymbol{s}_m^j
\end{array}
$$

A cada coluna do alinhamento damos uma valoração, independentemente de outras colunas:

- A valoração de uma coluna k ($1 \leq k \leq m$) do alinhamento das seqüências S^i e S^j é denotado como $\boldsymbol{P}_{s_k^i s_k^j}$.
- Se $\boldsymbol{s_k^i}$ ='–' ***ou*** $\boldsymbol{s_k^j}$ ='–' então $\boldsymbol{P}_{s_k^i s_k^j} = -\boldsymbol{G}$ onde $\boldsymbol{G}$ é chamado penalidade do gap.

A valoração total do alinhamento $\boldsymbol{A}$ pode ser calculada como a soma das valorações de todas as colunas do alinhamento.

Então $\boldsymbol{F}(\boldsymbol{A}) = \sum_{k=1}^{m} \boldsymbol{P}_{s_k^i s_k^j}$ onde m é a quantidade de colunas no alinhamento.

Note que a penalidade de uma coluna com gaps tem valor negativo.

Exemplo 2.2:

Dada as seqüências S^1: VEITGEIST e S^2: PRETERIT

e um esquema de valoração

- $\boldsymbol{P}_{s_k^i s_k^j} = 1$ ***se*** $\boldsymbol{s_k^i = s_k^j}$ ***e*** $\boldsymbol{P}_{s_k^i s_k^j} = 0$ ***se*** $\boldsymbol{s_k^i \neq s_k^j}$.
- $\boldsymbol{G} = 1$.

Calculamos a valoração total de três possíveis alinhamentos:

Alinhamento 1:

S^1:	V	-	E	I	T	G	E	I	S	T
S^2:	P	R	E	-	T	E	R	I	-	T
P	0	-1	1	-1	1	0	0	1	-1	1

F(*A*) = 1

Alinhamento 2:

S^1:	V	E	I	T	G	E	I	S	T
S^2:	P	R	E	T	-	E	R	I	T
P	0	0	0	1	-1	1	0	0	1

F(*A*) = 2

Alinhamento 3:

S^1:	-	V	E	I	T	G	E	-	I	S	T	
S^2:	P	R	E	-	T	-	E	R	I	-	T	
P	-1	0	1	-1	1	-1	1	-1	1	-1	1	**F(*A*)** = 0

O alinhamento o qual têm o valor de **F(*A*)** mais alto depende do esquema de valoração escolhido. Podemos dizer então que encontrar o alinhamento com o maior valor de **F(*A*)** não necessariamente significa que este seja "o melhor alinhamento", se o esquema de valoração não é bom. Por isto, escolher qual esquema usar é uma importante e difícil tarefa. Também poderíamos ter mais de um alinhamento com o máximo valor.

2.4 Esquema de Valoração para um Alinhamento Múltiplo de Seqüências.

Para encontrar automaticamente o "melhor" alinhamento múltiplo entre várias alternativas, usamos um esquema de valoração que também é aplicável a subconjuntos de alinhamentos. Por um subconjunto de alinhamentos, entende-se um alinhamento de duas ou mais das *n* seqüências cujo alinhamento queremos obter. Um esquema de valoração intuitivo é calcular as valorações dos alinhamentos par a par definidos pelo alinhamento múltiplo, e simplesmente soma-los, para obter a valoração final do alinhamento múltiplo completo. Esta função é chamada Valoração da Soma de Pares (*Sum-of-pairs score*) e a chamaremos **SP *score*.** [CARRILLO H. & LIPMAN D., 1988].

Definição 2.2: SP *score*.

Seja ***A*** um alinhamento de ***n*** seqüências X^1, X^2, .. ,X^n onde S^1 # S^2 # ... # S^n são as seqüências alinhadas. Seja $\boldsymbol{C}(S^i,S^j)$ a valoração do alinhamento par das seqüências S^i e S^j. Então a valoração do alinhamento múltiplo é:

$$\boldsymbol{F_{SP}}(\boldsymbol{A}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}-1} \sum_{\boldsymbol{j}=\boldsymbol{i}+1}^{\boldsymbol{n}} \boldsymbol{C}(\boldsymbol{S}^{\boldsymbol{i}}, \boldsymbol{S}^{\boldsymbol{j}})$$

onde $\boldsymbol{C}$ é dada pela fórmula: $\boldsymbol{C}(\boldsymbol{S^i}, \boldsymbol{S^j}) = \sum_{\boldsymbol{k}=1}^{\boldsymbol{m}} \boldsymbol{P}_{s_k^i s_k^j}$

onde:

- m é o número de colunas no alinhamento.
- s_k^i é a k-ésima subunidade da seqüência S^i e P é a valoração dada por uma matriz de substituição, (seção 2.5).
- Se $s_k^i = s_k^j =$`–` então $\boldsymbol{P}_{s_k^i s_k^j} = 0$

A idéia por trás da função SP é que $\boldsymbol{C}(S^i,S^j)$ está baseado em um razoável esquema de valoração, e a melhor valoração deverá maximizar a soma de todas as valorações par a par. Estes pares de alinhamentos podem ter colunas com gaps, elas não têm significado para o alinhamento par a par e por isso sua valoração é zero.

2.5 Matrizes de Valoração de Subunidades

O esquema de valoração utilizado no exemplo 2.2 é muito simples para ser aplicado em um alinhamento real de seqüências de proteínas ou DNA. O principal objetivo de uma busca em uma base de dados é encontrar seqüências homólogas (ancestral comum) às seqüências consultadas. Por conseguinte, um esquema de valoração deve estar baseado na similaridade das subunidades que ocorrem nas seqüências. Para duas subunidades ($\boldsymbol{s_k^i}, \boldsymbol{s_k^j}$), precisamos calcular a probabilidade que eles tenham um ancestral comum, ou que um deles seja o resultado de uma ou varias mutações do outro.

Por esta razão foram desenvolvidas tabelas que relacionam os aminoácidos ou nucleotídeos conforme à similaridade entre eles. Essas tabelas são também chamadas matrizes de similaridade ou matrizes de substituição, elas estimam a taxa em que cada possível aminoácido (ou nucleotídeo) em uma seqüência transforma-se em outro aminoácido (ou nucleotídeo) sobre o tempo. No processo de evolução, desde uma geração à seguinte os aminoácidos de uma seqüência de proteínas de um organismo são gradualmente alterados através da ação de mutações de DNA.

Exemplo 2.3: Evolução de uma seqüência de aminoácidos.

A seqüência ALEIRYLRD

poderia mutar na seqüência ALEINYLRD em uma geração

e possivelmente em AQEINYQRD sobre um grande período de tempo evolucionário.

Cada aminoácido tem mais ou menos probabilidade de mutar em vários outros aminoácidos. Se tivermos duas seqüências de aminoácidos, devemos ser capazes de determinar a probabilidade de que eles derivem de um ancestral comum ou homólogo. Então, construímos uma matriz 20x20 onde a entrada (i,j) é igual à probabilidade do *i-ésimo* aminoácido ser transformado no *j-ésimo* aminoácido em certo tempo evolucionário. Existem muitas maneiras diferentes de construir esta matriz, chamada matriz de substituição. A seguir descrevemos as mais comumente usadas para seqüências de proteínas.

2.5.1 Matrizes PAM

Uma das primeiras matrizes de substituição de aminoácidos, a matriz PAM (*Point Accepted Mutation*) foi desenvolvida por Margareth Dayhoff em 1978 [DAYHOFF et al., 1978]. Esta matriz é calculada observando diferenças em proteínas proximamente relacionadas. A matriz PAM1 estima que taxa de substituição deverá ser esperada se 1% dos aminoácidos muda. A matriz PAM1 é usada como base para calcular outras matrizes sobre a hipótese que mutações repetidas deveram seguir o mesmo patrão que PAM1, e múltiplas substituições podem ocorrer no mesmo sítio. Usando esta lógica, Dayhoff derivou o PAM250 (Tabela 2.2), o número indica a unidade evolucionária, que é obtida fazendo 250 multiplicações de PAM1.

2.5.2 Matriz BLOSUM

A metodologia de Dayhoff comparando espécies proximamente relacionadas não é adequada para trabalhar com alinhamentos de seqüências divergentes (seqüências de relacionamento distante). Mudanças de seqüências sobre escalas de tempo evolucionarias longas não são bem aproximadas comparando com mudanças pequenas que ocorrem sobre escalas de tempo curtas. A série de matrizes BLOSUM (BLOck SUbstitution Matrix) resolve este problema [HENIKOFF & HENIKOFF, 1992].

Henikoff e Henikoff construíram matrizes usando alinhamentos múltiplos de proteínas evolucionariamente divergentes. A matriz BLOSUM62 (Tabela 2.1) é calculada sobre substituições observadas entre proteínas que compartilham 62% ou menos de identidade. As matrizes BLOSUM com numeração mais alta são indicadas para alinhar seqüências proximamente relacionadas e aquelas com numeração mais baixa para seqüências mais divergentes.

2.5.3 Diferenças entre PAM e BLOSUM

1. Matrizes PAM são baseadas sobre um modelo evolucionário explícito (por ex. substituições são contadas sobre uma arvore filogenética), no entanto matrizes BLOSUM estão baseadas sobre um modelo implícito de evolução.
2. Matrizes PAM estão baseadas sobre mutações obtidas através de um alinhamento global, este inclui tanto regiões altamente conservadas como aquelas altamente mutáveis. As matrizes BLOSUM estão baseadas sobre regiões altamente conservadas em blocos de alinhamentos que não contêm gaps.
3. O método usado para contar as substituições também é diferente. Ao contrario que na matriz PAM, o procedimento da BLOSUM usa grupos de seqüências dentro do qual não todas as mutações são consideradas com o mesmo valor de contagem.

Matrizes PAM nomeadas com numeração mais alta denotam um esquema de distância evolucionária maior, por enquanto matrizes BLOSUM nomeadas com numeração maior denotam um esquema de similaridade de seqüências mais alta e conseqüentemente uma distancia evolucionaria menor. Por ex. PAM150 é usada para seqüências mais distantes que PAM100; BLOSUM62 é usada para seqüências mais proximamente relacionadas que BLOSUM50.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	-1	4																		
T	-1	1	5																	
P	-3	-1	-1	7																
A	0	1	0	-1	4															
G	-3	0	-2	-2	0	6														
N	-3	1	0	-2	-2	0	6													
D	-3	0	-1	-1	-2	-1	1	6												
E	-4	0	-1	-1	-1	-2	0	2	5											
Q	-3	0	-1	-1	-1	-2	0	0	2	5										
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8									
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5								
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5							
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5						
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4					
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4				
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	2			
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Tabela 2.1: Matriz BLOSUM 62

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	12																			
S	0	2																		
T	-2	1	3																	
P	-3	1	0	6																
A	-2	1	1	1	2															
G	-3	1	0	-1	1	5														
N	-4	1	0	-1	0	0	2													
D	-5	0	0	-1	0	1	2	4												
E	-5	0	0	-1	0	0	1	3	4											
Q	-5	-1	-1	0	0	-1	1	2	2	4										
H	-3	-1	-1	0	-1	-2	2	1	1	3	5									
R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	5								
K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5							
M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6						
I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5					
L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6				
V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4			
F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9		
Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10	
W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Tabela 2.2: Matriz PAM 250

2.6 Penalidade dos Gaps

Um gap pode ser o resultado de uma ou várias mutações (inserções e deleções). Conseqüentemente, escolher como penalizar os gaps é uma tarefa que implica muita dificuldade. Usualmente, uma forma de penalidade local é utilizada, a qual determina que a penalidade de um gap seja encontrada independentemente de outros gaps no alinhamento, este tipo de gap é chamado ***gap linear***. Então, aquele alinhamento com menor número de gaps será menos penalizado. A função da penalidade de gap linear é g_l = G x l onde l é a quantidade de gaps e g_l é a função de penalidade de gap.

Biologicamente, estender um gap deverá penalizar menos que abrir um, por tanto diferenciamos dois tipos de gaps, gap de abertura e gap de extensão:

Gap de abertura (g_a): é a penalidade considerada para o primeiro gap que segue a uma subunidade.

Gap de extensão (g_e): é a penalidade considerada para o gap que segue a outro gap.

Por conseguinte, uma fórmula melhor para a penalidade de gap é usar uma função que calcule a penalidade destes tipos de gaps, ela é chamada de ***penalidade de gap afim*** [Altschul, S.F., 1989], sua formula é $g_l = g_a + l$ x g_e ou $g_l = g_a + (l\text{-}1)$ x g_e.

Alguns estudos argumentam que a penalidade para estender gaps deverá decrescer com o comprimento, esta função é chamada penalidade de ***gap logarítmica*** e sua formula é $g_l = g_o + \log l$.

A seguir mostramos dois exemplos, primeiro exemplificamos em um alinhamento o gap de abertura e o gap de extensão e segundo calculamos a Função SP *score* usando penalidade de gap afim.

Exemplo 2.4: Dadas duas seqüências X^1: NHCG e X^2: NCKCG e o seguinte alinhamento entre elas S^1#S^2:

	1	2	3	4	5	6
S^1:	N	H	-	-	C	G
S^2:	N	C	K	-	C	G

Figura 2.2: Alinhamento de duas seqüências

Na coluna 3 aparece um gap de abertura em S^1, já que a subunidade que a precede não é gap. Na coluna 4 aparece um gap de extensão em S^1, já que a subunidade que a precede é um gap e em S^2 aparece um gap de abertura na mesma coluna.

Exemplo 2.5: Cálculo de $F_{SP}(A)$ do alinhamento da figura 2.2, com a matriz BLOSUM62 e penalidade de gap afim.

	1	2	3	4	5	6
S^1:	N	H	-	-	C	G
S^2:	N	C	K	-	C	G
P:	6	-3	$-g_a$	0	9	6

$$C(S^1, S^2) = 18 - g_a$$

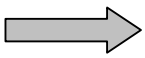
A valoração para as subunidades na primeira coluna é dada por $P_{NN} = 6$ e na segunda coluna por $P_{HC} = -3$. Na terceira coluna apresentamos um gap de abertura em S^1, portanto a pontuação é $P_{gaK} = g_a$. Na quarta coluna apresentamos um gap de extensão em S^1 e em S^2 um gap de abertura, então $P_{gega} = 0$.

Na quinta e sexta colunas, as valorações são $P_{CC} = 9$ e $P_{GG} = 6$. Estas valorações são somadas para obter $C(S^1, S^2)$. Seja $g_a = -1$ então $C(S^1, S^2) = 18$.

A seguir descrevemos um exemplo onde mostramos a avaliação da qualidade de um alinhamento múltiplo de três seqüências, utilizando a matriz de substituição PAM250 e penalidade de gap afim.

Exemplo 2.6: Alinhamento múltiplo de seqüências calculado com SP.

X^1: MHCG
X^2: CKKT
X^3: MCKCG



S^1: M H C - - G
S^2: - - C K K T
S^3: M - C K C G

Parâmetros utilizados: Tabela 2.2 com gap abertura $g_a = 10$ e gap extensão $g_e = 0.5$

S1	M	H	C	-	-	G
S2	-	-	C	K	K	T
P	-10	-0.5	12	-10	-0.5	0

$C(S^1, S^2) = -9$

S1	M	H	C	-	-	G	
S3	M	-	C	K	C	G	
P	6	-10	12	-10	-0.5	5	$C(S^1,S^3) = 2.5$

S2	-	-	C	K	K	T	
S3	M	-	C	K	C	G	
P	-10	0	12	5	-5	0	$C(S^2,S^3) = 2$

$\mathbf{F_{SP}(A)} = C(S^1,S^2) + C(S^1,S^3) + C(S^2,S^3) = -4.5$

2.7 Função Soma Ponderada de Pares de Seqüências.

Há um inconveniente com a função soma de pares (**SP**): todas as seqüências são ponderadas igualmente. Existem razões biológicas para considerar que as seqüências têm diferentes ponderações. As seqüências para as quais estamos procurando um alinhamento múltiplo usualmente constituem uma família ou subfamílias, elas podem dar um retrato distorcido da família, como também podem ser muitas de um "tipo" de família e poucas de outro "tipo". Se considerarmos a hipótese que a descrição deverá retratar uma distribuição uniforme de seqüências, esta representação distorcida poderia ser corrigida dando as ponderações mais altas às seqüências isoladas e as ponderações mais baixas àquelas que têm muitas seqüências similares conhecidas na família. Então a função SP toma a forma:

$$F_{WSP}(A) = \sum_{i=1}^{n-1} \sum_{j-i+1}^{n} C(S^i, S^j) w^i w^j$$

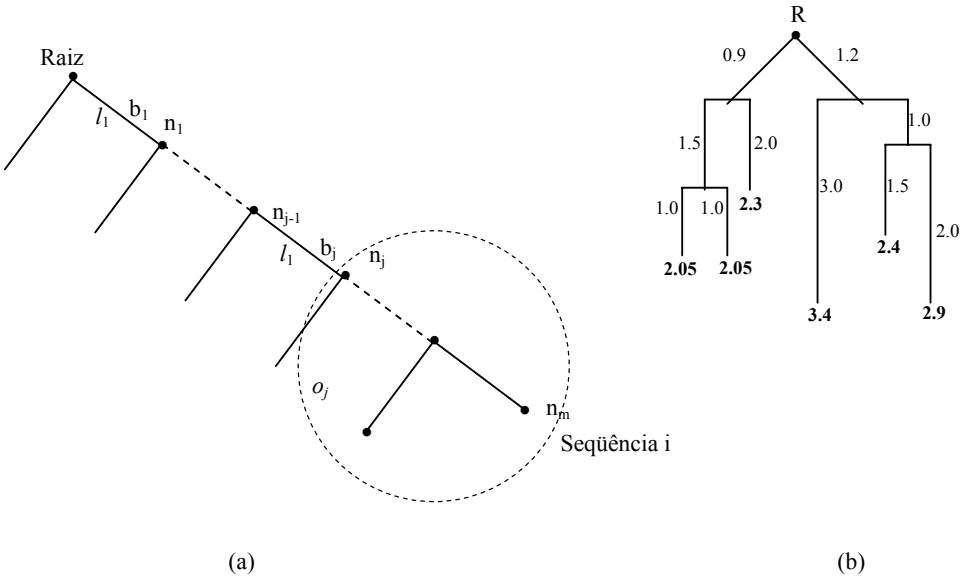
Onde w^i é a ponderação da seqüência *i*.

Ponderações de Seqüências

Descrevemos um método para calcular as ponderações nas seqüências usando a aproximação utilizada pelo software de alinhamento CLUSTAL [THOMPSON et al., 1994], software muito popular para alinhamento múltiplo de seqüências.

Um esquema de ponderação pode ser definido usando uma árvore guia construída com as seqüências consideradas. O que devemos calcular é um número que

represente a divergência destas seqüências. As seqüências são representadas pelos nós da arvore e estes podem ser etiquetados nos lados (linha que une dois nós) pelas estimativas do número de mutações que ocorreram ao longo de cada lado. A Figura 2.3 mostra a árvore.



(a) (b)

Figura 2.3: (a) Parte de uma árvore guia ilustrando como pesos de seqüências são calculados. O circulo em linha de pontos ilustra a subárvore com raiz n_j. (b) Os pesos de cada uma das folhas de uma arvore exemplo, calculada desde as etiquetas sobre os lados.

A árvore pode ser usada para calcular os pesos:

- Os pesos devem crescer com o aumento da diferença (número de mutações) à raiz (a distância para a 'metade da seqüência').
- Os pesos devem diminuir com o incremento do número de seqüências na vizinhança (seqüências similares).

Para descrever como isto poder ser feito introduzimos algumas variáveis, ilustradas na Figura 2.3(a).

Sejam

- $b_1, b_2, \ldots, b_n$ os lados sobre o caminho desde a raiz a uma seqüência i;
- l_j o valor (número de mutações) sobre o lado b_j;

- m_j o nó conectado ao lado b_j mais distante da raiz;
- o_j o número de seqüências em uma subárvore com raiz no nó n_j (o_n é conseqüentemente 1).

Uma expressão para o peso da seqüência i, pode ser:

$$w_i = \sum_{j=1}^{n} \frac{l_j}{o_j}$$

Isto significa que o peso de uma seqüência é a soma das razões dos pesos (valores) sobre os lados desde a raiz à folha representando a seqüência. O peso de cada lado é igualmente compartilhado entre as seqüências sendo deixadas sobre a subarvore abaixo do lado.

A figura 2.3(b) mostra os pesos que são calculados baseados na árvore. w_i pode ser normalizado, tal que o mais alto peso é, por exemplo 100.

2.8 Função COFFEE

A fim de incrementar a precisão do alinhamento múltiplo de seqüências, Notredame et al. propuseram uma estratégia para otimizar MSA por meio de um algoritmo genético e usando uma função de consistência para a comparação das seqüências, eles chamaram esta função de COFFEE (*Consistency based Objective Function For alignmEnt Evaluation*). Esta função reflete o nível de consistência entre um alinhamento múltiplo de seqüências e uma biblioteca que contém alinhamentos par a par das mesmas seqüências [NOTREDAME et al., 1998].

A idéia é gerar um conjunto de alinhamentos par a par e procurar por consistência entre estes alinhamentos, em outros termos, dado um conjunto de seqüências gerar uma coleção "todos-contra-todos" de alinhamentos pares destas seqüências (biblioteca), a Figura 2.4 mostra um exemplo. A valoração de um alinhamento múltiplo de seqüências é definida como uma medida de sua consistência com a biblioteca. Esta função requer dois componentes: (i) um conjunto de alinhamentos de referência par a par (biblioteca), (ii) a função objetivo para avaliar a consistência entre um alinhamento múltiplo e um alinhamento par contido na biblioteca.

A criação da biblioteca pode ser construída a partir do software CLUSTAL W (www.ebi.ac.uk/clustalw/) usando os parâmetros providos pelo programa. A avaliação é feita comparando cada par de subunidades alinhadas observadas no alinhamento múltiplo que estão presentes na biblioteca (ex. duas bases alinhadas um com outro ou uma base alinhada com um gap). Em cada comparação, subunidades são identificadas por sua posição na seqüência (os gaps não são diferenciados). A pontuação da consistência total é igual ao número de pares de subunidades presentes no alinhamento múltiplo que são também encontrados na biblioteca, dividido pelo número total de pares observados no alinhamento múltiplo de seqüências.

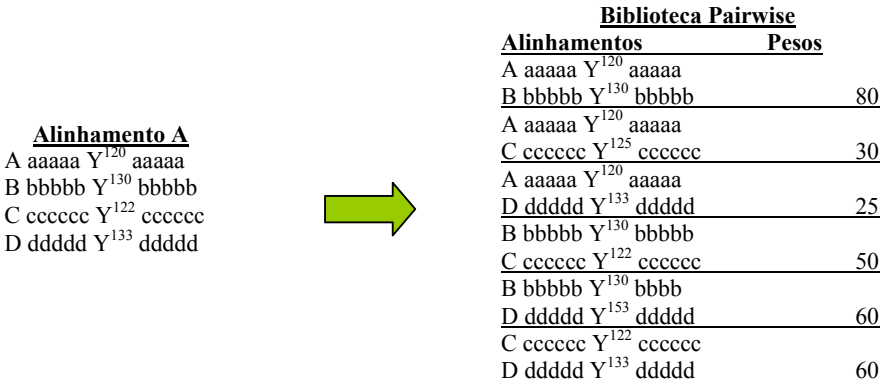


Figura 2.4: Biblioteca par a par do alinhamento A

A função COFFEE pode ser formalizada como segue. Dadas *N* seqüências não alinhadas $X^1...X^N$ em um alinhamento múltiplo, $\boldsymbol{S^i \# S^j}$ é um alinhamento par a par das seqüências X^i e X^j (obtido desde o alinhamento múltiplo), LEN($\boldsymbol{S^i \# S^j}$) é o comprimento de este alinhamento, SCORE($\boldsymbol{S^i \# S^j}$) é a consistência total (nível de identidade) entre $\boldsymbol{S^i \# S^j}$ e o alinhamento correspondente par a par na biblioteca (é melhor definida posteriormente), e $W_{i,j}$ é o peso associado a este alinhamento par.

$$\text{COFFEE } score = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{i,j}\,\boldsymbol{SCORE}(S^i \# S^j)}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{i,j}\,\boldsymbol{LEN}(S^i \# S^j)}$$

SCORE($\boldsymbol{S^i \# S^j}$)= número de pares alinhados de resíduos que são compartilhados entre $\boldsymbol{S^i \# S^j}$ e a biblioteca.

As principais diferenças de COFFEE com respeito a SP são: (i) não são aplicadas penalidades de gap extra, já que esta informação já está contida na biblioteca (ii) a pontuação COFFEE é normalizada pelo valor de pontuação máxima e (iii) o custo das substituições depende da posição, graças à biblioteca (ex. dois pares de subunidades similares terão potencialmente diferentes valorações se o índice das subunidades é diferente). A Figura 2.5 mostra um exemplo.

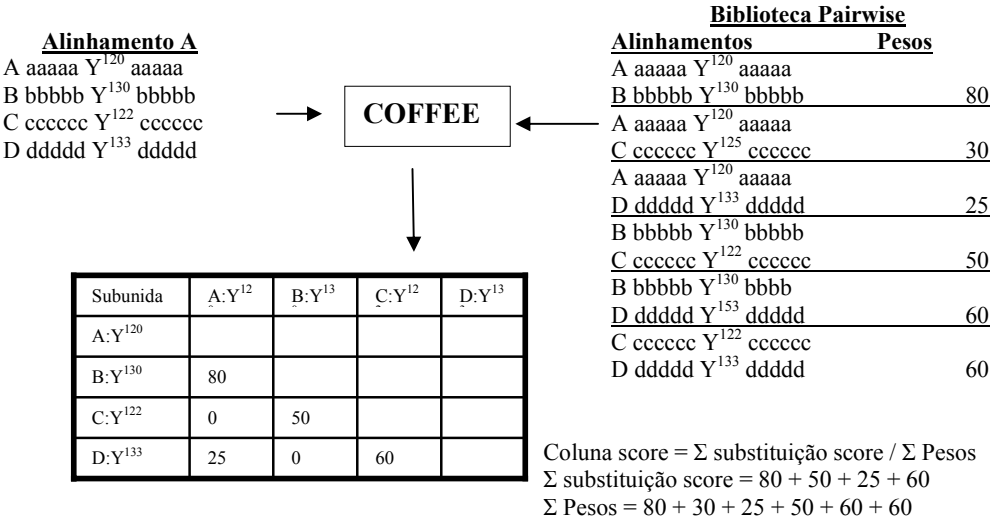


Figura 2.5: Função COFFEE. O cálculo da coluna Y onde o sobreíndice representa uma subunidade se realiza comparando cada par de seqüências alinhadas com a biblioteca, então se as subunidades estão presentes na biblioteca o peso é copiado na matriz na cela que intersecta as seqüências comparadas, senão estiver presente então a valoração é zero como nos alinhamentos das seqüências A,C e B,D.

Logo todos os valores na matriz são somados e por outro lado também somamos todos os pesos na biblioteca, então a valoração total da coluna Y é obtida dividindo esses dois valores.

A valoração total do alinhamento é dada por : Σ coluna score / Σcomprimento do alinhamento

2.9 Heurísticas mais utilizadas em MSA

2.9.1 Programação dinâmica

Este método é baseado em um paradigma geral chamado programação dinâmica, foi proposto por Needleman e Wunsch em 1970. A tarefa de encontrar a mais alta valoração de um alinhamento é feito em dois passos:

1. Usando programação dinâmica, encontrar a valoração mais alta possível.
2. Encontrar (um, vários ou todos) alinhamentos com a valoração mais alta usando os resultados intermediários desde o primeiro passo.

A continuação descrevemos os passos do processo para um alinhamento de duas seqüências.

Sejam duas seqüências X^1 = a c g c e X^2 = c a t g, cujos comprimentos são *m* e *n* respectivamente, calculamos a valoração do alinhamento entre as mesmas.

Para facilitar o processo de alinhamento, utilizamos uma matriz bi-dimensional **H** de tamanho (*m* + 1)x(*n* + 1) como se mostra na Figura 2.6.

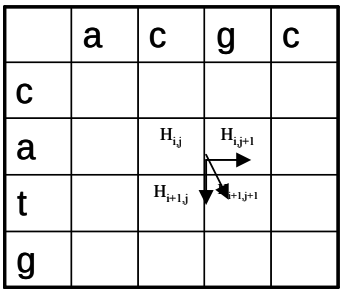


Figura 2.6 : Programação Dinâmica. As setas mostram as celas cujo valor foi calculado mais cedo, estes valores são utilizados para calcular o valor da cela $H_{i,j}$

A matriz é preenchida linha a linha desde o canto inferior direito até o canto superior esquerdo ou seja desde a célula $\boldsymbol{H}_{m,n}$ até a célula $\boldsymbol{H}_{0,0}$. Ao tentar alinhar sucessivamente as bases das seqüências, o algoritmo atribui uma valoração (*score*) para cada uma das bases que compõem as seqüências. A valoração é computada de forma a penalizar as diferenças entre as bases e privilegiar as similaridades.

Consideramos um esquema de valoração simples neste exemplo:

- $\boldsymbol{P}_{x_k^i x_k^j} = 1$ **se** $\boldsymbol{x_k^i = x_k^j}$ **e** $\boldsymbol{P}_{x_k^i x_k^j} = 0$ **se** $\boldsymbol{x_k^i \neq x_k^j}$.

A valoração da cela $H_{i,j}$ é dada pela seguinte formula:

$$\boldsymbol{H}_{i,j} = \boldsymbol{P}_{x_i^1, x_j^2} + \max \begin{cases} \boldsymbol{P}_{x_{i+1}^1, x_{j+1}^2} \\ [\boldsymbol{H}_{i+1:j+2,n}] \\ [\boldsymbol{H}_{j+1:i+2,m}] \end{cases}$$

Onde $[\boldsymbol{H}_{i+1:j+2,n}]$representa os valores contidos nas celas na linha *i*+1 desde a coluna *j*+2 a *n*, e $[\boldsymbol{H}_{j+1:i+2,m}]$ representa os valores contidos nas celas na coluna j+1 desde a linha i+2 até *m*.

Por exemplo o valor da cela $\boldsymbol{H}_{2,0}$ será (Figura 2.7):

$H_{2,0} = \boldsymbol{P}_{x_i^1, x_j^2} + \max \{1,[0,0],[0]\} = 1 + 1 = 2$

	a	c	g	c
c	1	2	0	1
a	2	1	0	0
t	1	1	0	0
g	0	0	1	0

Figura 2.7: Cálculo da cela $H_{2,0}$

Uma vez que a matriz é completamente preenchida procede-se à construção do alinhamento. Percorremos a matriz a partir da cela $H_{0,0}$ até a última linha ou coluna da seguinte forma (Figura 2.8):

	a	c	g	c
c	1 →	2	0	1
a	2	1 ↓	0	0
t	1	1 ↓	0	0
g	0	0	1 ↘	0

Figura 2.8: Caminho que indica um alinhamento ótimo

O avanço realiza-se comparando os valores das celas vizinhas da direita, embaixo e na diagonal de $H_{i,j}$ estes são $H_{i,j+1}$, $H_{i+1,j}$ e $H_{i+1,j+1}$ e escolhendo o maior valor entre eles. Em caso de igualdade entre os três valores escolher sempre o elemento da diagonal inferior $H_{i+1, j+1}$. Na figura 2.8 as setas mostram o caminho escolhido.

Então o alinhamento é construído, escrevendo-se uma seqüência cima da outra de forma que um gap é inserido na primeira seqüência quando a seta pula uma coluna é na segunda seqüência quando a seta pula uma linha.

No caso do exemplo, o alinhamento resultante será:

a c - - g c
- c a t g -

Outro alinhamento resultante é:

	a	c	g	c
c	1	2	0	1
a	2	1	0	0
t	1	1	0	0
g	0	0	1	0

- a c g c
c a t g -

2.9.2 Alinhamento Progressivo

O problema de alinhar um número "*n*" maior que 3 quaisquer de seqüências é um problema NP-completo se for utilizado o princípio do algoritmo de programação dinâmica. Então foi preciso desenvolver outro método para alinhar múltiplas seqüências, a primeira aproximação útil foi desenvolvida por D. Sankoff em 1983, baseada em filogenética [SANKOFF D., 1983].

Baseando-se na aproximação de Sankoff, o alinhamento múltiplo progressivo é uma abordagem para o alinhamento de várias seqüências que tenta inferir uma evolução entre as espécies sendo comparadas. Podemos dividi-lho em 3 fases principais:

Alinhamento par a par: Nesta fase é construída uma matriz de distâncias dos alinhamentos de todos os possíveis pares do conjunto de seqüências de entrada utilizando-se o algoritmo de programação dinâmica (Figura 2.9).

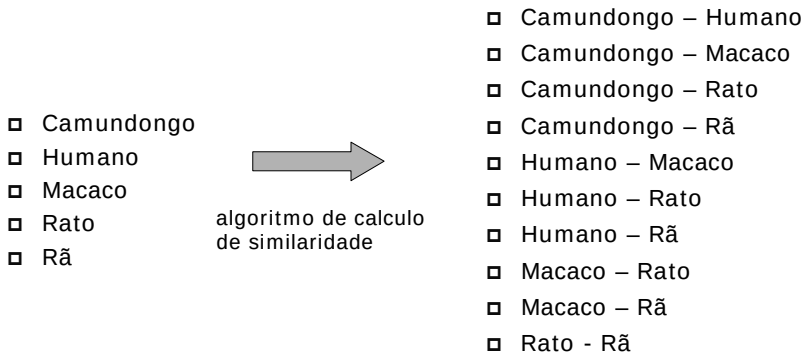


Figura 2.9: Alinhamento par a par

Construção da árvore guia: Obtida a matriz de distâncias, cria-se uma árvore guia que mais tarde conduzirá o alinhamento múltiplo propriamente dito. Existem vários algoritmos utilizados na construção da árvore guia, o algoritmo *neighbor-joining* é um deles.

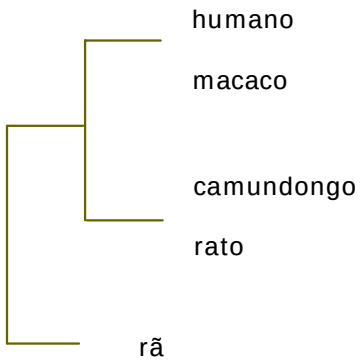


Figura 2.10: Árvore guia.

Alinhamento Progressivo: O alinhamento progressivo produz o alinhamento múltiplo propriamente dito a partir da árvore guia criada. O alinhamento é feito progressivamente, de forma que as seqüências em ramos mais próximos sejam alinhadas primeiramente.

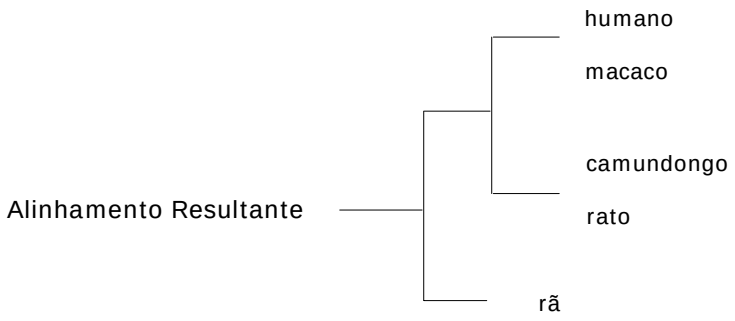


Figura 2.11: Alinhamento progressivo propriamente dito.

Alguns softwares baseados neste método são: Multialign [BARTONT & STERNBERG, 1987], Multal [TAYLOR, 1988], Pileup (*Wisconsin Package v.10 Genetic Computer Group*) e ClustalX [THOMPSON et al., 1997] e com a interface gráfica ClustalW [THOMPSON et al., 1994].

Há algumas questões que devem ser consideradas sobre este método de alinhamento múltiplo.

- Sempre retorna uma resposta, independentemente se ela for certa ou errada.
- As respostas deste método são em geral boas e próximas da solução desejada.
- Não investiga todas as possibilidades de alinhamentos exaustivamente.
- É muito comum que seja necessário fazer correções ou ajustes manuais para melhorar o alinhamento.
- Dependência do alinhamento inicial par- par.
- Propagação de erros do alinhamento inicial.

2.9.3 Métodos Iterativos

Podem ser determinísticos ou estocásticos, dependendo da estratégia usada para melhorar o alinhamento. Permitem boa separação conceitual entre o processo de otimização e a Função objetivo (FO).

São muito lentos iniciando a partir de um conjunto de seqüências não alinhadas, por tanto é utilizado com mais sucesso para refinar e melhorar um conjunto alinhado de seqüências como conjunto inicial.

Exemplos: PRRP program [GOTOH O., 1996], Hmmt program [EDDY, S.R., 1995], programas baseados em algoritmos genéticos, SAGA [NOTREDAME, C. AND HIGGINS, D. G., 1996].

Cadeia de Markov

Outra alternativa é usar modelos de Cadeia de Markov (HMMs) [BALDI et al, 1994; KROGH et al., 1994; EDDY, S.R., 1995], que tenta encontrar simultaneamente um modelo de alinhamento e probabilidade de substituições, inserções e deleções que são auto consistentes.

Esta é uma aproximação limitada a casos de muitas seqüências (ex. 100 ou mais) mas têm a vantagem de uma sólida análise probabilística.

Algoritmos Genéticos

AGs são um conjunto de algoritmos estocásticos para busca eficiente e robusta. O problema de alinhar seqüências de biomoléculas pode ser convertido em uma busca para pontos ótimos no espaço determinado pelo problema. Assim um AG pode ser utilizado para alinhamento de seqüências. Os resultados experimentais indicam que AGs poderiam prover uma abordagem eficiente e precisa para comparação e análise de seqüências de biomoléculas [ZHANG, 1997; ISOKAWA et al., 1996; NOTREDAME, C. & HIGGINS, D. G., 1996].

Os AG simulam o processo evolutivo biológico aplicando operações genéticas sobre cromossomas [Apêndice B], selecionam a partir de uma população evoluída o alinhamento que otimiza uma função objetivo FO, é clave incluir uma apropriada representação do alinhamento de seqüências de biológicas, e uma apropriada aplicação dos três operadores genéticos. Uma popular representação consiste de um string de ‘0’s e ‘1’s, onde os ‘0’s representam os gaps e os ‘1’s representam as bases que compõem as seqüências de biomoléculas. O fluxo de um AG é mostrado na Figura 2.12.

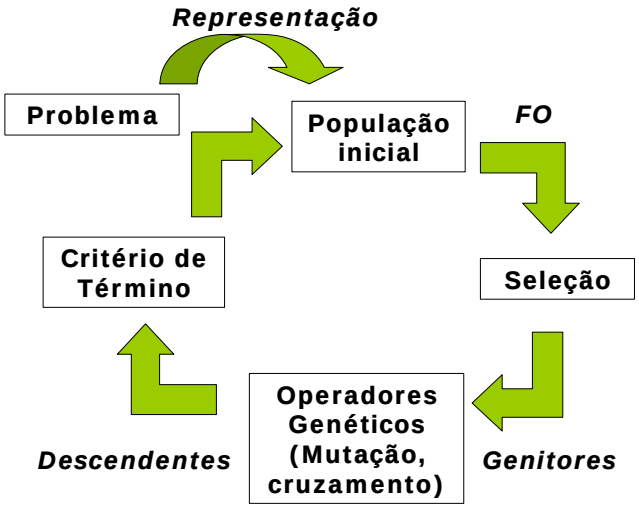


Figura 2.12: Fluxo de um Algoritmo Genético

Esta abordagem tem como desvantagem a baixa velocidade de cálculo em comparação com as heurísticas progressivas, mas são capazes de encontrar um ótimo alinhamento múltiplo global em tempo razoável, iniciando desde seqüências completamente não alinhadas obtendo alinhamentos resultantes muito consistentes, pode ser usada para otimizar qualquer função objetivo.

2.10 Métrica de Performance: BAliBASE.

Uma avaliação e comparação compreensiva de programas de alinhamentos requere um grande número de alinhamentos de referência precisos que podem ser usados como casos de testes.

BAliBASE [THOMPSON *et al.*, 1999] é uma base de dados de alinhamentos múltiplos de seqüências de proteínas. Ele contém alinhamentos de razoável nível de qualidade, que estão bem documentados e foram construídos usando uma variedade de programas e verificação manual.

A atual versão de BAliBASE contêm 143 alinhamentos de referência, com um total de mais de 1000 seqüências. BAliBASE é dividido em 5 categorias diferentes,

considerando o comprimento e similaridade de seqüências nos blocos centrais do alinhamento e a presença de inserções e extensões terminais.

Categoria 1: contêm mais que 80 alinhamentos de seqüências de similar comprimento. Cada alinhamento têm um pequeno número (3-7) de seqüências.

Categoria 2: tem alinhamentos de seqüências mais divergentes que contêm múltiplos subgrupos com menos de 25% de identidade de resíduos (elementos das seqüências de proteínas) entre grupos.

Categoria 3: contêm famílias de seqüências divergentes, eqüidistantes dentro de um simples alinhamento.

Categoria 4 e 5: contêm seqüências com extensões terminais (acima de 400 resíduos). Este é o único subconjunto que favorece alinhamentos locais sobre globais.

Procedimento de Avaliação baseado em BAliBASE.

Para determinar a similaridade do alinhamento obtido por um programa qualquer e o alinhamento de referência em BAliBASE, calculamos as valorações do alinhamento obtido: Sum-of-Pair (SPS) e Column Score (CS).

O ***SPS*** score conta os pares de subunidades corretamente alinhados.

Para duas seqüências S^i e S^j, valoramos com 2 pontos a cada par de subunidades $\boldsymbol{s_k^i}$ e $\boldsymbol{s_k^j}$ que estão alinhados um com outro tanto no alinhamento testado como no alinhamento de referência. Também valoramos com 1 ponto cada subunidade que é alinhada com um gap tanto no alinhamento testado como no alinhamento de referência.

A valoração total é normalizada pela máxima valoração possível, de 0 a 1. O valor 1 indica que o alinhamento múltiplo é idêntico ao benchmark.

Formalmente, em um alinhamento de N seqüências de comprimento M, seja P_{kij} a valoração da coluna k nas seqüências S^i e S^j, se as seqüências S^i e S^j são alinhadas da mesma forma como no alinhamento de referência, então o valor par P_{kij} é positivo. Se, na coluna k, S^i e S^j têm subunidades alinhadas no alinhamento de referência, então P_{kij} =2; se uma das seqüências tem um gap (em ambos alinhamentos), então P_{kij} =1, de outra forma P_{ixy} = 0. A valoração $\boldsymbol{S_k}$ para a *k-ésima* coluna é:

$$S_k = \sum_{j=1}^{N} \sum_{i \neq j} P_{kij}$$

O SPS é então:

$$SPS = \sum_{k=1}^{M} S_k / \sum_{k=1}^{M_r} S_{rk}$$

Onde M_r é o número de colunas no alinhamento de referência e S_{rk} é a valoração S_k para a *k-ésima* coluna no alinhamento de referência.

O ***CS*** conta o número de colunas no alinhamento de referência que são alinhadas corretamente em todas as seqüências, normalizado pelo número de colunas alinhadas. Formalmente, em cada posição k do alinhamento, C_k = 1 se todas as seqüências são alinhadas da mesma forma como no alinhamento de referência, de outra forma C_k = 0. Assim, ***CS*** do alinhamento total é:

$$CS = \sum_{k=1}^{M_c} C_k / M_c$$

Onde M_c é o número total de colunas no alinhamento de referência. Nenhuma coluna é ignorada em nosso cálculo (exceto as colunas contendo todos gaps, que não contêm informação acerca do alinhamento das seqüências)

2.11 Considerações Gerais

Alinhamentos múltiplos ótimos são difíceis de construir. Primeiro de tudo, é difícil avaliar a qualidade do alinhamento múltiplo. Segundo, quando temos uma função disponível para avaliação, muitas vezes ele é algoritmicamente muito custoso ou se torna em um problema NP-completo para produzir alinhamentos tendo a melhor valoração possível (ótimo alinhamento) [WANG & JIANG, 1994; DELLA VEDOVA & BONIZZONI, 2001]. Como é o caso da programação dinâmica para um número superior a três seqüências

As funções de custo ou funções de valoração geralmente dependem de uma matriz de substituição. As matrizes de substituição usualmente são estabelecidas pela análise estatística de um grande número de alinhamentos, mas estes podem não ser

necessariamente adaptados ao conjunto de seqüências de nosso interesse. Esta é a principal limitação destes esquemas de valoração. Qualquer seja o esquema de valoração que se deseje usar, o problema de otimização pode ser muito difícil, erros cometidos no inicio do procedimento nunca são corrigidos e podem cair em desalinhamentos, sobre tudo em estratégias progressivas.

De forma a evitar isto, é necessário um esquema que filtre a informação inicial e permita seu uso global, a função de consistência tenta fazer isso. O principio básico é gerar um conjunto de alinhamentos par a par e olhar pela consistência entre estes alinhamentos. Deste modo define o alinhamento múltiplo ótimo como o mais consistente.

A tendência atual é o incremento do uso de estratégias iterativas (estocásticos e não estocásticos) e o uso de esquemas de valoração baseados em consistência. T-COFFEE e métodos HMM.

CAPÍTULO

3

INTRODUÇÃO À OTIMIZAÇÃO MULTI-OBJETIVO

No capítulo precedente descrevemos os conceitos necessários para entender o problema do alinhamento múltiplo de seqüências de biológicas. Neste capítulo fazemos uma introdução da Otimização Multi-objetivo, baseado nos conceitos expostos por DEB [DEB, 2001] e COELLO [COELLO, 1999]. Ao final do capítulo descrevemos os Algoritmos Evolutivos Multi-Objetivo que implementamos nesta pesquisa.

3.1 Otimização Multi-Objetivo e Otimização Simples.

No mundo real, a maior parte dos problemas tem vários objetivos (possivelmente conflitantes) que se deseja sejam satisfeitos de forma simultânea. Muitos destes problemas freqüentemente são convertidos em mono-objetivo transformando todos os objetivos originais, e considerando algumas restrições adicionais.

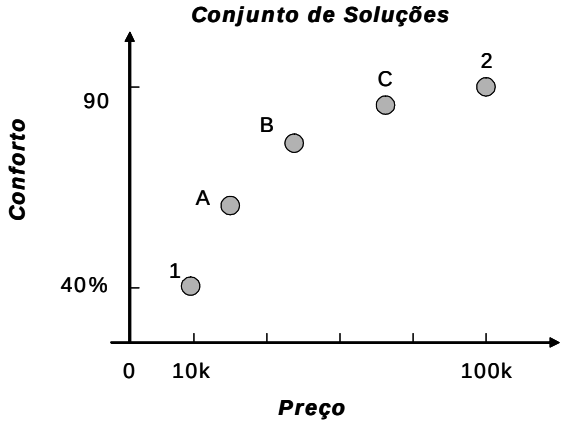


Figura 3.1: Soluções hipotéticas para o problema de tomada de decisão de compra de um carro.

Quando temos duas funções objetivo, cada objetivo corresponde a uma solução ótima diferente. No problema de tomada de decisões para a compra de um carro mostrado na Figura 3.1, os pontos 1 e 2 representam estas soluções ótimas. Neste caso um dos objetivos é sacrificado (no caso da solução 1 é sacrificado o conforto e no caso da solução 2 é sacrificado o preço). O exemplo mostra um conjunto de soluções ótimas (tal como 1, 2, A, B e C) onde o ganho em um objetivo sacrifica o outro objetivo. Nenhuma destas soluções é melhor com respeito a ambos objetivos. Por conseguinte, em problemas com mais de um objetivo conflitante, não existe uma única solução ótima. Sem qualquer adicional informação, nenhuma solução desde o conjunto de soluções ótimas pode ser dito "melhor" que qualquer outra.

Duas abordagens para Otimização Multi-objetivo.

Em um Problema de Otimização Multi-Objetivo (MOP) o conjunto de soluções ótimas é muito importante. Esta é a diferencia fundamental entre uma tarefa de otimização de objetivo simples e de uma tarefa de objetivo múltiplo.

Depois de encontrar um conjunto de soluções ótimas para um problema particular, o usuário pode então utilizar considerações qualitativas de alto nível para a tomada de decisão. DEB sugere um principio para um procedimento de otimização multi-objetivo ideal [DEB, 2001], este é o seguinte:

Passo 1: Encontrar múltiplas soluções ótimas.

Passo 2: Escolher uma das soluções obtidas utilizando informação de alto nível.

O esquema é mostrado na Figura 3.2

Este procedimento mostra que otimização de objetivo simples é um caso degenerado de otimização multi-objetivo.

Fazendo uma análise, cada solução corresponde a uma ordem específica de importância de objetivos. Por exemplo, na Figura 3.1 a solução A da mais importância ao preço que ao conforto. Por outro lado, a solução C da mais importância ao conforto que ao preço.

Assim se conhecemos o fator de preferência entre os objetivos para um problema específico, não precisamos seguir o principio anterior para a resolução de um MOP. Um método simples deverá formar uma FO composta como a soma ponderada

dos objetivos, onde um peso para cada objetivo é proporcional ao fator de preferência determinado a um objetivo particular. Este método de escalar um vetor objetivo em uma FO simples composta converte o MOP em um problema de otimização de objetivo simples. Quando a FO composta é otimizada, na maioria dos casos é possível obter uma solução particular. Este procedimento é muito mais simples.

Chamamos a este procedimento otimização multi-objetivo baseado em preferência. Um esquema deste procedimento é mostrado na Figura 3.3. As soluções obtidas usando esta estratégia são sensíveis ao vetor de preferência relativo usado para formar a função composta. Qualquer mudança neste vetor mudará a solução.

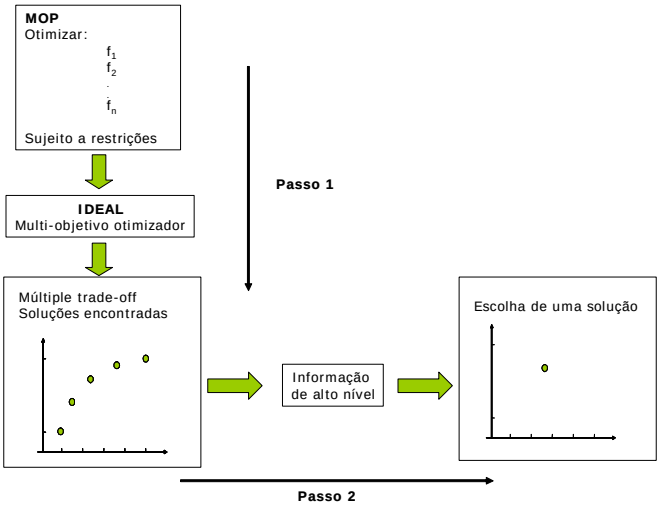


Figura 3.2: Esquema do procedimento de otimização multi-objetivo ideal.

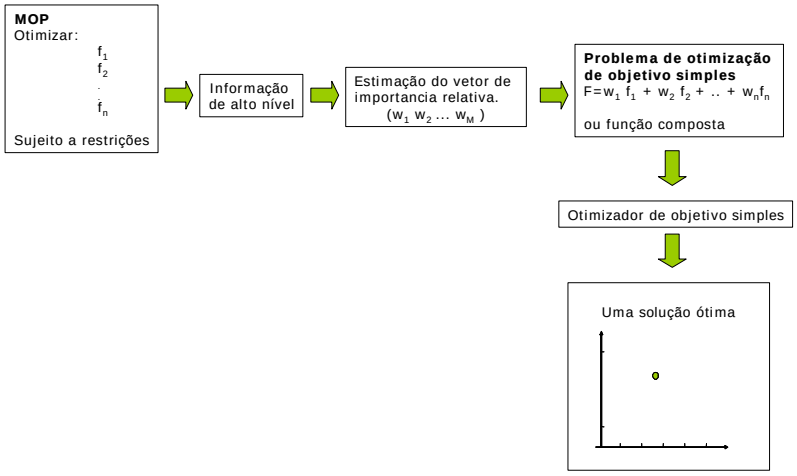


Figura 3.3: Esquema de um procedimento de otimização multi-objetivo baseado em preferência.

3.2 Problema de Otimização Multi-objetivo.

O problema de Otimização Multi-objetivo (MOP) (chamado também multi critério vetorial) pode ser definido [OSYCZKA, 1985] como o problema de encontrar um vetor de variáveis de decisão que satisfaçam certo conjunto de restrições e otimize um conjunto de funções objetivo. Estas funções formam uma descrição matemática dos critérios de desempenho que com freqüência, estão em conflito uns com outros e que freqüentemente são medidos em unidades diferentes. O termo "otimizar", neste caso, toma então um significado diferente ao do caso de problemas mono-objetivo.

Definição 3.1: Um problema de otimização multi-objetivo possui um número de funções objetivo a serem otimizadas (maximizar ou minimizar). Além disso, o problema possui restrições que devem ser satisfeitas por qualquer solução factível. A formulação geral de um MOP é a seguinte [Déb, 2001]:

$$\begin{array}{lll} \text{Maximizar/minimizar} & \boldsymbol{F}(\boldsymbol{x}) = [\boldsymbol{f}_1(\boldsymbol{x}), \boldsymbol{f}_2(\boldsymbol{x}),..., \boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x})]^T, & M \text{ funções objetivo} \\ \text{Sujeito a} & \boldsymbol{g}_j(\boldsymbol{x}) \geq 0, & j=1, 2,\ldots, J; \\ & \boldsymbol{h}_k(\boldsymbol{x}) = 0, & k=1, 2,..., K; \\ & \boldsymbol{x}_i^{(L)} \leq \boldsymbol{x}_i \leq \boldsymbol{x}_i^{(U)}, & i=1, 2,..., n. \end{array} \quad (3.1)$$

Onde $\boldsymbol{x}$ é o vetor de $\boldsymbol{n}$ variáveis de decisão $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2,..., \boldsymbol{x}_n)^T$. Cada elemento de $\boldsymbol{x}$ é um número real. Os valores $\boldsymbol{x}_i^{(L)}$ e $\boldsymbol{x}_i^{(U)}$, representam o valor mínimo e o valor máximo respectivamente para a variável $\boldsymbol{x_i}$. Estes definem o *espaço de variáveis de decisão* ou *espaço de decisão* **D**. Além, o vetor $\boldsymbol{x}$ será referido também como solução.

As J desigualdades $(\boldsymbol{g}_j)$ e as K igualdades $(\boldsymbol{h}_k)$ são chamadas funções de restrição. Uma solução $\boldsymbol{x}$ factível será aquela que satisfaça as J+K funções de restrição e os 2*n* limites. Caso contrário a solução será não-factível. O conjunto de todas as soluções factíveis forma a *região factível* ou *espaço de busca* **E**.

Cada uma das M funções objetivo $\boldsymbol{f}_1(\boldsymbol{x}), \boldsymbol{f}_2(\boldsymbol{x}),..., \boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x})$ podem ser maximizadas ou minimizadas. O vetor de funções objetivo $\boldsymbol{F}(\boldsymbol{x})$ conforma um espaço multidimensional chamado *espaço de objetivos* **Z** onde cada elemento de $\boldsymbol{F}(\boldsymbol{x})$ são números reais. Então para cada solução $\boldsymbol{x}$ no espaço de decisão **D**, existe um $\boldsymbol{f}_{\boldsymbol{m}}(\boldsymbol{x})$, onde m =1,2,...M, no espaço de objetivos. Esta é uma diferença fundamental em relação

à otimização de objetivos simples, cujo espaço de objetivos é unidimensional. O mapeamento acontece então entre um vetor $\boldsymbol{x}$ (n-dimensional) e um vetor $\boldsymbol{F}(\boldsymbol{x})$ (M-dimensional).

$$\mathbb{R}^n \rightarrow \mathbb{R}^M$$

A seguir enunciamos um problema de otimização.

Problema 3.1: Deseja-se encontrar o conjunto de soluções ótimas que minimize o problema de otimização Multi-objetivo cujas funções objetivo são as seguintes:

$$\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{x}^2$$
$$\boldsymbol{f}_2(\boldsymbol{x}) = (\boldsymbol{x} - 2)$$

Em um espaço de busca **E** de $[-10^3, 10^3]$

A variável de decisão neste caso é um escalar $\boldsymbol{x}$ e as funções objetivo, chamado *espaço de objetivos* **Z**, e são representadas vetorialmente como:

$$\text{Minimizar } \boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{f}_1(\boldsymbol{x}) \\ \boldsymbol{f}_2(\boldsymbol{x}) \end{bmatrix}$$

3.3 Conceitos Básicos da Otimização Multi-Objetivo

3.3.1 MOP convexo e não convexo

Antes de definir um MOP convexo, primeiro damos a definição de função convexa.

Definição 3.2 : Uma função $\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^M$ é uma função **convexa** se para dois pares de soluções qualquer $\boldsymbol{x_1}$ e $\boldsymbol{x_2} \in \mathbb{R}^n$, a seguinte condição é verdadeira:

$$\boldsymbol{f}(\theta \boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2) \leq \theta \boldsymbol{f}(\boldsymbol{x}_1) + (1-\theta)\boldsymbol{f}(\boldsymbol{x}_2) \qquad (3.2)$$

onde θ é um escalar nos limites $0 \leq \theta \leq 1$.

Definição 3.3 : A função $\boldsymbol{f}(\boldsymbol{x})$ é estritamente convexa se, para $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, o sinal $\leq$ da equação 3.2 pode ser cambiada com uma desigualdade (<).

Uma função convexa não pode ter nenhum valor maior que os valores da função obtidos por meio de interpolação linear entre $f(x_1)$ e $f(x_2)$. A Figura 3.4 ilustra uma função convexa.

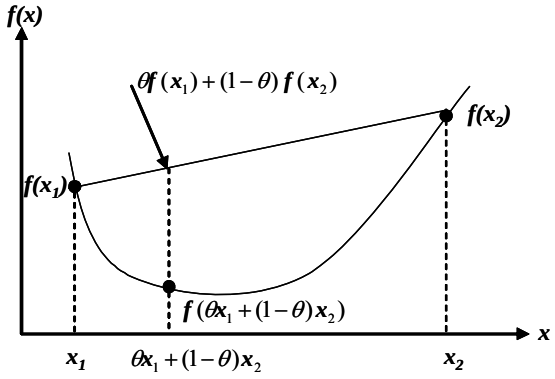


Figura 3.4 : Uma função convexa.

Definição 3.3 : Um conjunto é definido como **conjunto convexo** em um espaço *n*-dimensional se, para todos os pares de ponto x_1 e x_2 no conjunto, o segmento de reta que os une está também completamente dentro do conjunto. Desta forma, todo ponto x onde,

$$x = \theta x_1 + (1-\theta)x_2, 0 \le \theta \le 1 \tag{3.3}$$

está também no conjunto (Figura 3.5).

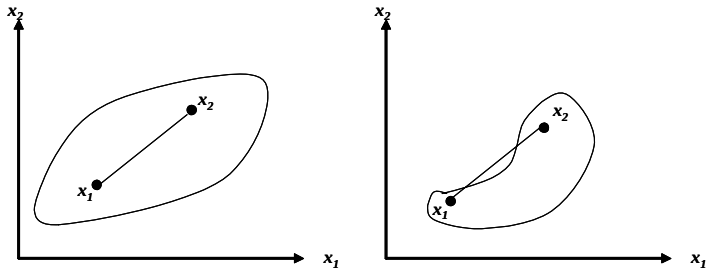


Figura 3.5 : Conjunto convexo e não convexo.

Definição 3.4: Um MOP é convexo se todas as funções objetivos são convexas e a região factível é convexa (ou todas as restrições desiguais são não convexas e restrições de igualdade são linear).

3.3.2 Dominância e Ótimalidade de Pareto.

Quando se têm mais de uma função objetivo, a noção de ótimo muda, já que em problemas multi-critério o objetivo é chegar a um bom compromisso, em lugar de uma solução única como em otimização de objetivo simples. A noção do ótimo em problemas Multi-objetivo foi originalmente proposta por Edgeworth em 1881 e depois generalizada por Vilfredo Pareto em 1896 [COELLO, C. A., 1999]. Nesta seção definimos o conceito de dominância e alguns termos relacionados.

Conceito de dominação

A maioria dos algoritmos usados para resolver um MOP usam o conceito de dominação. Em estes algoritmos, duas soluções são comparadas sobre a base de se uma domina a outra função ou não. Descrevemos este conceito a continuação:

Assumimos que existem M funções objetivos. Para exemplificar a minimização e maximização de funções objetivo, introduzimos o operador $\lhd$ entre duas soluções *i* e *j* como $i \lhd j$ para denotar que a solução i é melhor que a solução *j* sobre um particular objetivo. Similarmente, $i \rhd j$ para um objetivo em particular implica que a solução *i* é pior que a solução *j* sobre este objetivo.

Definição 3.5: Uma solução $\boldsymbol{x_1}$ domina a outra solução $\boldsymbol{x_2}$, se ambas condições 1 e 2 são satisfeitas:

1. A solução $\boldsymbol{x_1}$ não é pior que $\boldsymbol{x_2}$ em todos os objetivos, ou $\boldsymbol{f_j}(\boldsymbol{x_1}) \not\rhd \boldsymbol{f_j}(\boldsymbol{x_2})$ para todo $\boldsymbol{j} = 1, 2, ..., M$.
2. A solução $\boldsymbol{x_1}$ é estritamente melhor que $\boldsymbol{x_2}$ em ao menos um objetivo, ou $\boldsymbol{f}_{\bar{j}}(\boldsymbol{x}_1) \lhd \boldsymbol{f}_{\bar{j}}(\boldsymbol{x}_2)$ para ao menos um $\bar{\boldsymbol{j}} \in \{1,2,...,\boldsymbol{M}\}$.

Matematicamente podemos expressar que $\boldsymbol{x_1}$ domina a $\boldsymbol{x_2}$ como $\boldsymbol{x_1} \preceq \boldsymbol{x_2}$. O que também significa em palavras:

- x_2 é dominado por x_1;
- x_1 é não dominado por x_2, ou;
- x_1 é não inferior a x_2.

Definição 3.6: As soluções x_1 e x_2 são não comparáveis ou indiferentes se e só se:

$$(x_1 \not\preceq x_2) \wedge (x_2 \not\preceq x_1) \wedge (x_2 \neq x_1) \tag{3.4}$$

Neste caso, nem x_1 é melhor que x_2, nem x_2 pode ser considerado melhor que x_1. Se bem que os vetores x_1 e x_2 não são estritamente iguais, ambas soluções podem ser consideradas igualmente boas devido que nenhuma é superior à outra quando consideramos todos os objetivos. Diz-se então que o vetor x_2 é indiferente ou não comparável ao vetor x_1 e denota-se como $x_2 \sim x_1$.

A continuação mostramos um exemplo:

Exemplo 3.1: Na hora de comprar um carro suponha-se que se está procurando o preço e conforto. O objetivo é minimizar o custo e maximizar o conforto. Neste casso mostrado na Figura 3.6 tem-se cinco possíveis opções de compra. Intuitivamente, descarta-se a solução 1, já que a solução 5 fornece mais conforto por igual preço. A solução 2 é descartada pela mesma razão. Têm-se então três soluções: 3,4,5, que são boas alternativas de compra. Em termos quantitativos nenhuma é melhor que a outra, pois uma é mais confortável, mas menos barata, e vice-versa. Existe então um "compromisso" entre os objetivos. Quanto maior o conforto, maior o preço caro e vice-versa.

Diz-se que uma solução domina uma outra se seus valores são melhores em todos os objetivos. Por exemplo, a solução 5 domina a solução 1. Então a solução 5 é não dominada por nenhuma outra. O mesmo acontece com as soluções 3 e 4. Se não se conhece a priori a importância relativa de cada objetivo, pode-se dizer que as soluções 3,4, e 5 são igualmente boas. Portanto, existe um conjunto de soluções ótimas, este conjunto é chamado de conjunto não dominado. As outras soluções (1 e 2) formam o conjunto dominado.

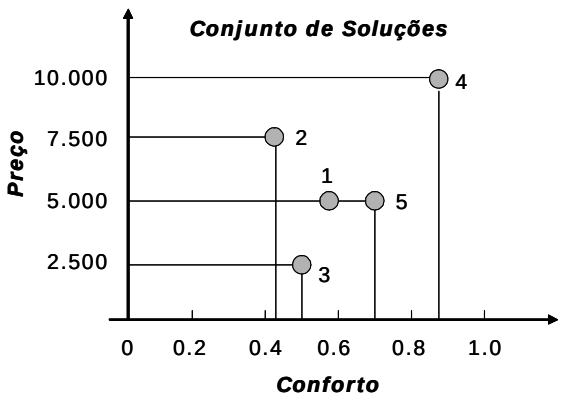


Figura 3.6 : Gráfica das opções de compra.

Ótimalidade de Pareto.

Para um conjunto finito de soluções dado, podemos calcular todas as comparações par a par e encontrar qual solução domina a qual e quais soluções são não dominadas com respeito a cada outra. Este conjunto tem a propriedade de que para qualquer outra solução fora dele, podemos encontrar uma solução neste conjunto a qual dominará a anterior. Por tanto, este conjunto tem a propriedade de dominar todas as outras soluções que não pertencem a ele. Isto significa que as soluções deste conjunto são melhores comparados com as soluções restantes. Este conjunto é chamado ***conjunto não dominado*** para um conjunto de soluções dado.

No exemplo 3.1, as soluções 3,4 e 5 constituem o conjunto não dominado de um conjunto dado de cinco soluções. Definimos um conjunto não dominado a continuação.

Definição 3.7: Em um conjunto de soluções P, o ***conjunto de soluções não dominadas*** P^* é aquele que contém soluções que não são dominadas por qualquer membro do conjunto P.

Quando o conjunto P é o espaço de busca completo, o conjunto resultante não dominado P^* é chamado ***conjunto de ótimos de Pareto***.

Definição 3.8: Dado um problema de otimização Multi-objetivo com um vetor de funções ***F(x)***, o ***conjunto de ótimos de Pareto*** ($\boldsymbol{P}^*$) define-se como:

$$\boldsymbol{P}^* := \{\boldsymbol{x} \in \boldsymbol{D} \mid \neg\exists \boldsymbol{D} \quad \boldsymbol{F}(\boldsymbol{x}') \preceq \boldsymbol{F}(\boldsymbol{x})\} \tag{3.5}$$

Definição 3.9: A Fronteira de Pareto está formada pelo conjunto de vetores de funções objetivo $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}))^T$, para cada solução $\boldsymbol{x}$ que está no conjunto de ótimos de Pareto.

A Fronteira de Pareto está formada então pelos valores das funções objetivos (ponto no espaço de objetivos) correspondentes a cada solução no espaço de busca. A relação de dominância também pode ser classificada em dominância forte e fraca. A dominância forte é definida como:

Definição 3.10: A solução $\boldsymbol{x_1}$ domina fortemente a solução $\boldsymbol{x_2}$ (representado como $\boldsymbol{x_1} \prec \boldsymbol{x_2}$) se é *estritamente melhor* à solução $\boldsymbol{x_2}$ em todos os M objetivos.

3.3.3 Condições de ótimalidade

Estabelecemos condições de ótimalidade para o MOP dado na equação 3.1. Assumimos que todas as funções objetivos e restrições são contínuas e diferençáveis. O seguinte teorema oferece suficientes condições para que uma solução seja Pareto-ótima para funções convexas.

Teorema 3.1: ***Condições de não inferioridade de Kuhn-Tucker***: Se uma população $\boldsymbol{x} \in \boldsymbol{D}$ é não inferior para um MOP, então existem $\boldsymbol{w_l} \geq 0$, $l = 1, 2, \ldots, M$ (w_r é estritamente positiva para alguma $r = 1, 2, \ldots, M$), onde M é o número de objetivos e $\lambda_i \geq 0, i = 1, 2, \ldots, J$, tal que:

$$\boldsymbol{x} \in \boldsymbol{D}$$

$$\lambda_i \boldsymbol{g_i}(\boldsymbol{x}) = 0 \quad i = 1, 2, \ldots, J$$

e $$\sum_{l=1}^{k} \boldsymbol{w_l} \nabla \boldsymbol{f_l}(\boldsymbol{x}) - \sum_{l=1}^{m} \lambda_i \nabla \boldsymbol{g_i}(\boldsymbol{x}) = 0$$

Onde $\boldsymbol{g_i(x)}$, é a *i-ésima* restrição.

Estas condições são necessárias para que uma solução seja não inferior, e quando todas as $\boldsymbol{f_i(x)}$ são côncavas e **D** é um conjunto convexo, então também são condições suficientes.

Exemplificação

Voltamos ao problema 3.1 a fim de ilustrar os conceitos apresentados. Considerando agora o intervalo [-5;5] .

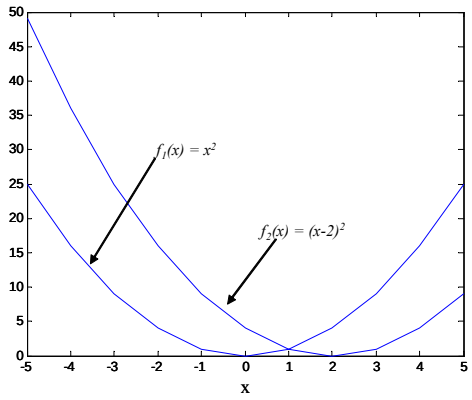


Figura 3.7: Gráfica do problema 1.

Note que a função f_1 alcança o valor mínimo quando x é zero, por enquanto o mínimo para f_2 obtém-se quando x é igual a dois. Para valores inferiores a zero, ambas funções têm pendentes de igual sinal. Quando x é zero, a pendente de f_1 muda de sinal, por enquanto f_2 muda de sinal só quando $x = 2$. Então, para os valores de x em [0;2], uma função diminui e a outra aumenta. Por essa razão, o conjunto Pareto Ótimo para o MOP pranteado no problema 1 é:

$$P^* = \{x \mid 0 \leq x \leq 2\}$$

Para qualquer solução que não esteja no conjunto P^*, existe sempre uma solução que a domina. O Frente Pareto Ótimo correspondente é formada pelo conjunto de vetores bidimensionais obtidos ao avaliar a função f_2 para cada valor de $x \in P^*$. A figura 3.8 mostra os valores de PF^* para $f_1(x)$ e $f_2(x)$ no intervalo [0;2].

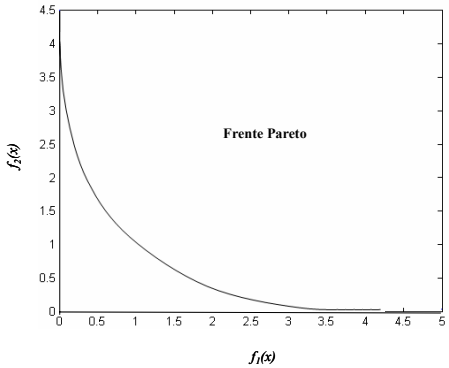


Figura 3.8: Frente Pareto das funções objetivo f_1 e f_2 do problema 1.

3.4 Metas em Otimização Multi-Objetivo

Quando a informação adicional sobre importância dos objetivos é desconhecida, todas as soluções Pareto-ótimas são igualmente importantes. Deb [DEB, 2001] assinala duas importantes metas em Otimização Multi-Objetivo:

1. Encontrar um conjunto de soluções o mais próximo possível da Fronteira de Pareto.

2. Encontrar um conjunto de soluções com a maior diversidade possível.

A primeira meta é comum para qualquer processo de otimização. Soluções muito distantes da Fronteira de Pareto não são desejáveis. Porém, encontrar a maior diversidade dentro das soluções é uma meta específica para Otimização Multi-Objetivo. A Figura 3.9a mostra-se uma boa distribuição de soluções na fronteira de Pareto, entanto na Figura 3.9b as soluções estão distribuídas apenas em algumas regiões. É necessário assegurar a maior cobertura possível da fronteira, já que implica que se tem um bom conjunto de soluções "comprometidas" com os objetivos desejados. Como um MOP trabalha-se com o espaço de decisões e o espaço de objetivos, é desejável que as soluções tenham uma boa diversidade nestes espaços. Normalmente, uma boa diversidade em um de estes espaços garante também a diversidade no outro. Em alguns problemas isto não acontece.

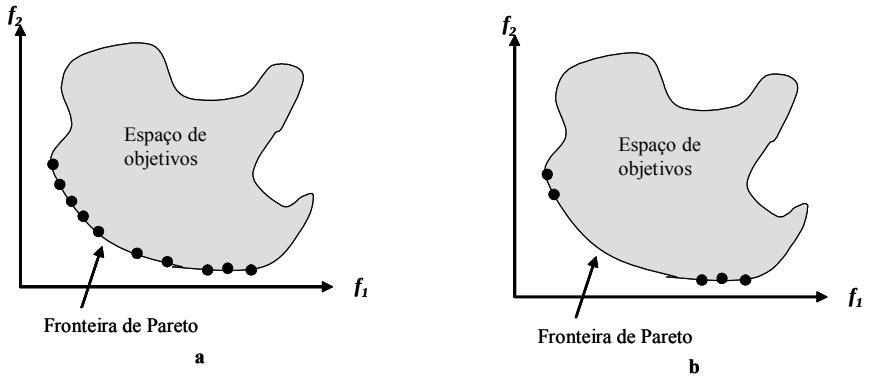


Figura 3.9: Distribuição de soluções na Fronteira Pareto.

3.5 Diferenças com a Otimização de Objetivos Simples

Deb [DEB, 2001] identifica três importantes diferenças entre Otimização Multi-Objetivo e Otimização de Objetivo Simples:

1. Em problemas de otimização com um objetivo, a meta é achar uma solução ótima global (máximo ou mínimo). Em problemas multi-modal podem existir mais de um ótimo global. Já em MOP, achar o conjunto de soluções da Fronteira de Pareto é tão importante quanto preservar a diversidade neste conjunto. Um algoritmo eficiente para otimização multi-objetivo deve considerar ambos aspectos.

2. Um MOP trabalha com dois espaços (variáveis e objetivos) no lugar de um. Problemas com objetivo simples trabalham unicamente no espaço de variáveis já que procuram apenas uma solução no espaço de objetivos. Novamente, manter a diversidade em ambos espaços complica mais o problema, dado que a proximidade de duas soluções no espaço de variáveis não implica proximidade no espaço de objetivos.

3. Os métodos tradicionais de otimização multi-objetivo estão baseados em uma função simples a qual pondera cada objetivo. Podem também tratar cada objetivo separadamente, utilizando os demais objetivos como restrições. Portanto, um MOP pode ser convertido mediante algumas técnicas, em um problema de otimização simples.

3.6 Convergência e diversidade das soluções de um MOP

A principal ferramenta computacional para resolver MOPs são os algoritmos evolutivos, como os baseados em AG. Outras técnicas evolutivas são colônias de formigas – ACO [DORIGO M., 1992], *Tabu Search* – TS [GLOVER, F., 1989; Glover, F., 1990], *Simulating Annealing* – SA [KIRKPATRICK S., 1983], etc.

A diferença dos algoritmos genéticos simples que procuram uma solução única, os algoritmos evolucionários multi-objetivo tentam encontrar todos os elementos do conjunto de Pareto como seja possível. Por isto, é importante ter em conta o seguinte:

1. Convergência das soluções para o conjunto de ótimos de Pareto.

2. Manter a diversidade das soluções dentro do conjunto de ótimos de Pareto.

A primeira tentativa para manter a diversidade na população surge com a idéia de Holland ao introduzir o conceito de agrupamento (*crowding*), identificando situações onde os indivíduos se concentram em espaços ou nichos (isto último em referência ao que acontece na natureza onde diferentes espécies de animais se agrupam em função de suas características), competindo por recursos e com isto diminuindo as expectativas de vida e taxa de nascimento. Mas, o fundamento dado por Golberg e Richarson [GOLBERG & RICHARDSON, 1987] que permite obter um valor modificado de *fitness* para cada um dos indivíduos, é o mais utilizado atualmente, descrevemos este brevemente a continuação.

Ordena-se a população em diferentes subpopulações (nichos), e a *aptidão* de uma população em particular, depende de sua localização dentro da sub população e da quantidade de soluções vizinhas que ela tem. A este número de soluções vizinhas que uma solução tem chama- se contador do nicho e a proximidade entre soluções pode-se medir dentro do espaço das funções objetivo ($\boldsymbol{f_i}$) utilizando uma métrica Euclidiana. O valor de bondade ou aptidão do indivíduo *i* é obtido pela seguinte expressão:

$$\boldsymbol{F_{si}} = \frac{\boldsymbol{f_i}}{\sum_{\boldsymbol{j}=1}^{\boldsymbol{M}} \phi(\boldsymbol{d_{ij}})}, \tag{3.6}$$

Onde *M* é o número de indivíduos localizados na vizinhança do *i-ésimo* indivíduo.

$\phi(\boldsymbol{d_{ij}})$ define-se como a função de compartilhamento (*sharing function*) e seu valor obtém-se de:

$$\phi(\boldsymbol{d_{ij}}) = \begin{cases} 1 - \left(\dfrac{\boldsymbol{d_{ij}}}{\sigma_{\boldsymbol{share}}} \right) \alpha, & \boldsymbol{d_{ij}} < \sigma_{\boldsymbol{share}} \\ 0 & \boldsymbol{outro \quad caso} \end{cases} \tag{3.7}$$

Onde normalmente $\alpha = 1; \boldsymbol{d_{ij}}$ é uma medida da distância entre os indivíduos *i* e *j* e $\sigma_{\boldsymbol{share}}$ é um parâmetro que determina a proximidade permitida entre indivíduos.

A manutenção da diversidade permite aos AGs buscar muitos picos em paralelo, evitando ficar trancados em ótimos locais do espaço de busca.

A seguir descrevemos alguns algoritmos que usamos em nossa implementação e estão baseados nos conceitos descritos anteriormente.

3.7 Classificação das técnicas utilizadas para resolução de um MOP

Os algoritmos evolucionários multi-objetivo, quanto a seus princípios básicos, podem ser classificados como aqueles que utilizam em forma direta os conceitos de Pareto e os que não [FONSECA C.M. & FLEMING P.J.].

A seguir, descrevemos as técnicas usadas em MOP, aquelas não baseadas em Pareto e aquelas baseadas em Pareto.

3.7.1 Técnicas Não Baseadas em Pareto.

Considera aquelas técnicas que não incorporam de maneira direta o conceito de dominância de Pareto (ou ótimo de Pareto). Estas técnicas são muito eficientes mas com freqüência adequados a poucas funções objetivos (não mais de três):

- Ordenamento lexicográfico . Alguns usos são:
    - Compactação de circuito com representação simbólica [FOURMAN, 1985]
    - Planejamento de movimentos de robôs [Gacôgne, 1999]
    - Planejamento de horarios [EL MOUDANI et al., 2001]
- Soma Ponderada. Alguns usos de esta técnica são:
    - Engenharia ambiental [GARRET et al., 1999].
    - Desenho de controladores [DONHA, 1997].
    - Desenho de filtros óticos para lâmpadas [Eklund, 2001].
    - Desenho de antenas [VAN VELDHUIZEN et al., 1998].
- VEGA (*Vector Evaluated Genetic Algorithm*) [SCHAFFER, 1984].

- Método ε-constraint. Alguns usos de esta técnica são:
    - Desenho preliminar de veículos marinos [LEE, 1997].
    - Contaminação de aqüíferos [CHETAN, 2000].
    - Desenho de sistemas tolerantes a falhas [SCOTT, 1995].
    - Problemas de engenharia ambiental [KUMAR, 2002].
- Satisfação de metas. Alguns usos de esta técnica são:
    - Desenho de filtros IIR [WILSON, 1993].
    - Otimização estrutural [SANDGREN, 1994; HAJELA, 1992].
    - Contrabalanço de braços de robô [COELLO, 1998].
- Teoria de jogos. Alguns usos de esta técnica são:
    - Otimização de armaduras [DHINGA, 1994; RAO, 1993].
    - Problemas de aerodinâmica [PÉRIAUX, 1996].

3.7.2 Técnicas Baseadas em Pareto

A idéia de utilizar o cálculo de aptidão baseado no conceito de ótimo de Pareto foi sugerida originalmente por David E. Golberg em 1989 afim de lidar com as limitações de VEGA, ele sugeriu o uso de uma seleção baseada em não dominância, afim de mover uma população até o frente de Pareto em um problema multi-objetivo.

A idéia fundamental é selecionar os indivíduos não dominados com respeito à população atual e calcular a hierarquia e a aptidão mais elevada. Posteriormente eliminam-se temporalmente estes indivíduos da competição e se re-hierarquiza a população restante. O processo repete-se até que toda a população esteja hierarquizada.

Golberg sugeriu também o uso de nichos ou alguma técnica similar para manter a diversidade. Isto é necessário devido ao ruído estocástico, que os algoritmos evolutivos tendem a gerar para uma solução única independente da distribuição inicial da população.

- Hierarquização de Pareto "pura". Alguns usos de esta técnica são:
    - Redes de monitoração [CIENIAWSKI, 1995].

- Planejamento de horários de bombeio [SCHWAB, 1996; SAVIC, 1997].
- Estudo de exeqüibilidade de submarinos [THOMAS, 1998].
- Planejamento de um sistema de distribuição de potência elétrica [RAMIREZ, 1999].

- MOGA (*Multi-Objective Genetic Algorithm*) [FONSECA & FLEMING, 1993].
- NSGA (*Nondominated Sorting Genetic Algorithm*) [DEB, 1994] e NSGAII (*Nondominated Sorting Genetic Algorithm II*) [DEB et al., 2000].
- NPGA (*Niched-Pareto Genetic Algorithm*) [HORN et al., 1993] e NPGA2 (*Niched-Pareto Genetic Algorithm 2*) [ERICKSON et al., 2001].

Enfoques recentes:

- PAES (*Pareto Archived Evolution Strategy*) [KNOWLES, 1999], PESA (*Pareto Envelope-based Selection Algorithm*) [CORNE et al., 2001] e PESA II (*Pareto Envelope-based Selection Algorithm II*) [CORNE et al., 2001].
- SPEA (*Strength Pareto Evolutionary Algorithms*) [ZITZLER, 1998] e SPEA2 (*Strength Pareto Evolutionary Algorithms 2*) [ZITZLER et al., 2001].
- MOMGA (*Multi-Objective Messy Genetic Algorithm*) [VAN VELDHUIZEN, 1999] e MOMGA II (*Multi-Objective Messy Genetic Algorithm II*) [ZYDALLIS et al., 2001].
- Micro-algoritmo genético. [TOSCANO & COELLO, 2001]

3.8 Algoritmos Evolutivos Multi-objetivo

O potencial dos algoritmos evolutivos para resolver problemas de otimização multi-objetivo remonta-se a finais de 1960 quando a tese doutoral de Rosenberg (1967) indicou a possibilidade de usar algoritmos genéticos neste domínio. Sem embargo, o estudo de Rosenberg não abordou realmente problemas multi-objetivo, mas apenas indicou essa possibilidade. A sugestão consistiu em usar múltiplas *"propriedades"*

(perto a alguma composição química específica) em sua simulação da genética e da química de uma população de organismos unicelulares. Dado que sua implementação usou apenas uma propriedade, o enfoque Multi-objetivo não foi requerido.

A primeira tentativa real de estender um algoritmo evolutivo a problemas multi-objetivo é o *Vetor Evaluated Genetic Algorithm* (VEGA) desenvolvido por David Schaffer em sua tese doutoral de 1984 e apresentado na Primeira Conferencia Internacional de Algoritmos Genéticos em 1985.

A seguir descrevemos os algoritmos evolucionários multi-objetivo que implementamos para o problema do Alinhamento Múltiplo de Seqüências de proteínas.

3.8.1 Soma Ponderada (*Weighted Sum*)

O algoritmo Soma Ponderada é um método considerado clássico em MOP, ele utiliza o procedimento de otimização multi-objetivo baseado em preferência, o qual foi descrito na seção 3.1.1 e na Figura 3.3. Como seu nome sugere, ele escala um conjunto de objetivos em um objetivo simples pre multiplicando cada objetivo com um peso o qual é especificado pelo usuário.

A escolha dos pesos é um problema importante que depende da relevância de cada objetivo. É necessário realizar o escalonamento de cada função objetivo dado que os diferentes objetivos podem ter diferentes magnitudes. Por exemplo o preço de um carro pode variar de R$/.3000 a R$/.30000, enquanto o conforto pode estar entre 0% e 100%.

Uma vez que os objetivos estejam normalizados pode-se formular uma função ***F(x)*** formada pela soma dos objetivos normalizados multiplicados por seus pesos. Assim, tem-se:

$$\begin{aligned} &\text{minimizar/maximizar} && \boldsymbol{F}(\boldsymbol{x}) = \sum_{\boldsymbol{m}=1}^{\boldsymbol{M}} \boldsymbol{w_m}\, \boldsymbol{f_m}(\boldsymbol{x}), && \\ &\text{restrita a} && \boldsymbol{g_j}(\boldsymbol{x}) \geq 0, && j = 1, 2, ..., J; \\ & && \boldsymbol{h_k}(\boldsymbol{x}) = 0, && k = 1, 2, ..., K; \\ & && \boldsymbol{x_i^{(L)}} \leq \boldsymbol{x_i} \leq \boldsymbol{x_i^{(U)}}, && i = 1, 2, ..., n. \end{aligned} \qquad (3.8)$$

Onde $w_m \in [0,1]$ é o peso para cada função objetivo f_m. Usualmente na prática escolhem-se os pesos tal que sua soma seja 1, ou $\sum_{m=1}^{M} w_m = 1$.

A vantagem principal da abordagem por Soma Ponderada é sua implementação direta. Dado que uma única função objetivo é usada no cálculo do fitness, um algoritmo genético simples pode ser usado com modificações mínimas. Também, esta abordagem é computacionalmente muito eficiente.

A principal desvantagem é que nem todas as soluções de Pareto ótimas podem ser pesquisadas quando a verdadeira frente de Pareto não é convexa. Por isto, algoritmos genéticos multi-objetivo baseados em Soma Ponderada têm dificuldades na busca de soluções uniformemente distribuídas sobre uma superfície de soluções não convexa [DEB, 2001]. Existem um número de interessantes teoremas que demonstram o relacionamento entre a solução ótima deste algoritmo com as soluções ótimas de Pareto, alguns deles são Chankong e Haimes, 1983; Ehrgott, 2000; Miettinen, 1999. A seguir enunciamos dois teoremas interessantes. [MIETTINEN, 1999].

Teorema 3.1. A solução de um problema representado pela equação 3.8 é Pareto-ótima se os pesos são positivos para todos os objetivos.

Teorema 3.2. Se x é uma solução Pareto-ótima para um MOP convexo, então existe um vetor de pesos positivos w não zero, tal que x é uma solução para um problema representado pela Equação 3.8. O teorema 3.2 garante que quando o MOP é convexo, qualquer solução Pareto-ótima pode ser achada usando o Método de Soma Ponderada.

DEB ilustra a forma como a abordagem por Soma Ponderada pode encontrar soluções ótimas de Pareto para um problema representado pela equação 3.8 [DEB, 2001]. Por simplicidade, consideramos um problema de dois objetivos, como mostrado na Figura 3.10.

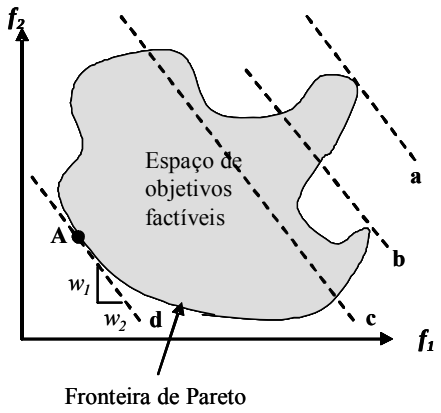


Figura 3.10: Ilustração da aproximação Soma Ponderada sobre um Frente Ótimo de Pareto convexo.

O espaço objetivo factível e o correspondente conjunto de soluções ótimas de Pareto são mostrados. Com dois objetivos, existem dois pesos w_1 e w_2, mas só um é independente. Conhecendo qualquer um, o outro pode ser calculado por simples subtração. Conhecendo os pesos, podemos calcular a função composta **F**, sua superfície de contorno pode então ser visualizada no espaço objetivo, como é mostrado pelas linhas '**a**', '**b**', '**c**' e '**d**' na Figura 3.10. Já que **F** é uma combinação linear de ambos objetivos f_1 e f_2, nós esperaríamos uma linha reta como a linha de contorno de **F** no espaço objetivo. Isto é porque qualquer solução sobre a linha de contorno terá o mesmo valor de **F**. Note que esta linha de contorno não é uma linha arbitraria, sua tangente esta relacionada à eleição do vetor peso. De fato, para dois objetivos, sua tangente é $-w_1/w_2$. A localização da linha depende do valor de **F** sobre qualquer ponto na linha. O efeito de abaixar a linha do contorno de '**a**' a '**b**' é de fato um salto das soluções de uns valores mais elevados de **F** a um mais baixo.

Se o problema representado pela equação 3.8 requer minimização de **F**, a tarefa é encontrar a linha de contorno com o valor de **F** mínimo. Isto acontece com a linha de contorno que é tangencial ao espaço de busca e que também cai no canto inferior-esquerdo deste espaço. Na Figura 3.10, esta linha é marcada como '**d**'. O ponto tangente '**A**' é a solução mínima de **F**, e conseqüentemente é a solução ótima de Pareto correspondente ao vetor peso.

Se um vetor de pesos diferente é usado, a tangente da linha de contorno deverá ser diferente e o procedimento descrito, em geral, deverá resultar em uma solução

ótima diferente. Usando o teorema 3.2, podemos argumentar que para um MOP convexo, múltiplas soluções ótimas de Pareto podem ser encontradas resolvendo o problema da equação 3.8 com múltiplos vetores de pesos positivos, um à vez, cada vez encontrando uma solução ótima de Pareto. O teorema pode ser estendido para um MOP não convexo, particularmente quando o frente de ótimos de Pareto é convexo.

Na Figura 3.11 mostramos o fluxo do algoritmo Soma Ponderada.

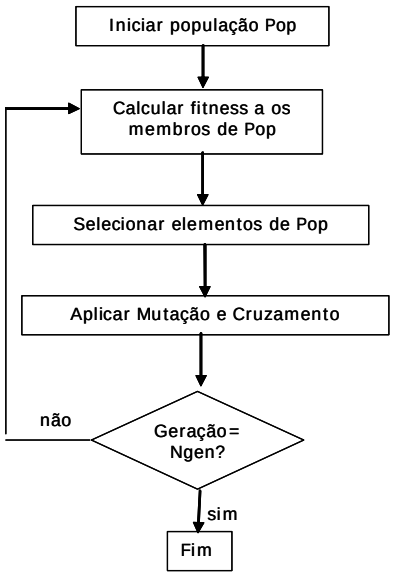


Figura 3.11: Fluxo do algoritmo Soma Ponderada.

3.8.2 SPEA (Strength Pareto Evolutionary Algorithm) [ZITZLER et al., 1999].

Zitzler e Thiele propuseram um algoritmo evolucionário elitista, o qual chamaram SPEA. Este algoritmo aplica elitismo através do armazenamento das soluções não dominadas em uma população externa $\mathbf{P_{nd}}$ de tamanho N_{max} fornecido como parâmetro, esta população participa do procedimento de seleção. Além disso, o cálculo do valor de aptidão de todos os elementos da população atual realiza-se utilizando um procedimento baseado na determinação de força (*strength value*, usa-se este conceito no sentido de dominação)**,** este valor é proporcional ao número de soluções que certo indivíduo domina. Este procedimento induz a formação de *nichos* a

partir do conceito de dominância de Pareto, chamado *niching for strength* [ZITZLER E., THIELE L., 1998].

O algoritmo começa criando uma população inicial aleatória **Pop** de tamanho N e uma população externa $\mathbf{P_{nd}}$ inicialmente vazia. A continuação calcula-se os elementos não dominados de **Pop** e estes são copiados à população externa $\mathbf{P_{nd}}$. Depois, calcula-se o valor da aptidão de $\mathbf{Q} = \mathbf{Pop} \cup \mathbf{P_{nd}}$, este valor é obtido em várias etapas. Primeiro, um valor ***count*** é encontrado para cada solução ***i*** de $\mathbf{P_{nd}}$ usando a Equação 3.9.

$$\boldsymbol{count} = |\{j, j \in \text{Pop, tal que } i \preceq j\} \tag{3.9}$$

O valor ***count*** é o número de soluções que *i* domina em **Pop**. As soluções não dominadas de **Pop** têm ***count*** = 0. Depois, calcula-se o valor $\boldsymbol{s_i}$ (*strength fitness*), conforme à Equação 3.10.

$$\boldsymbol{s_i} = \frac{\boldsymbol{count}}{|\boldsymbol{Pop}|} \tag{3.10}$$

O *strength* para cada indivíduo da população **Pop** calcula-se a partir dos *strenghts* de todas as soluções externas não dominadas que o dominem, conforme a Equação 3.11.

$$\boldsymbol{s_j} = 1 + \sum_{i \succeq j} \boldsymbol{s_i} \tag{3.11}$$

O valor de aptidão de cada indivíduo da população **Q** será igual à inversa do seu *strength*.

Como o conjunto de soluções na população externa pode ser grande e ela intervém no processo evolutivo, uma técnica de *Clustering* chamada método médio de enlace (*average linkage method*), é utilizado para reduzir o número de soluções de tal conjunto e manter a diversidade da população. Esta técnica se baseia na penalização das soluções vizinhas mais próximas à que avaliamos.

A técnica de *Clustering* assegura que uma melhor propagação das soluções não dominadas seja conseguida entre todas as soluções obtidas. Para fazer que esta técnica encontre o conjunto mais diverso das soluções, cada solução extrema pode ser forçada a permanecer em um conjunto independente.

A desvantagem do SPEA é que introduz um parâmetro extra N_{max}, o tamanho da população externa. Um balanço entre o tamanho regular da população N e o tamanho

da população externa N_{max} é importante para o sucesso do algoritmo SPEA. Se N_{max} é muito grande, a pressão da seleção para as soluções elites será grande e o SPEA pode não ser capaz de convergir ao frente de ótimos de Pareto. Por outro lado, se N_{max} é pequeno o efeito do elitismo será perdido.

O fluxo principal do SPEA é mostrado a seguir:

Algoritmo 3.1 : Procedimento principal do **SPEA**

Passo1: Gerar uma população inicial **Pop** e criar o conjunto não dominado externo $\mathbf{P_{nd}}$=0.

Passo 2: Copiar os membros não dominados de **Pop** a $\mathbf{P_{nd}}$.

Passo 3: Apagar as soluções em $\mathbf{P_{nd}}$ que são dominadas por qualquer outro membro de $\mathbf{P_{nd}}$.

Passo 4: Se o número de soluções no armazenamento externo $\mathbf{N_{nd}}$ excede um máximo $\mathbf{N_{max}}$, reduzir $\mathbf{P_{nd}}$ utilizando *clustering*.

Passo 5: Calcular o *fitness* (valor de aptidão) de cada individuo em **Pop**, assim como em $\mathbf{P_{nd}}$.

Passo 6: Selecionar indivíduos de **Pop**(t) + $\mathbf{P_{nd}}$ (t) (união multiconjunto), até encher **Pop**(t+1).

Passo 7: Aplicar os operadores de mutação e cruzamento específicos do problema.

Passo 8: t=t+1

Passo 9: Se alcançar o máximo número de gerações parar, senão ir ao Passo 2.

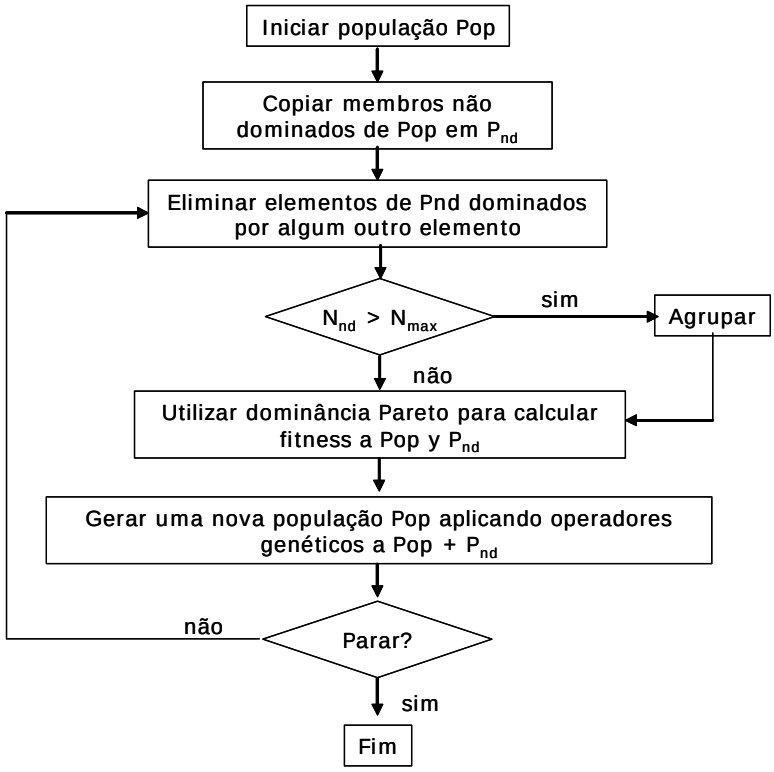


Figura 3.12: Fluxo do algoritmo SPEA.

3.8.3 NSGA-II (Non Sorting Genetic algorithm)

O NSGA-II é um algoritmo baseado em classificação por não dominância para calcular o valor de aptidão dos elementos da população genética [DEB K., 2000]. Incorpora elitismo e não precisa do parâmetro σ_{share} para incrementar a diversidade na população. A continuação descrevemos as fases do algoritmo.

Classificação por não dominância

É um mecanismo simples de ordenamento de uma população de tamanho N em diferentes níveis não-dominados. Para encontrar o primeiro frente não-dominado, comparamos cada indivíduo da população para determinar aqueles não-dominados. Para encontrar o segundo frente não-dominado separamos da população aqueles indivíduos encontrados no primeiro frente e aplicamos o mesmo procedimento utilizado para encontrar o primeiro frente. Os seguintes níveis são encontrados da mesma forma.

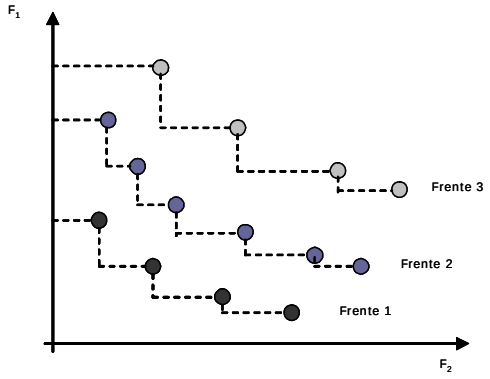


Figura 3.13: Níveis de não dominância

Preservação da diversidade na população

Durante o processo de convergência até o conjunto de ótimos de Pareto é desejável manter uma grande diversidade de elementos na população. O enfoque conhecido como *Function Sharing* apresenta um parâmetro chamado σ_{share} (*Sharing parameter*) que não tem um procedimento sistematizado para encontrar seu melhor valor. Então o NSGA-II substitui esta função de *sharing* por outro enfoque denominado *crowded-comparison* que não requer nenhum valor definido pelo usuário. A fim de entender este enfoque definimos o conceito de estimação de densidade conhecido como *density-estimation metric*.

Estimação de densidade (density-estimation metric)

Para calcular uma estimação da densidade dos indivíduos ao redor de um ponto em particular, calcula-se a distância média de dois pontos vizinhos, um em cada lado do ponto de interesse, utilizando o valor de suas funções objetivo. Este valor, $i_{distance}$, serve como uma estimação do perímetro de um cuboide sendo o vértice o ponto vizinho mais próximo, como se mostra na figura 3.14.

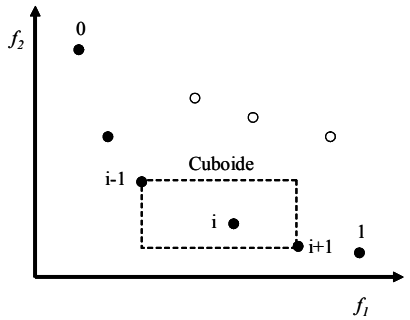


Figura 3.14: C*rowding distance*

Este valor é chamado distância de agrupamento (do inglês *crowding distance*), então a distância de agrupamento do *i-ésimo* indivíduo no frente (marcado como círculos cheios) é o comprimento médio de seus lados. Este processo requer ordenar a população de acordo com o valor de cada função objetivo, em ordem de magnitude ascendente. Então o valor da distância de agrupamento de um indivíduo calcula-se somando as distâncias entre os indivíduos imediatamente maior e menor considerando cada objetivo. Geralmente se normaliza cada função objetivo antes do cálculo, já que podem ser de magnitudes diferentes. O procedimento é o seguinte:

Algoritmo 3.2 : Cálculo da distância de agrupamento(I)

---

l = |I|; número de soluções em I

para cada i, $I[i]_{distance}=0$; inicializar o contador de distancia por cada objetivo m

I = ordenar(I,m); ordena-se usando o valor de cada função objetivo

$I[1]_{distance} = I[l]_{distance} = \infty$; intervalo para os valores extremos

Para i=2 a (l-1); para todos os pontos

$$I[i]_{distance} = I[i]_{distance} + \frac{I[i+1]\boldsymbol{m} - I[i-1]\boldsymbol{m}}{\boldsymbol{f_m^{max}} - \boldsymbol{f_m^{min}}}$$

Onde I[*i*].*m* refere-se ao valor de I da *m-ésima* função objetivo, correspondente ao *i-ésimo* indivíduo no conjunto I. Os parâmetros $f_m^{\max}$ e $f_m^{\min}$ são o máximo e mínimo valor da *m-ésima* função objetivo. Depois que todos os indivíduos tenham seu valor de distância podem-se comparar dois integrantes do conjunto I por sua separação ou proximidade com os outros indivíduos. Uma solução com valor de distância menor tem mais vizinhos e por tanto é menos desejável.

Operador "*crowded-comparison*"

O operador *crowded-comparison* $(\prec_n)$ guia o processo de seleção encaminhando diretamente o algoritmo até um frente ótimo de Pareto. Assume-se que cada indivíduo da população *i* tem dois atributos:

1. Hierarquia de não-dominação (i_{rank});
2. Distância de agrupamento ou *Crowding distance*($i_{distance}$)

Então definimos uma ordem parcial $\prec_n$ como

$i \prec_n j$ se ($i_{rank} < j_{rank}$) ou (($i_{rank} = j_{rank}$) e ($i_{distance} > j_{distance}$))

Significa que entre duas soluções com diferentes níveis de não-dominação, prefere-se a solução com o menor nível (a melhor). É dizer se duas soluções pertencem ao mesmo frente, então se prefere a solução que esta localizada em uma região de menor densidade.

Procedimento geral do NSGA-II

Inicialmente e de forma aleatória, cria-se uma população P_0 de tamanho *N*. Esta população é ordenada em base ao principio de não-dominação. Na primeira iteração calcula-se para cada solução um valor de aptidão (ou hierarquia), igual ao nível de sua dominação (o nível 1 é o melhor, o 2 é o seguinte melhor e assim sucessivamente). Aplica-se seleção por torneio na forma usual, assim como os operadores de cruzamento e mutação para criar uma população de descendentes Q_0 de tamanho *N*. Ambas populações são reunidas em uma população $R_0 = P_0 \cup Q_0$ de tamanho 2*N*. Aplica-se elitismo ao comparar à população atual com a melhor população de indivíduos não dominados encontrados previamente. O procedimento difere depois de produzir a geração inicial, como se descreve a continuação com a *i-ésima* geração.

Primeiro obtém-se uma população formada por $R_t = P_t \cup Q_t$ de tamanho $2N$. Depois, a população R_t é ordenada por níveis de acordo aos princípios de não-dominação. Assim a combinação das populações previa e atual em R_t asseguram o processo de elitismo. Depois que são ordenadas, as melhores soluções pertencem ao conjunto F_1 e estes vão ocupar um papel preponderante durante o processo. Os seguintes melhores indivíduos pertenceram ao conjunto F_2 e assim sucessivamente até formar o último frente F_l. Se o tamanho de F_1 é menor a N, então se devem considerar todos seus elementos para formar a nova população P_{t+1}. O espaço restante na nova população será preenchido pelos subseguintes níveis não-dominados. Este procedimento continua até que nenhum conjunto pode mais ser acomodado. Se a soma dos conjuntos F_{l-1} e F_l é maior que N, ordena-se em forma descendente o último conjunto F_l usando o operador *crowded-comparison* $(\prec_n)$ para selecionar as melhores soluções e preencher exatamente o espaço da nova população do tamanho N. Descreve-se graficamente na Figura 3.15. Aplicam-se os operadores de seleção, cruzamento e mutação à nova população P_{t+1} para formar uma nova população Q_{t+1} de tamanho N. É importante dizer que na técnica de seleção de torneio binário aplica-se o operador *crowded-comparison* $(\prec_n)$. Este operador requer que se conheça a hierarquia e a distância de agrupamento (*crowding-distance*) de cada solução na população. Estes últimos valores calculam-se por enquanto se forma a população P_{t+1} como se mostra no algoritmo 3.3.

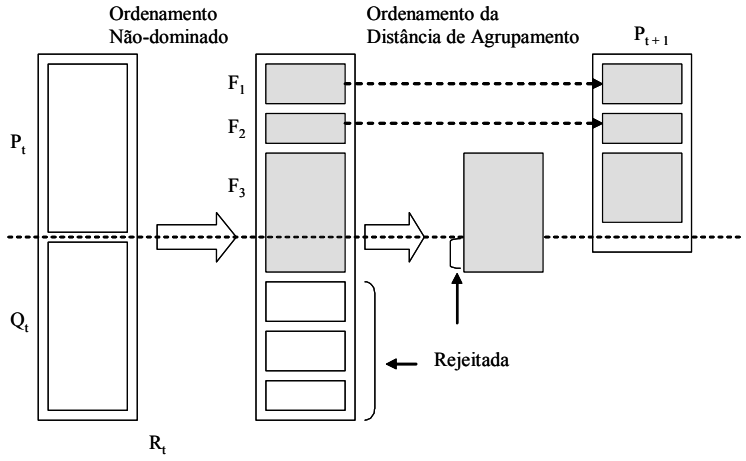


Figura 3.15: Procedimento NSGA-II

Algoritmo 3.3 : NSGA-II

Inicializar (P_0)

t = 0

Avaliar (P_0)

Quando (não critério de parada) fazer

$R_t = P_t \bigcup Q_t$; combina pais e filhos da população

F= **ordenamento_não-dominado** (R_t); F=(F_1, F_2, ...), todos os frentes não dominados de R_t

P_{t+1}=0; *i*=1; Nova população vazia

Quando ($|P_{t+1}|+ |F_i| \leq N$); fazer preencher a população pai.

Calcular a *crowding-distance* (F_i);

$P_{t+1}=P_{t+1} \bigcup F_i$; inclui o *i-ésimo* frente não-dominado na população pai

i++;

ordenar(F_i, $\prec_n$); função que ordena em forma descendente usando $\prec_n$

$P_{t+1}=P_{t+1} \bigcup F_i[(1:(N - |P_{t+1}|)]$; procura os primeiros ($N$ - $|P_{t+1}|$) elementos de F_i

Q_{t+1} = reprodução (P_{t+1}); aplicação de operadores genéticos para criar a nova população Q_{t+1}

t=t+1; incremento do contador de geração.

P_t= Nova população

Retornar melhor solução achada

Podem-se identificar varias áreas promissoras de pesquisa dentro da otimização multi-objetivo (utilizando técnicas de programação matemática).

Mas ainda é preciso desenvolver novos algoritmos que sejam mais eficientes e que possam gerar soluções *compromisso* para problemas mais gerais e requerendo menos restrições. Também se devem desenvolver mais aplicações do mundo real que envolva problemas de grande porte (com muitas variáveis de decisão, funções objetivo custosas de avaliar e processos de tomada de decisões de alta complexidade).

CAPÍTULO

4

MÉTODO DE SOLUÇÃO PROPOSTO PARA O PROBLEMA DE ALINHAMENTO MULTIPLO DE SEQÜÊNCIAS DE PROTEÍNAS

Até agora expusemos o problema do MSA que foi resolvido neste trabalho e a técnica de otimização que foi aplicada.

Neste capítulo descrevemos o método proposto para obter o alinhamento ótimo ou quase-ótimo de seqüências de proteínas com um modelo de valoração dado.

4.1 Justificativa do Método Proposto para Resolução de MSA.

Como foi descrito no capítulo 2, algumas técnicas popularmente utilizadas no problema de MSA, apresentam boas respostas, mais ainda podem ser melhoradas. Existem muitas outras técnicas de otimização que podem ser consideradas para a resolução deste problema, mas uma grande carência atual é que ainda não existe uma única função objetivo que possa ser considerada a "*melhor*" para produzir múltiplos alinhamentos ótimos.

Foi demonstrado [MC CLURE et al., 1994] que o desempenho dos programas de alinhamento depende do número de seqüências, o grado de similaridade entre seqüências e o número de inserções em um alinhamento (gaps). Então precisamos de uma ferramenta que possua grande flexibilidade, adaptabilidade e sólido desempenho a fim de considerar estes critérios.

O problema de alinhar seqüências de biomoléculas pode ser visto como um problema de caráter combinatório, que podemos converter em uma busca de pontos ótimos em um *espaço de alinhamentos*, onde um alinhamento pode ser visto como um estado ou um ponto no *espaço alinhamento*. Em um alinhamento múltiplo de

seqüências existe um grande número de possíveis alinhamentos, então a tarefa de alinhar seqüências torna nosso espaço de busca muito grande para o ótimo ou um quase-ótimo alinhamento. Esta busca, as vezes, se torna intratável como no caso da programação dinâmica para um alinhamento múltiplo de mais de três seqüências.

Os Algoritmos Evolutivos (AE) são especialmente apropriados para resolver problemas de otimização de domínio multidimensional (muitas variáveis de decisão) e/ou quando o espaço de busca é muito grande [VAN VELDHUIZEN, 1999].

Se a cada ponto de nosso espaço de busca associamos uma função de aptidão **(*fitness*)** que mede a "bondade" do correspondente alinhamento dentro do espaço, então precisamos de uma ferramenta que leve a busca da solução a uma "boa" direção criando novas gerações desde alinhamentos com bons valores de aptidão. Os MOEAs provêem um conjunto de técnicas de otimização eficazes para otimizar esta busca, eles são adequados para problemas com grande, complexo e espaços mal compreendidos de busca e reduzem as chances de cair dentro de um extremo local [VAN VELDHUIZEN, 1999]. Esta eficiência no uso de tais problemas é, preferencialmente, o que a diferencia das demais técnicas convencionais de busca [MICHELAWICZ, 1992; SAID, 2005].

A clave deste método para a comparação de seqüências inclui uma boa representação dos alinhamentos de seqüências de biomoléculas, e a aplicação própria das três operações genéticas (*seleção, cruzamento e mutação*) (Apêndice B).

4.2 Método do Processo de Avaliação da Qualidade do MSA de Proteínas.

Existem duas tarefas que são desenvolvidas atualmente na resolução do alinhamento múltiplo de seqüências:

1. A tarefa de encontrar um algoritmo para obter o ótimo ou quase-ótimo alinhamento com um modelo de valoração dado, o qual é usado como procedimento de avaliação para obter ótimos alinhamentos,
2. E a tarefa de encontrar um modelo de valoração tal que o verdadeiro alinhamento matemático ótimo coincida com o verdadeiro alinhamento biológico ótimo.

Este trabalho limita-se à primeira tarefa.

O processo de avaliação dos alinhamentos aqui proposto é realizado utilizando Algoritmos Evolutivos Muli-Objetivo (MOEA) para otimizar duas funções objetivo que avaliam a qualidade do conjunto de MSA analisados.

A primeira função objetivo calcula a qualidade do Alinhamento utilizando a Função Soma de Pares (SP score), a qual foi definida no capítulo 2 Definição 2.2, baseada na matriz de substituição BLOSUM62 (tabela 2.1).

A segunda função objetivo utiliza também a função SP score para calcular a qualidade do Alinhamento, mas baseada na matriz de substituição PAM 250 (tabela 2.2).

Não há uma matriz de substituição perfeita; cada matriz tem suas próprias limitações, então deve ser possível usar matrizes múltiplas de modo que uma complemente os limites da outra [Altschul, S. F. 1991]. As matrizes PAM e BLOSUM foram criadas desde métodos evolucionários muito diferentes (Capitulo 2, seção 2.5), usando estas matrizes em nosso modelo de avaliação pretendemos obter mais diversidade de soluções que por tanto cumpriram com as propriedades evolucionarias destes modelos de matrizes de substituição.

Aplicamos a penalidade de gap afim com os seguintes valores:

gap de abertura = 10

gap de estensão = 0.2

Os algoritmos MOEA aplicados foram descritos no capítulo 3. O SPEA e o NSGAII aplicam uma estratégia elitista para preservar as melhores soluções. O terceiro algoritmo é um clássico utilizado em MOP, chamado Soma Ponderada, ele utiliza um método baseado em preferência para escalar as duas funções objetivo em uma função de objetivo simples.

4.3 Função de Aptidão para SPEA e NSGAII.

A seguir formulamos a função de aptidão utilizada no SPEA e no NSGAII, afim de dar sua forma vetorial representamos a primeira função objetivo como $\boldsymbol{F_{Q1}}(\boldsymbol{A})$ e a segunda como $\boldsymbol{F_{Q2}}(\boldsymbol{A})$. Como já foi mencionado cada uma destas funções estão

baseadas na função ***SP score*** tal como foi definida e exemplificada no capítulo 2. Escrevemos as duas funções objetivos como segue:

$$\boldsymbol{F}_{Q1}(\boldsymbol{A}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}-1} \sum_{\boldsymbol{j}=\boldsymbol{i}+1}^{\boldsymbol{n}} \boldsymbol{C}_{\boldsymbol{BLOSUM}\,62}(\boldsymbol{S}^{\boldsymbol{i}}, \boldsymbol{S}^{\boldsymbol{j}})$$

$$\boldsymbol{F}_{Q2}(\boldsymbol{A}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}-1} \sum_{\boldsymbol{j}=\boldsymbol{i}+1}^{\boldsymbol{n}} \boldsymbol{C}_{\boldsymbol{PAM}\,250}(\boldsymbol{S}^{\boldsymbol{i}}, \boldsymbol{S}^{\boldsymbol{j}})$$

A formulação vetorial para o problema de Alinhamento Múltiplo de Seqüências está dada por:

$$\boldsymbol{F}_{\boldsymbol{Q}}(\boldsymbol{A}) = \begin{bmatrix} \max \boldsymbol{F}_{Q1}(\boldsymbol{A}) \\ \\ \max \boldsymbol{F}_{Q2}(\boldsymbol{A}) \end{bmatrix}$$

4.4 Função de Aptidão para o Algoritmo Soma Ponderada.

Para o caso do algoritmo Soma Ponderada a função de aptidão é escalada a um único objetivo pre multiplicando cada função objetivo $\boldsymbol{F}_{Q1}(\boldsymbol{A})$ e $\boldsymbol{F}_{Q2}(\boldsymbol{A})$ por um valor chamado peso. Este peso é escolhido em proporção à importância relativa das FOs. A formulação para o problema de Alinhamento Múltiplo de Seqüências está dada por:

$$\text{maximizar } \boldsymbol{F}_{\boldsymbol{Q}}(\boldsymbol{A}) = \boldsymbol{P}_1\boldsymbol{F}_{Q1}(\boldsymbol{A}) + \boldsymbol{P}_2\boldsymbol{F}_{Q2}(\boldsymbol{A}),$$

onde P_1 e P_2 são às ponderações dadas às diferentes funções objetivo.

Variando os valores dos pesos poderemos encontrar soluções ótimas diferentes. Em nossa pesquisa os valores escolhidos para os testes foram $P_1 = P_2 = 0.5$.

4.5 Codificação do MSA de Proteínas.

Afim de implementar os algoritmos citados anteriormente, precisamos definir uma representação para o alinhamento, que será a representação das soluções que os algoritmos se encarregaram de evolucionar.

Dado que o alfabeto base das proteínas tem 20 letras e o alfabeto estendido 21 elementos incluindo o gap, codificamos as soluções como números inteiros. Mapeamos cada letra do alfabeto estendido $\mathcal{A}'$ com um vetor de números inteiros de 0 até 20, onde o gap é representado por 0, como é mostrado na tabela 4.1.

A	R	N	D	E	C	G	Q	H	I	L	K	M	F	P	S	Y	T	W	V	-
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	0

Tabela 4.1: mapeio das letras do alfabeto estendido.

Exemplo 4.1: A seqüência S^1: NLF-VAT, sua representação será:

N	L	F	-	V	A	T
3	11	14	0	20	1	18

Exemplo 4.2: Dada três seqüências X^1: ACRGTLR, X^2: DRSN e X^3: RTNR, e um possível alinhamento para as mesmas $S^1\#S^2\#S^3$, o mapeamento se dá a seguir:

S^1	A	C	R	G	T	L	R		1	6	2	7	18	11	2
S^2	-	D	R	-	S	N	-	→	0	4	2	0	16	3	0
S^3	-	-	R	-	T	N	R		0	0	2	0	18	3	2

4.6 Processo de Obtenção do Alinhamento Inicial

Dado um conjunto de seqüências não alinhadas, precisamos obter um alinhamento válido das mesmas, utilizando a codificação descrita anteriormente.

A fim de igualar o comprimento das seqüências a serem alinhadas, inserimos gaps no final das mesmas. A quantidade de gaps inseridos $\boldsymbol{l}$, limita o comprimento mais longo que os alinhamentos podem representar. É importante selecionar o valor de $\boldsymbol{l}$ que envolve o tamanho do espaço da busca dos alinhamentos. Se o valor de $\boldsymbol{l}$ é muito pequeno, alinhamentos ótimos de seqüências de menor similaridade não poderão ser encontrados. Por outro lado, se o valor de $\boldsymbol{l}$ é muito grande, precisaremos de mais tempo para encontrar o ótimo alinhamento de seqüências com alta similaridade.

Consequentemente definimos um valor que chamamos ***taxa de gaps*** para determinar o valor de $\boldsymbol{l}$. Então, se o comprimento mais longo da seqüência a ser alinhada é $\boldsymbol{m_{max}}$, então

$$\boldsymbol{l} = \boldsymbol{m_{max}} \text{ x } (1 + \textbf{\textit{taxa de gaps}})$$

A continuação mostramos um exemplo.

Exemplo 4.3: Processo de obtenção do alinhamento inicial.

$\boldsymbol{m_{max}} = 8$, taxa de gaps = 0.2 então $\boldsymbol{l}$ = (inteiro) (8 x (1+0.2)) = **9.6 → 9**

4.7 Processo de Obtenção da População Inicial.

Cada elemento da população inicial de tamanho N será obtido a partir do alinhamento inicial aplicando deslocamento aleatório dos gaps inseridos ao final de cada seqüência no alinhamento.

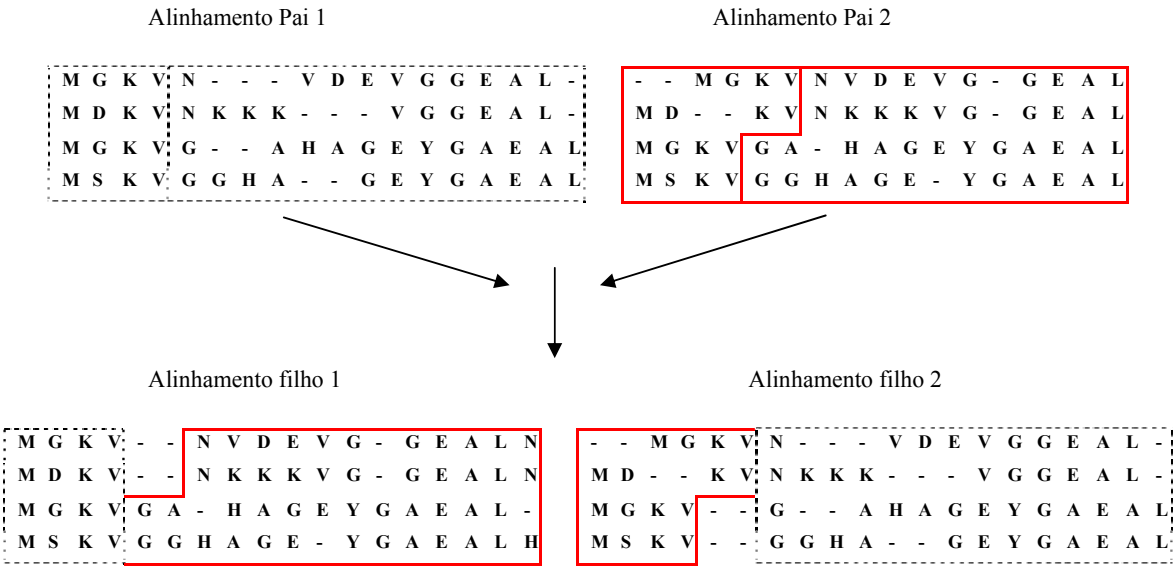
A geração de alinhamentos iniciais é um importante passo para obter o alinhamento final melhorado. Uma boa solução inicial pode efetivamente convergir mais rápido e melhorar o custo computacional.

4.8 Operador de Cruzamento

O operador de cruzamento é o responsável de combinar dois alinhamentos diferentes afim de obter dois novos alinhamentos. Ao realizar esta combinação precisamos que o operador de cruzamento preserve a validade dos alinhamentos.

O tipo usado é *one-point*, que combina os alinhamentos através de um intercâmbio simples. A seguir mostramos um exemplo.

Exemplo 4.4: As linhas tracejadas indicam a posição de corte realizada no pai1, a posição de corte é escolhida aleatoriamente, depois se calcula a posição de corte no pai2 por cada seqüência, de modo a não perder nenhuma unidade, como indicam as linhas continuas, depois combinam-se as partes para produzir os novos filhos. Os espaços são preenchidos com gaps.



O procedimento utilizado é descrito no algoritmo 4.1, que é mostrado a seguir.

Algoritmo 4.1: Procedimento cruzamento (Pai1, Pai2)

Inicio

Seja m_1 e m_2 o comprimento do alinhamento do Pai1 e Pai2 respectivamente, n o número de seqüências alinhadas.

1. No Pai1 escolher aleatoriamente uma posição k entre 1 a m_1 onde o corte será realizado.
   Para cada seqüência em Pai1
      Desde a posição 1 a k fazer
         Contar a quantidade de unidades por cada seqüência e guardar em (a_1, a_2,… a_n).
2. No Pai2 por cada seqüência procurar a posição que corresponde a (a_1, a_2,… a_n) e guardar em (b_1, b_2,… b_n). A fim de que não se perda nenhuma unidade no pai2.
3. Formar o filho1, cujo comprimento m_a será k + $\max_i(m_2 - b_i)$.
   Desde i=1 a i=n fazer
      Copiar a filho1:
         a. As subunidades de Pai1 desde a posição 1 a k
         b. Inserir m_a-(m_2-b_i) gaps
         c. As subunidades de Pai2 desde a posição b_i a m_2
   Fim Desde
4. Formar o filho2, cujo comprimento m_b será $\max_i b_i$ + (m_1-k).
   Desde i=1 a i=n fazer
      Copiar a filho2:
         a. As subunidades de Pai2 desde a posição 1 a b_i
         b. Inserir m_b-(m_1-k) gaps
         c. As subunidades de Pai1 desde a posição k a m_1
   Fim Desde

Fim

4.9 Operadores de Mutação.

Como no caso do operador de Cruzamento, também nos operadores de mutação devemos ter cuidado a fim de preservar a validade dos alinhamentos analisados.

O processo de mutação é mostrado no Algoritmo 4.2. Aplicamos duas classes de mutação, chamados ***união de gaps*** e ***deslocamento de gaps***. No processo de mutação, uma ou mais seqüências de um indivíduo de entrada são mutadas por um dos operadores selecionados aleatoriamente desde os dois operadores de mutação. Os dois operadores têm igual freqüência de seleção.

Operador de união de gaps: Este operador escolhe alguns espaços (dois ou três) de uma seqüência e os une em outra posição da seqüência. (Algoritmo 4.3)

Operador de gap de deslocamento: gaps no pai são aleatoriamente deslocados para produzir um novo *offspring*. (Algoritmo 4.4)

Algoritmo 4.2: Procedimento de Mutação (indivíduo A)

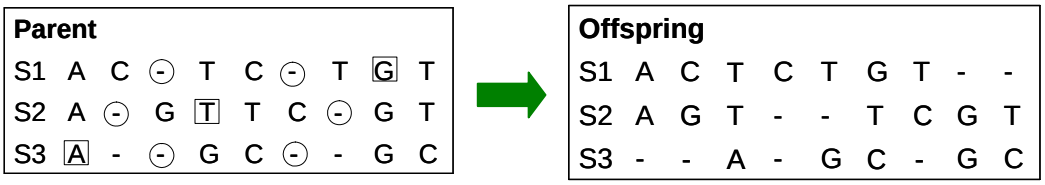
```
Inicio
Desde i=1 a popsize fazer
        op = Aleatoriamente selecionar um operador
        Se(op=1) então união_gaps(A);
        senão deslocamento_gaps(A);
        Apagar_colunas_gaps(A);
Fim Desde
Fim
```

Algoritmo 4.3: Procedimento união_gaps(indivíduo A)

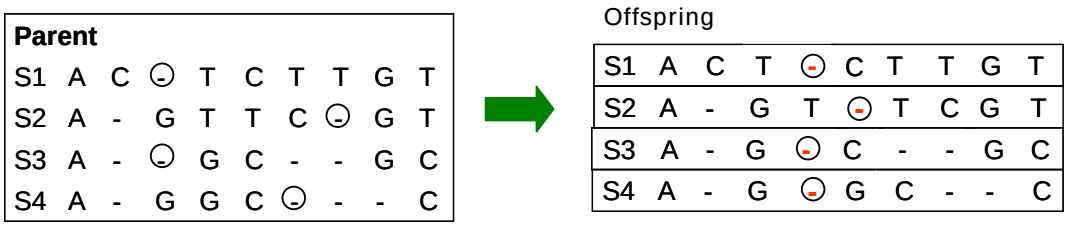
```
Inicio
Escolher n seqüências de A a ser mutadas
Desde seqüência=seq_1 a seq_n fazer
        Escolher aleatoriamente duas posições que contêm gaps
        Escolher aleatoriamente uma posição aonde vão se destinar os gaps
        Realizar a substituição
Fim Desde
Fim
```

Exemplo 4.5: Operador de união de gaps aleatório, os círculos mostram os gaps escolhidos aleatoriamente e o quadrado mostra a posição onde esses gaps serão

destinados. O *Offspring* mostra o resultado final da mutação. Neste exemplo todas as seqüências são mutadas, mas pode ser que só algumas delas sejam escolhidas.



Exemplo 4.6: Operador gap de deslocamento, os círculos no pai mostram a posição inicial do gap o qual será deslocado e os círculos no *offspring* mostram a posição onde o gap foi alocado.



Algoritmo 4.4: Procedimento gap_deslocamento (indivíduo A)

Inicio

Seja *m* o comprimento do alinhamento A, *n* a quantidade de seqüências alinhadas e s_j^i a subunidade j da seqüência i.

Desde i=1 a i=*n* fazer

Escolher duas posições aleatoriamente entre 1 a *m*, k_1 e k_2 tal que, $k_1 \neq k_2$ e k_1 sejam a posição de um gap.

Se $k_1 < k_2$

Desde j= k_1 a j= k_2-1 fazer

Copiar $s_j^i = s_{j+1}^i$

j=j+1

Fim Desde

$s_j^i = 0$

Senão

Desde j= k_1 a j= k_2+1 fazer

Copiar $s_j^i = s_{j-1}^i$

j=j-1

Fim Desde

$s_j^i = 0$

Fim Se

i=i+1

Fim Desde

Fim

4.10 Modelo Paralelo aplicado aos MOEAs SPEA e NSGAII.

O modelo de paralelização proposto funciona com sub-populações independentes, conhecido como modelo de ilhas (Apêndice C), cujo esquema aplicado a um algoritmo evolutivo multi-objetivo se apresenta no Algoritmo 4.5.

Emigrantes denota o conjunto de indivíduos a serem trocados com outra ilha, estes são selecionados de acordo com uma política determinada por *SeleçãoMigração*, eventualmente diferente da utilizada para selecionar indivíduos para o processo de reprodução. O operador Migração intercambia os indivíduos entre ilhas de acordo com um grafo de conectividade definido entre eles, geralmente um anel unidirecional. A *CondiçãoMigração* determina quando se realiza o intercambio de indivíduos. Os indivíduos Imigrantes são inseridos na população destino, eles intercambiam aos indivíduos locais de acordo a uma política de intercambio determinada.

Algoritmo 4.5: Esquema de um Algoritmo Evolutivo Paralelo Multi-objetivo de População Distribuída.

```
Inicializar(P(0))
geração = 0
Avaliar(P(0))
quando(não CritérioParada) fazer
        Operador de diversidade(P(geração))
        Calcular fitness(P(geração))
        Pais = Seleção(P(geração))
        Filhos = Operadores de Reprodução(Pais)
        NovaPop = Intercambiar(Filhos, P(geração))
        geração ++
        P(geração) = NovaPop
        Se (CondiçãoMigração)
        Emigrantes = SeleçãoMigração(P(geração))
        Imigrantes = Migração(Emigrantes)
        Inserir(Imigrantes, P(geração))
retornar Melhor Solução Achada
```

Seguindo o esquema apresentado definiu-se um operador de migração para comunicar em forma síncrona os processos que evoluem em paralelo. Adota-se uma topologia de migração considerando os processos conectados em anel unidirecional.

A evolução de cada sub-população finaliza ao alcançar a condição determinada pelo critério de parada. Nesse momento, cada ilha envia a totalidade do frente de indivíduos não dominados de sua população a uma ilha distinguida que age como receptora dos indivíduos não dominados do resto das ilhas. O número de gerações antes da interação é um parâmetro do modelo paralelo implementado, sugere-se utilizar em geral um valor reduzido para manter as características do modelo distribuído.

Operadores

O desenho dos operadores utilizados no processo de migração foi baseado nas sugestões de Veldhuizen et al. (2003), onde se aplica uma estratégia de seleção elitista aleatória para a eleição de indivíduos emigrantes, isto implica selecionar aleatoriamente um conjunto de indivíduos não dominados, de cardinalidade igual a certo porcentual do tamanho da população. O tamanho do conjunto de emigrantes especifica-se como parâmetro do algoritmo, mas a idéia é utilizar valores pequenos para este parâmetro.

Mediante o uso desta estratégia de eleição de emigrantes tentamos obter uma pressão seletiva "relativamente alta" tentando lograr um compromisso que permita às subpopulações evoluírem de modo semi-independente, evitando convergência para o mesmo conjunto de pontos não dominados, mas permitindo a cooperação entre subpopulações, determinada pelo intercâmbio ocasional de um conjunto reduzido de indivíduos bem adaptados para a resolução do problema.

Em relação à política de intercâmbio, aplicamos novamente uma estratégia elitista, que consiste em trocar os indivíduos dominados da população destino pelos imigrantes recebidos. De este modo, trata-se de manter os indivíduos não dominados, determinados no processo de busca da sub-população destino, tratando de obter uma melhor convergência para o frente de Pareto. Levando-se em conta que em problemas contínuos "simples" as subpopulações obtêm rapidamente um conjunto de pontos não dominados que abarca o total da população, uma segunda estratégia de intercambio pode ser aplicada em estes casos. Esta política de intercâmbio utiliza como critério a distância de crowding calculada pelo algoritmo, apagando indivíduos com valores de distância de crowding pequenos, considerando que existem outros indivíduos representativos dessa seção do espaço fenotípico.

Os mecanismos de migração são executados em forma assíncrona, já que as operações de envio e recepção de indivíduos não bloqueiam a execução do algoritmo, possibilitando que continue a evolução em cada sub-população. De acordo a capacidade de processamento dos processos. O intercambio poderá ser realizado em diferentes gerações nas distintas subpopulações.

4.11 Modelo Paralelo aplicado ao Algoritmo Soma Ponderada

No modelo escolhido para este algoritmo cada processador gera sua própria população inicial. Para a comunicação dos resultados intermédios entre os processadores, utilizamos um esquema de anel unidirecional: após um período pré-determinado de computação, cada processador envia para o seu vizinho seu melhor indivíduo, isto é, a melhor solução encontrada até o momento. Este modelo segue a idéia de ilhas de migrações (Apêndice C).

A comunicação dá-se através de *Receives* síncronos. Cada processador envia sua mensagem e espera pela mensagem do próximo. Sendo a carga e os processadores uniformes, isto não constitui um problema: os processadores chegam aproximadamente ao mesmo tempo no ponto de sincronização e, desta forma, não esperam muito pelas mensagens dos seus respectivos vizinhos.

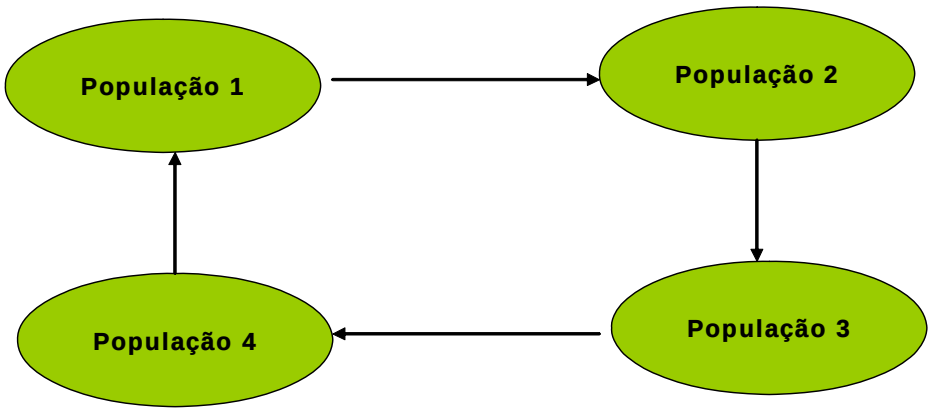


Figura 4.1: Comunicação em anel.

4.12 Considerações Gerais

A vantagem deste método é que pode ser estendido no caso que se queiram adicionar novas funções objetivo, como por exemplo COFFEE, ou alguma outra função que se queira usar para medir a qualidade de um alinhamento. Também, para melhorar a rapidez de convergência das soluções é possível utilizar como alinhamento inicial um alinhamento obtido através de outro método, por exemplo CLUSTALW que provê soluções razoavelmente boas com rapidez.

Usando SP score como função de qualidade podemos utilizar um conjunto de matrizes de substituição que melhor se adaptam ao conjunto de seqüências que queremos analisar.

CAPÍTULO

5

EXPERIMENTOS E RESULTADOS

Apresentamos neste capítulo os resultados obtidos com o modelo descrito no capítulo anterior, nosso objetivo é encontrar um alinhamento ótimo ou quase-ótimo utilizando um modelo de valoração determinado. Detalhamos a seguir o entorno de programação, o conjunto de testes utilizado, a metodologia de avaliação e o analise dos resultados obtidos.

5.1 Conjunto de teste utilizado

O conjunto de seqüências utilizado nos testes foi obtido da base de dado de BAliBASE.

Conjunto	BBS11001	BBS11025	BBS11035	BBS12006	BBS12030
Número de seqüências	4	4	5	4	6
Tamanho do alinhamento	78	70	129	237	243
Tamanho da seqüência de maior comprimento	78	63	110	235	239
Tamanho da seqüência de menor comprimento	76	56	70	219	234
% de identidade	Seqüências muito divergentes (<20% identidade)			Seqüências meio até divergente (20-40% identidade)	

Tabela 5.1: Conjunto de seqüências desde a base de dado BAliBASE consideradas no teste.

5.2 Plataforma de testes utilizada

- Cluster SGI Altix 350(Saturn)
- 14 processadores Intel Itanium II (palavra de 64 bits) de 1.5 Ghz.
- 28 GB de RAM de memória (compartilhada NUMA) e armazenamento em disco de 360 Gbyte.
- Pico teórico 6 Gflops/s por CPU.
- Sistema operacional RedHat Enterprise Linux + SGI ProPack.
- Compilador Intel e GNU (Fortran-90 e C/C++) com suporte OpenMP e MPI

5.3 Método de avaliação dos resultados

Afim de avaliar a qualidade do método proposto, utilizou-se como referência o Benchmark BAliBASE [THOMPSON, J.D et al, 1998]. As soluções obtidas através de nosso método foram comparadas com as soluções do benchmark.

O BAliBASE é uma base de dados de alinhamentos múltiplos de seqüências de proteínas criada especificamente para a realização de benchmarks de qualidade. Todos os seus alinhamentos foram construídos manualmente a partir da comparação da estrutura de proteínas. Os alinhamentos do BAliBASE são divididos em grupos de características semelhantes que representam problemas comuns encontrados no alinhamento de seqüências de proteínas.

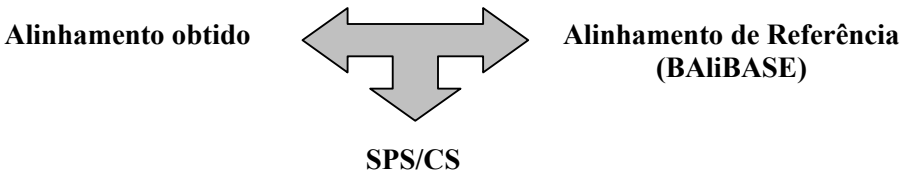
O benchmark provê um programa chamado “bali_score” que compara um alinhamento de seqüências de proteínas da base de dados BAliBASE com um outro alinhamento que se deseja avaliar das mesmas seqüências.

O bali_score atribui um valor para cada coluna do alinhamento sendo avaliado, e resume tudo em um valor porcentual. Teoricamente um alinhamento que obtiver um valor de 100% pode ser considerado muito bom, ao passo que um valor de 0% indica

um alinhamento de péssima qualidade. Os valores de qualidade obtidos com bali_score são **Sum-of-pair score** (SPS) e **Column Score** (CS):

Sum-of-pair score (SPS): conta quantos pares de subunidades foram corretamente alinhadas, comparando com o alinhamento de referência de BAliBASE.

Column score (CS): é uma pontuação binária que testa a habilidade dos programas para alinhar todas as seqüências corretamente.



Obtivemos os alinhamentos da base de dados do BAliBASE considerados em nosso conjunto de teste, para cada um deles calculamos as Funções Objetivo utilizadas (F_{Q1} e F_{Q2}), desta forma foi possível comparar a qualidade dos alinhamentos da base de dados de BAliBASE com a qualidade dos alinhamentos obtidos com nosso modelo.

Também comparamos nossos resultados com os obtidos por outros softwares freqüentemente utilizados em MSA, para a obtenção destes programas referenciamos a European Bioinformatics Institute (www.ebi.ac.uk), que permite um livre acesso a vários softwares de avaliação de seqüências de biomoléculas através de seu site na Internet. Os resultados obtidos são mostrados nas tabelas 5.2 e 5.3.

5.4 Resultados Obtidos

A seguir apresentamos os resultados obtidos em nossa avaliação. A tabela 5.2 mostra por cada conjunto testado o algoritmo utilizado, os valores obtidos em cada Função objetivo, o comprimento do alinhamento resultante, os valores obtidos em SPS e CS, e o tempo de execução do algoritmo.

Sendo que os algoritmos NSGA2 e SPEA podem obter mais de uma solução denotamos cada uma destas com o nome do algoritmo seguido do número de solução, por exemplo NSGA2_1 denota a solução 1 do algoritmo NSGA2.

Conjunto	Algoritmo	FQ_1	FQ_2	Comp. Alinham.	SPS	CS	Tempo (segundos)
BBS11001	BAliBASE	420.4	333.4	78			
	NSGA2_1	**420.39**	**333.39**	**78**	**1**	**1**	**26.218**
	NSGA2_2	421.39	332.39	78	0.987	0.98	26.218
	NSGA2_3	422.39	331.39	78	0.987	0.98	26.218
	SPEA_1	275	182	78	0.928	0.86	99.038
	Soma Pon.	286	189	78	0.928	0.86	73.094
BBS11025	BAliBASE	-148.4	-199.4	70			
	NSGA2_1	***-122***	***-182***	**70**	**0.503**	**0.33**	**162.5**
	NSGA2_2	***-127.6***	***-169.6***	70	0.493	0.33	162.5
	NSGA2_3	***-117***	***-186***	70	0.464	0.25	162.5
	NSGA2_4	***-116***	***-199***	68	0.458	0.33	162.5
	NSGA2_5	***-122.6***	***-173.6***	70	0.454	0.25	162.5
	NSGA2_6	-113.6	-204.6	66	0.435	0.25	162.5
	SPEA_1	-328	-378	65	0.412	0.19	59.929
	Soma Pon.	-471	-554	66	0.395	0.27	229.3
BBS11035	BAliBASE	187.8	-5.2	129			
	NSGA2_1	**-59**	**-264**	**118**	**0.335**	**0.12**	**189.3**
	NSGA2_1	-60	-260	118	0.33	0.12	189.3
	SPEA_1	-353.6	-508.6	115	0.256	0.1	290.339
	SPEA_2	-400.8	-499.8	115	0.250	0.1	290.339
	Soma Pon.	-767.4	-820.4	118	0.18	0.0	374.375
BBS12006	BAliBASE	1905.8	1615.8	237			
	NSGA2_1	**821.4**	**682.4**	**237**	**0.802**	**0.60**	**785.5**
	SPEA	761.6	530.6	237	0.702	0.54	633.35
	Soma Pon.	134.4	-92.6	237	0.334	0.33	433.25

Conjunto	Algoritmo	FQ_1	FQ_2	Comp. Alinham.	SPS	CS	Tempo (segundos)
BBS12030	BAliBASE	4301.4	3670.4	243			
	NSGA2_1	***4821.2***	***4171.2***	**243**	**0.811**	**0.58**	**984**
	NSGA2_2	***4815.2***	***4179.2***	243	0.811	0.58	984
	NSGA2_3	***4847.2***	***4120.2***	243	0.810	0.58	984
	SPEA_1	1510	787	243	0.608	0.27	606.516
	SPEA_2	1441	821	243	0.601	0.27	606.516
	Soma Pon.	896	679	243	0.428	0.16	703.094

Tabela 5.2: Comparação dos resultados obtidos com o benchmark para o grupo de seqüências testados. A melhor solução obtida em cada conjunto de teste é apresentado em negrito. As soluções que superam em qualidade as soluções do benchmark são apresentadas em negrito e itálica.

A seguir apresentamos uma comparação do método proposto com outros softwares utilizados em MSA. Estes softwares são amplamente utilizados em problemas de alinhamento, eles são ClustalW, T-coffee, Kalign e Maftt. Utilizamos como parâmetro de comparação os valores obtidos com SPS e CS (benchmark BAliBASE) para cada solução desde os softwares. As soluções foram obtidas executando os programas desde o site http://www.ebi.ac.uk

Conjunto	NSGAII SPS/CS	SPEA SPS/CS	Soma Pon SPS/CS	ClustalW SPS/CS	T-coffee SPS/CS	Kalign SPS/CS	Mafft SPS/CS
BBS11001	**1.00/1.00**	0.928/0.86	0.928/0.86	0.962/0.94	0.961/0.94	**1.00/1.00**	0.961/0.94
BBS11025	**0.503/0.33**	0.412/0.19	0.395/0.27	0.15/0.00	0.402/0.07	**0.624/0.41**	0.435/0.25
BBS11035	**0.335/0.12**	0.256/0.1	0.18/0.0	**0.518/0.34**	0.491/0.34	0.439/0.24	0.444/0.3
BBS12006	**0.802/0.6**	0.702/0.54	0.334/0.33	0.869/0.82	0.880/0.83	**0.940/0.89**	0.888/0.84
BBS12030	**0.811/0.58**	0.606/0.27	0.428/0.16	**0.931/0.88**	0.911/0.84	0.905/0.82	0.902/0.82

Tabela 5.3: Tabela comparativa do método proposto com outros softwares utilizados em MSA. Na segunda coluna da tabela são mostrados os melhores resultados obtidos

de acordo aos valores de SPS e CS em cada algoritmo proposto. Na terceira coluna são apresentados os valores de SPS e CS para as soluções obtidas nos softwares ClustalW, T-coffee, Kalign e Mafft respectivamente. Os melhores valores por cada conjunto de teste são apresentados em negrito.

5.5 Análise dos resultados obtidos

As soluções finais apresentam uma variedade de formas em que um conjunto de seqüências pode ser alinhado, o que pode ser visto também como a procura de diferentes posições onde podem ser encontradas similaridades em maior o menor grau. Isto pode ser observado nas Figuras 5.1 e 5.2 que mostram as soluções do Frente de Pareto encontradas pelo algoritmo NSGA2 para o conjunto de seqüências BBS11001 e BBS11025 respectivamente.

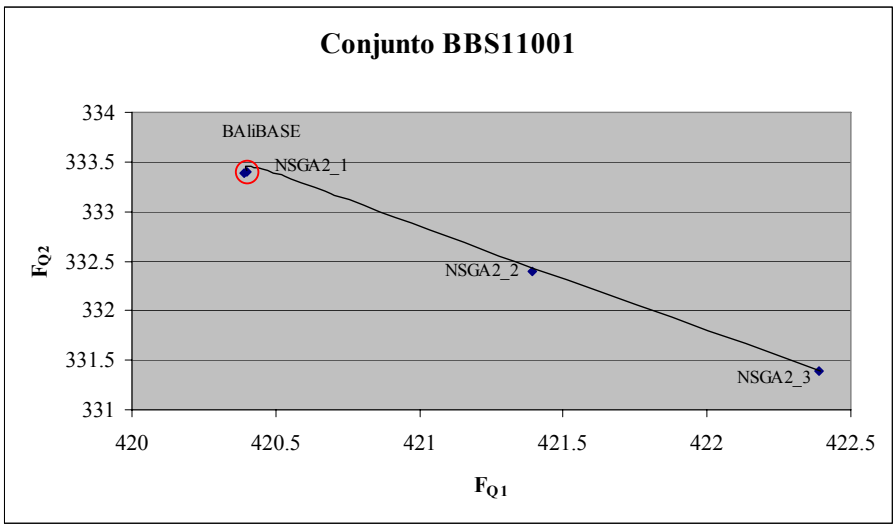


Figura 5.1: Frente de Pareto das soluções do NSGA2 para o conjunto BBS11001. O gráfico mostra o conjunto de soluções ótimas obtidas por o algoritmo NSGA2. A solução Balibase é mostrada com um circulo.

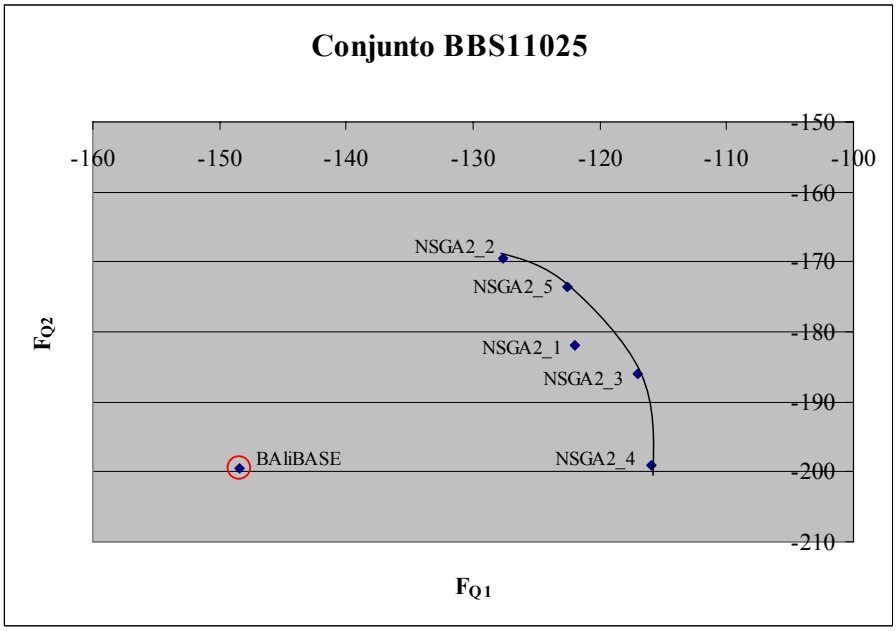


Figura 5.2: Frente de Pareto das soluções do NSGA2 para o conjunto BBS11025. O gráfico mostra o conjunto de soluções ótimas obtidas por o algoritmo NSGA2. A solução Balibase é mostrada com um circulo. O conjunto de 5 soluções mostrados formam o conjunto de soluções não dominadas e todas elas dominam a solução Balibase.

Na tabela 5.2 e na Figura 5.1 pode-se observar que no conjunto de seqüências testado BBS11001, o NSGA2 obteve três soluções, o NSGA2_1 com valor exatamente igual ao benchmark e as soluções NSGA2_2 e NSGA2_3 que mostram uma qualidade maior na Função Objetivo F_{Q1}. O SPEA e o Soma Ponderada obtiveram soluções que são consideradas muito boas em relação ao benchmark e apesar de que seus valores de F_{Q1} e F_{Q2} são diferentes eles têm o mesmo valor de SPS e CS, isto demonstra que em relação ao benchmark estas soluções são igualmente boas.

No conjunto BBS11025, o NSGA2 não apresenta soluções com valores iguais ao benchmark, mas estas soluções têm melhor qualidade considerando F_{Q1} e F_{Q2}, a exceção é a solução NSGA2_6 cujo valor é levemente menor em F_{Q2}.

As soluções para os conjuntos BBS11035 e BBS12006, ficaram abaixo do benchmark. Isto se deve a que utilizamos um esquema de valoração muito simples, não consideramos alguns fatores muito importantes como por exemplo similaridade das

seqüências e uso de outras matrizes de substituição, critérios que poderiam ser mais apropriados para o conjunto de seqüências testados.

As soluções obtidas por NSGA2 para o conjunto BBS12030 apresentam uma qualidade maior considerando F_{Q1} e F_{Q2} que o Balibase.

Comparado a outros programas usados na resolução de MSA, o método apresentado neste trabalho obtém maior variedade de soluções finais devido à aplicação de critérios de Otimização Multi-Objetivo, cuja característica é justamente brindar um conjunto de alternativas factíveis para um problema.

A tabela 5.3 apresenta uma comparação das soluções que obtiveram o melhor valor SPS e CS para cada algoritmo implementado e as soluções obtidas pelos softwares ClustalW, T-coffee, Kalign e Mafft.

Os três algoritmos implementados obtêm soluções tão boas quanto aos outros softwares para o conjunto BBS11001. No caso do conjunto BBS11025 o NSGA2 e o SPEA mostram resultados muito interessantes, em especial o NSGA2 que só fica abaixo do valor obtido por Kalign.

Nenhum dos três algoritmos superam os softwares comparados nos conjuntos BBS11035 e BBS12006, como já dizemos anteriormente é preciso considerar outros critérios afim de melhorar estes resultados.

Análise de desempenho

Este análise tem por objetivo estabelecer uma comparação do desempenho dos algoritmos utilizados neste trabalho de pesquisa para resolver o problema de MSA, em termos de velocidade de processamento.

As comparações são efetuadas utilizando os tempos de processamento dos três algoritmos implementados e os conceitos de aceleração e eficiência, comumente usados na literatura como medidas de desempenho de algoritmos paralelos. Para este fim os três algoritmos considerados foram executados utilizando 1, 2, 4 e 8 processadores da máquina paralela. Os tempos de processamento necessários para obter as soluções do conjunto de seqüências BBS11001 são mostrados na Tabela 5.4. Nota-se que os tempos de processamento dos algoritmos decresce à medida que aumenta o número de processadores.

Tempo de processamento. Conjunto BBS11001 (segundos)				
Algoritmo	**1 proc**	**2 proc**	**4 proc**	**8 proc**
Soma Ponderada	239.094	73.094	26.062	9.438
SPEA	430.428	99.038	30.75	13.212
NSGA2	332.625	90	26.218	8.28

Tabela 5.4: Tempos de processamento para o conjunto BBS11001 em relação ao número de processadores utilizados nos algoritmos.

As soluções finais obtidas pelos algoritmos não se mantiveram quando incrementamos o número de processadores. A seguir, na Tabela 5.5 são mostradas as soluções de cada algoritmo em relação ao número de processadores utilizados para o conjunto BBS11001. Nota-se que existe uma tendência decrescente de qualidade a medida que aumentamos o número de processadores.

Soluções Finais encontradas para BBS11001				
Soluções Finais	**1 proc** F_{Q1}/F_{Q2}	**2 proc** F_{Q1}/F_{Q2}	**4 proc** F_{Q1}/F_{Q2}	**8 proc** F_{Q1}/F_{Q2}
Soma Ponderada	286/189	286/189	280.8/163.8	273.4/178.4
SPEA_1	275/182	275/182	237.4/151.4	262.4/171.4
SPEA_2			255/138	
NSGA2_1	420.39/333.39	420.39/333.39	420.39/333.39	420.39/333.39
NSGA2_2	421.39/332.39	421.39/332.39	421.39/332.39	421.39/332.39
NSGA2_3	422.39/331.39	422.39/331.39	422.39/331.39	

Tabela 5.5: Soluções finais encontradas em relação ao número de processadores para o conjunto BBS11001.

Aceleração do conjunto BBS11001				
Algoritmo	**1 proc**	**2 proc**	**4 proc**	**8 proc**
Soma Ponderada	1.00	1.63	2.37	3.15
SPEA	1.00	1.55	2.97	4.22
NSGA2	1.00	1.91	3.44	5.21

Eficiência para o conjunto BBS11001				
Algoritmo	**1 proc**	**2 proc**	**4 proc**	**8 proc**
Soma Ponderada	100.00%	81.70%	59.22%	39.43%
SPEA	100.00%	77.52%	74.17%	52.74%
NSGA2	100.00%	95.27%	86.04%	65.10%

Tabela 5.6 : Medidas de desempenho da implementação paralela dos algoritmos utilizados neste trabalho para o conjunto BBS11001.

O ganho obtido ao paralelizar os algoritmos foi medido através da aceleração e da eficiência, definidos no Apêndice C. A Tabela 5.6 e a Figura 5.3 mostram os valores da aceleração e da eficiência do conjunto BBS11001 obtida pelos algoritmos implementados neste trabalho em relação a sua execução com 1 processador.

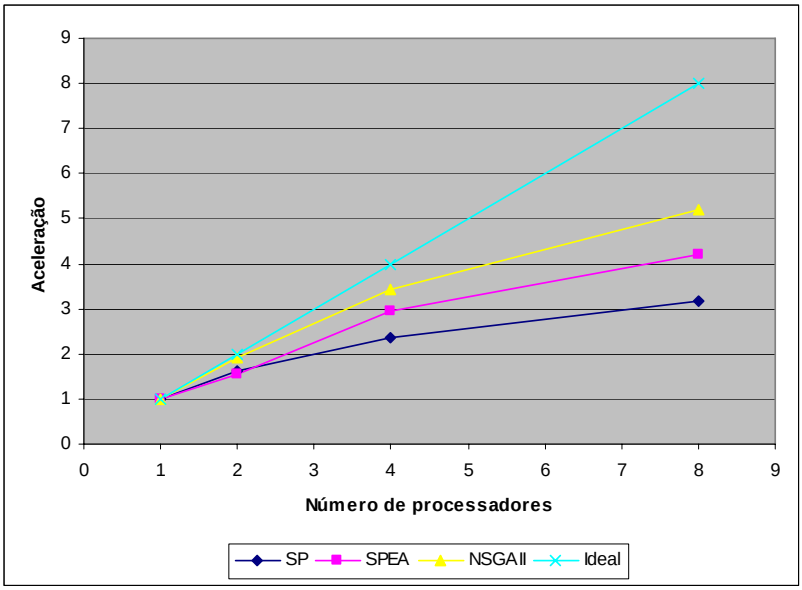


Figura 5.3 (a): Curva de aceleração dos algoritmos implementados para o conjunto de seqüências BBS11001

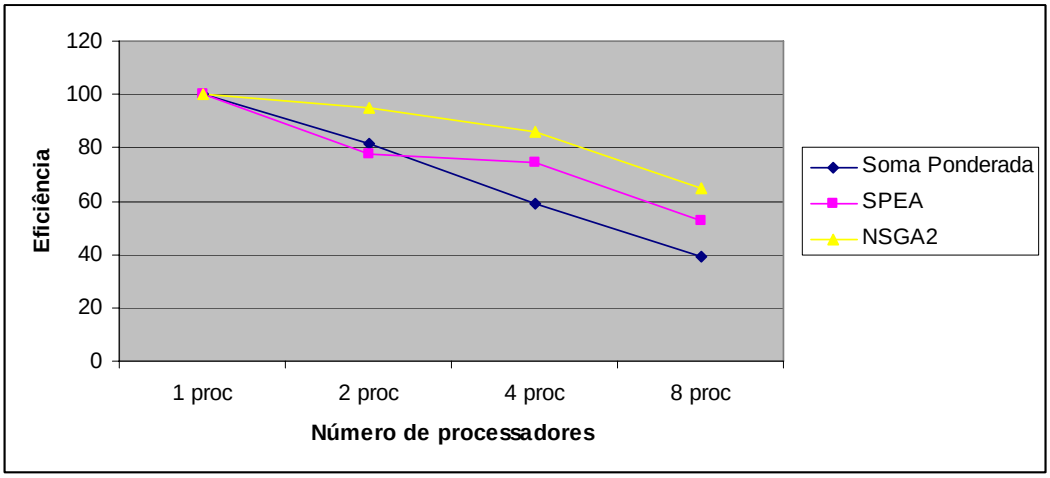


Figura 5.3 (b): Curva de Eficiência para os algoritmos implementados para o conjunto de seqüências BBS11001.

Na Figura 5.3 (a) é mostrada a curva de aceleração, indicando que o ganho obtido pela paralelização do algoritmo em relação à implementação com 1 processador é significativo. No entanto, os valores de aceleração estão muito por abaixo dos valores ideais, que são os da aceleração linear.

A curva de eficiência é mostrada na Figura 5.3 (b), os algoritmos apresentam uma tendência de decréscimo na medida em que o processamento é realizado com mais processadores o que indica que a fração de tempo utilizada em tarefas de sincronização e comunicação de processos aumenta rapidamente para este algoritmo no conjunto testado. Observe também que os valores obtidos da eficiência, estão muito por abaixo do valor ideal.

5.6 Conclusão do Trabalho

O método utilizado demostrou ser um otimizador eficiente, uma vez determinado o domínio do problema de MSA. Para determinar este domínio foi necessário realizar um processo de testes e erros que nos levou ao ajuste dos parâmetros utilizados. A dificuldade no ajuste empírico dos parâmetros foi que o tempo investido foi muito grande já que não tivemos a intervenção de um biólogo.

O principal inconveniente na resolução do MSA é que não existe uma função objetivo que nos permita obter um alinhamento de tal modo que o alinhamento matemático ótimo coincida com o alinhamento biológico ótimo. Esta pesquisa utiliza duas funções objetivo para obter o alinhamento matemático ótimo, a função SP score calculada com a matriz BLOSUM e a função SP calculada com a matriz PAM.

As soluções obtidas e comparadas com o benchmark BAliBASE demostram que nossos alinhamentos resultantes em alguns casos coincidem com o benchmark e em os outros casos são tão bons quanto os softwares freqüentemente utilizados em MSA, como é mostrado nas tabelas 5.2 e 5.3.

O problema de MSA é NP-completo, o que significa que o tempo requerido para computar uma solução ótima aumenta exponencialmente com o tamanho do problema. A velocidade de convergência dos algoritmos pode ser melhorada utilizando como alinhamento inicial um pré-alinhamento que pode ser obtido utilizando um método que proveja soluções razoáveis em pouco tempo, como por exemplo CLUSTALW, este critério pode ser adotado no caso de seqüências de grande comprimento (superior a 300). Foi precisso a paralelização dos algoritmos para melhorar o tempo requerido de processamento.

Outra consideração acerca do método proposto é que as soluções obtidas com os MOEAs baseados em Pareto não foram muito diversas, isto se deve ao fato que as funções objetivo utilizadas são muito correlacionadas. Usando funções objetivo conflitantes ou procurando por matrizes de substituição não correlacionadas poderíamos superar este inconveniente.

Para finalizar, podemos dizer que os MOEAs são métodos eficientes de otimização e podem ser utilizados em problemas de MSA, quando o domínio do problema é determinado, isto foi suficientemente demonstrado pelos resultados obtidos.

5.7 Perspectiva Futura

Este trabalho pode ser o ponto de partida para outras pesquisas utilizando MOEAs como um método de resolução do problema de MSA, pois permitiu o conhecimento de alinhamentos interessantes que não foram possíveis obter mediante o uso das técnicas tradicionais. Os trabalhos futuros sugeridos são:

1. Obter alinhamentos com maior número de seqüências e de maior comprimento. Neste trabalho os experimentos foram feitos com seqüências pequenas para facilitar a análise dos resultados, mas as seqüências reais normalmente são bem maiores.

2. Utilizar Funções Objetivo baseados em consistência. A determinação do conjunto de matrizes de substituição e das penalidades para os gaps a utilizar no cálculo da maioria das Funções Objetivo para alinhamento de seqüências sempre é uma grande dificuldade. Uma alternativa é utilizar Funções Objetivo baseados em consistência, como a função COFFEE (Capítulo 2.8), de forma a obter os valores de substituição de aminoácidos utilizando o próprio conjunto de seqüências cujo alinhamento se quer obter. Usando este tipo de função não precisamos da penalização extra dos gaps porque já está incluso no modelo.

3. Utilizar Simulação baseada em MOP. O problema de MSA é computacionalmente custoso, pois para garantir uma solução ótima é necessário uma busca exaustiva, na qual todas as soluções possíveis são tratadas e avaliadas. Uma prática muito comum e aceitável é sacrificar a ótimalidade por eficiência heuristicamente guiando a busca e avaliando somente uma fração de todas as configurações.

   A Simulação baseada em MOP envolve interação direta com o tomador de decisões durante o processo de busca. O ciclo de tomada de decisão e busca ou busca e tomada de decisão, é realizado na obtenção de soluções aceitáveis. Os múltiplos objetivos são manipulados no processo de otimização considerando a preferência do tomador de decisões.

   A eficiência do processo de otimização é realçada realizando a exclusão das soluções antes de usar o modelo da simulação, evitando as soluções pouco promissórias que serão processadas desnecessariamente.

APENDICE

A

INTRODUÇÃO À BIOLOGIA MOLECULAR

Neste capítulo descrevemos alguns conceitos básicos da Biologia Molecular, importantes para o entendimento deste trabalho.

A.1 Conceitos Básicos de Biologia molecular

A.1.1 Introdução Histórica

Genética como um conjunto de princípios e procedimentos analíticos não começou até 1866, quando um monge Augustiniano chamado Gregor Mendel realizou um conjunto de experimentos que apontou à existência de elementos biológicos chamados genes, a unidade básica responsável pela possessão e passo de uma simples característica. Até 1944, foi geralmente assumido que cromossomas de proteínas carregavam informação, e que o DNA jogava um rol secundário. Avery and McCarty demonstraram que a molécula *deoxy-ribonucleic acid* (DNA) é o maior carregador de material genético nos organismos viventes, por exemplo é responsável da herança . Em 1953 James Watson e Francis Crick deduziram três estruturas dimensionais de DNA e imediatamente inferiram seu método de replicação.

O Projeto Genoma Humano (PGH) teve por objetivo o mapeamento do genoma humano, e a identificação de todos os nucleótidos que o compõem. Consistiu em um esforço mundial para se decifrar o genoma. Após a iniciativa do National Institutes of Health (NIH) estadunidense, centenas de laboratórios de todo o mundo se uniram à tarefa de seqüenciar, um a um, os genes que codificam as proteínas do corpo humano e também aquelas seqüencias de DNA que não são genes. Laboratórios de países em desenvolvimento também participaram do empreendimento com o objetivo de formar mão-de-obra qualificada em genômica.

Para o seqüenciamento de um gene, é necessário que ele seja antes amplificado numa reação em cadeia da polimerase, e então clonado em bactérias. Após a obtenção

de quantidade suficiente de DNA, executa-se uma nova reação em cadeia, desta vez utilizando didesoxirribonucleotídeos marcados com fluoróforos para a determinação da seqüência.

O projeto foi fundado em 1990, com um financiamento de 3 milhões de dólares do Departamento de Energia dos Estados Unidos e dos Institutos Nacionais de Saúde dos Estados Unidos, e tinha um prazo previsto de 15 anos. Devido à grande cooperação da comunidade científica internacional, associada aos avanços no campo da bioinformática e das tecnologias de informação, um primeiro esboço do genoma foi anunciado em 26 de Junho de 2000, dois anos antes do previsto. (http://pt.wikipedia.org/wiki/Projeto_Genoma_Humano)

A.2 Proteínas e Ácidos Nucléicos

As proteínas e os ácidos nucléicos são as moléculas fundamentais de todos os seres vivos. Tanto proteínas como ácidos nucléicos (DNA) são moléculas formadas pela ligação de várias unidades semelhantes chamados *monômeros ou resíduos*, estes são denominados nucleotídeos no caso de DNA e aminoácidos no caso das proteínas. Dessa forma, uma molécula de DNA pode ser descrita completamente se for dada a seqüência de nucleotídeos que a compõem, e no caso das proteínas se for dada a seqüência de aminoácidos que a compõem. A determinação da seqüência exata dos monômeros (ou *resíduos*) que formam uma molécula de ácido nucléico ou proteína chama-se s*eqüenciamento* dessa molécula.

No caso do DNA, os monômeros são os nucleotídeos. Quatro deles ocorrem naturalmente no DNA e são simbolizados pelas letras A, C, G, e T, que identificam as bases que os compõem, como é mostrado na Tabela A.1.

	Letra	***Nome***
1	A	Adenina
2	C	Citosina
3	G	Guanina
4	T	Timina

Tabela A.1: Bases que compõe o DNA

Nas proteínas, os monômeros são os *aminoácidos*. Vinte aminoácidos ocorrem naturalmente em proteínas e encontram-se listados na tabela A.2.

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

Tabela A.2: Os vinte aminoácidos naturais

Nas últimas décadas os processos laboratoriais de seqüenciamento foram sendo aperfeiçoados ao ponto de poderem ser executados a custo e tempo razoáveis, a partir daí houve uma grande produção de seqüenciamentos que causaram o aparecimento de repositórios (bases de dados biológicas) geralmente mantidos por entidades públicas, para o armazenamento de tais informações. Hoje em dia existem várias destas bases de dados disponíveis para consulta de forma eletrônica. Exemplo: *GenBank* (NCBI-NIH, Estados Unidos), *Protein Identification Resource* (PIR, Estados Unidos), *DNA Data Bank Of Japan* (Japão), *European Molecular Biology Laboratory* (EMBL) e no Brasil EMBRAPA .

A.2.1 Estrutura do DNA

A estrutura do DNA é descrita como duas fitas anti-paralelas entrelaçadas em forma de dupla hélice, Figura A.1. Cada hélice é uma cadeia de nucleotídeos que são mantidas juntas através de pontes de hidrogênio.

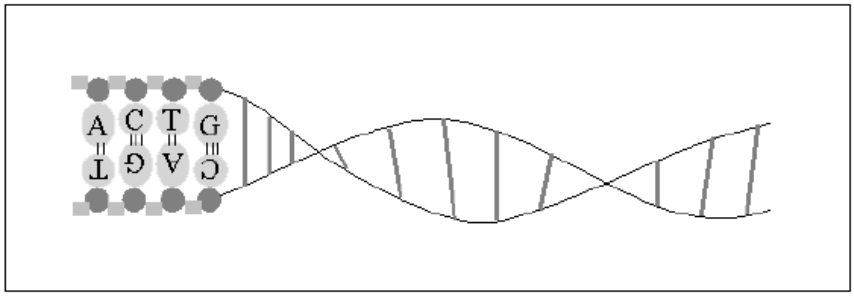


Figura A.1: Estrutura de uma molécula de DNA

A molécula de DNA é direcional, devido à estrutura assimétrica dos açucares que constituem o esqueleto da molécula. Cada açúcar conecta uma fita ("*upstream*"), que a precede na cadeia em seu quinto carbono, com outra fita ("*downstream*"), que a segue na cadeia em seu terceiro carbono. Por isto, as fitas de DNA vão desde 5' (cinco prima) a 3' (três prima). As direções das duas fitas complementares estão invertidas uma na outra. As bases A e T, assim como as C e G são chamadas bases complementares.

A.2.2 Genes e Cromossomas

Cada molécula de DNA é empacotado num *cromossoma*, e sua informação genética total armazenada nos cromossomas de um organismo constitui seu *genoma*. Com poucas exceções, cada célula de um organismo multicelular chamado *Eucarióto* contêm um conjunto completo do genoma, a diferença na funcionalidade das células é devido à variável de expressão dos genes correspondentes. O genoma humano contêm cerca de 3 x 109 pares de bases (abreviado *bp*), organizado em 46 cromossomas – 22 cromossomas *autossomos* pares diferentes, e dois cromossomas *sexuais*: XX ou XY. A quantidade de DNA varia entre diferentes organismos. Por exemplo, o organismo *Amoeba dubia* (um organismo unicelular) tem 200 vezes mais que o DNA humano. Os organismos viventes dividem-se em dois grandes grupos: Procariótos, os quais são organismos formados por uma única célula desprovida de membrana nuclear, e

Eucariótos, os quais são organismos de alto nível, e sua célula tem núcleo. Com o conhecimento contemporâneo das bases bioquímicas da herança, Mendel abstraiu o conceito de que um gene pode ser redefinido como uma entidade física. Um gene é uma região de DNA que controla uma discreta característica hereditária, usualmente correspondente a um único cromossomo carregando a informação para construir uma proteína (chamado mRNA). Em 1977 biologistas moleculares descobriram que a maioria dos genes Eucariótos têm suas seqüências codificadas, chamadas *exons*, interrompidas por seqüências não codificadas chamadas *introns*. (Figura A.2).

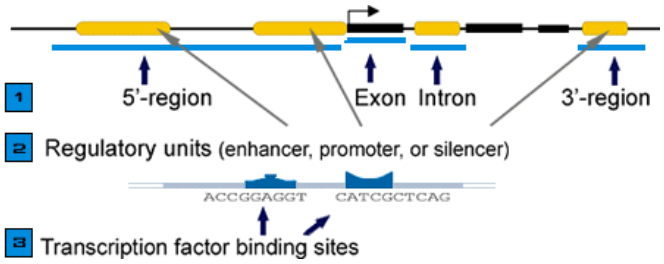


Figura A.2: Exons e Introns

A.2.3 RNA

O RNA é uma molécula formada, geralmente, por uma única cadeia (fita). Sua estrutura é similar ao DNA, exceto pelo fato de que a 2'-desoxirribose é substituída pela ribose e a timina pelo uracil. Existem três classes de RNA: *RNA mensageiro* (mRNA), que contém a informação genética para a seqüência de aminoácidos; o *RNA transportador* (tRNA), que identifica e transporta as moléculas de aminoácido até o ribossomo; e ainda o *RNA ribossômico* (rRNA) que facilita a interação das outras moléculas de RNA na síntese protéica. Todos os tipos de RNA interagem na síntese protéica.

A.3 Dogma Central

O dogma central estabelece o relacionamento entre DNA, mRNA e Proteína.

A expressão da informação gênica armazenada no DNA envolve a *translação* da seqüência de nucleotídeos em uma seqüência co-lineal de aminoácidos e depois em Proteínas.

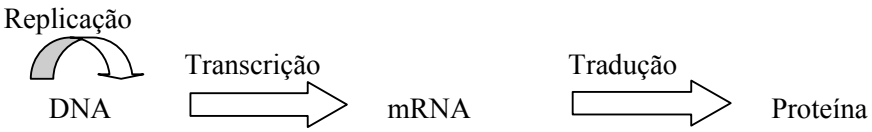


Figura A.3: Síntese Protéica

O mecanismo de síntese de uma Proteína começa na fase de *transcrição*, que consiste na produção de uma molécula de RNA, chamada *RNA mensageiro* (mRNA) que é uma cópia da seqüência do gene, exceto pelo fato de se ter U (*Uracil*) no lugar de T (Timina) no gene. O gene, sendo um trecho de DNA, possui duas fitas. Apenas uma delas é copiada.

O mRNA produzido será então usado na fase de *tradução*. Nesta parte, a seqüência de bases que forma o mRNA é lida para formar a Proteína. As bases são lidas 3 a 3, sendo que cada tripla (chamada *codon*) especifica um aminoácido.

O mecanismo pelo qual uma seqüência de nucleotídeos de um gene é traduzido dentro de uma seqüência de aminoácidos da correspondente Proteína, é chamado *código genético*. (Tabela A.3)

Segunda letra

Primeira letra	U	C	A	G	Terceira letra
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr UAA Stop UAG Stop	UGU Cys UGC Cys UGA Stop UGG Trp	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU His AUC His AUA His AUG Met	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

Tabela A.3 - Código genético

A.4 Estrutura das Proteínas

Uma Proteína é um polímero de aminoácidos entrelaçados por ligações peptidas. A média do tamanho das proteínas é de 200 aminoácidos de comprimento, mas grandes proteínas podem alcançar milhares de aminoácidos de comprimento. Em geral, as células estão feitas de proteínas, as quais constituem mais que a metade de seu peso. Proteínas determinam a forma e estrutura da célula, e também servem como principais instrumentos de reconhecimento molecular e catálise. Elas têm uma complexa estrutura a qual pode ser vista como níveis de estrutura hierárquica. A seqüência de aminoácidos de uma cadeia de proteínas é chamada sua *estrutura primária*. Regiões diferentes da seqüência formam regulares *estruturas secundárias* locais, tal como *α-hélices* as quais são fitas de uma hélice de aminoácidos, e *β-folhas* as quais são folhas planas de segmentos da cadeia que são organizados linearmente. A estrutura terciária é formada empacotando cada estrutura em um ou vários 3D domínios. No final, a proteína contém vários domínios de proteínas organizados em uma estrutura quaternária. A estrutura complexa completa (primária à quaternária) é determinada pela seqüência primária de aminoácidos e sua interação físico-química no meio.

A estrutura de uma proteína determina sua funcionalidade. Embora que a seqüência de aminoácidos determina diretamente a estrutura das proteínas, a identidade de 30% dos aminoácidos na seqüência será, na maioria dos casos, conduzido a uma estrutura de alta similaridade.

A.5 Mutação Biológica.

Uma mutação é definida como uma mudança herdada numa seqüência de aminoácido na proteína, causada por um processo de falha da replicação. Este erro na replicação ocorre com freqüência devido à exposição de radiação ultra violeta ou outra condição do ambiente. Existem dois níveis diferentes no qual a mutação pode tomar lugar. *Mutação gênica* de um gene, onde a mudança ocorre no único gene e mapeia a um *locus cromossomal*, a mutação gênica é também chamada mutação de ponto. Outro nível de mudança hereditária é a *Mutação cromossomal ou por reorganização*, no qual

se produzem intercâmbios de segmentos sob o mesmo cromossoma ou sob outros diferentes (translocações) .

Existem vários tipos de mutações de ponto:

- Substituição: mudança de um ou mais aminoácidos na seqüência de proteínas.
- Inserção: adição de um ou mais aminoácidos na seqüência de proteínas.
- Deleção: deleção de um ou mais aminoácidos na seqüência de proteínas.

Apesar de que uma mutação poderia mudar a seqüência de aminoácidos de uma proteína, não necessariamente afeta a funcionalidade da proteína. Este fenômeno pode ser explicado pelo fato que similaridade química entre diferente aminoácidos poderiam resultar em pequeno ou nenhum impacto sob a estrutura final da proteína, e por isto preservar sua funcionalidade. Além disso, existem regiões na proteína que tem pequena influencia sobre a funcionalidade estrutural da molécula.

Mutações são importantes por varias razões. São responsáveis pelos desordens herdados e outras doenças. Por exemplo, *Sickle-cell anemia*, é uma doença causada pela mutação de substituição da timina pela adenina, a qual resulta no *codon* Valine em vez de Acido Glutâmico, no sexto aminoácido da proteína de hemoglobina. Esta mutação causa um desordem letal no qual o oxigênio é pobremente substituído pela hemoglobina nas células de sangue, causando dano terminal aos tecidos. Em outro aspecto, mutações são a fonte da variação fenotípica na qual a seleção natural atua, criando novas espécies e adaptando outras existentes à mudanças das condições ambientais. A variação dos genes entre organismos nos permite pesquisar a evolução das espécies usando evidência molecular e também obter avanços médicos na busca por novas e melhores drogas.

APENDICE

B

COMPUTAÇÃO EVOLUTIVA E ALGORITMOS GENÉTICOS

Nesta seção descrevemos alguns conceitos importantes utilizados neste documento.

B.1 Computação Evolutiva

B.1.1 Conceitos Básicos

As técnicas evolutivas baseiam seu desenvolvimento e melhoras na imitação direta da natureza através da representação de soluções. O Neo-Darwinismo [FOGEL L., 1966] baseia a evolução da vida em nosso planeta nos processos de reprodução, mutação e competência para a seleção dos indivíduos em gerações futuras. As técnicas evolutivas usam os princípios do Neo-Darwinismo e tentam reproduzir-los estatisticamente através de um algoritmo que simule a evolução e de forma natural chegue a uma boa solução.

A Computação Evolutiva (CE) é um enfoque para abordar problemas complexos de busca e aprendizado através de modelos computacionais de processos evolutivos e à implementação destes processos chama-se Algoritmos Evolutivos (AEs). O propósito dos AEs consiste em guiar uma busca estocástica evoluindo um conjunto de estruturas e selecionando de modo iterativo as mais adequadas.

Os AEs usam um vocabulário derivado da genética natural. Estes se definem a continuação:

População: É o conjunto de indivíduos (genótipos ou estruturas) que representam as soluções do problema, freqüentemente estes indivíduos são chamados também *strings* ou *cromossomos*. Em términos de implementação a população é um vetor de dados no qual se armazenam em forma ordenada as cadeias de informação genética de todos e cada um dos indivíduos (soluções potenciais do problema).

Indivíduos ou cromossomas: estão formadas de unidades –genes (também característica, caráter ou decodificador)- arranjadas em sucessão linear; cada gene

controla a herança de um ou vários caracteres. Genes de certos caracteres estão localizados em certos lugares dos cromossomos, os quais são chamados *loci*(posição do string). Qualquer caráter de indivíduos (como cor do cabelo) pode-se manifestar em forma diferente; o gene é dito que têm vários estados, estes são chamados alelos (valor característico).

Operadores genéticos: São aqueles procedimentos que manipulam a informação genética da população para produzir novos indivíduos que são vistos como variações da população original. Os operadores genéticos mais usados em algoritmos evolutivos (AEs) são:

Mutação: é a alteração sobre um o mais alelos da genética do indivíduo. Ao aplicar este operador em forma aleatória o dirige a uma região diferente do espaço de busca do indivíduo original; isto faz que a busca não fique num ótimo local; conseqüentemente se diz que a mutação é o procedimento do algoritmo que dedica-se à exploração do espaço de busca.

Cruzamento: é o procedimento pelo qual se "misturam" as caraterísticas de dois cromossomos (chamados de pais) para formar as cadeias cromossômicas dos descendentes (chamados de filhos), preservando os cromossomos presumivelmente "bons" que nos levaram a obter melhores soluções ao problema; a recombinação é o procedimento do algoritmo que *explora* as zonas mais prometedoras no espaço de busca.

Seleção: é o processo que escolhe os indivíduos que passaram a formar parte da seguinte geração. O processo de seleção toma em conta a função de aptidão, então o algoritmo simula o principio de sobrevivência do mais apto.

Função de aptidão (*fitness*): Cada um dos indivíduos possui um valor que determina sua adequação de sobrevivência ante seu entorno. Dito valor de adequação determina quanto é bom o indivíduo como solução ao problema que se trata resolver. A determinação de este valore chama-se avaliação da *função de aptidão*.

Deve-se ter em conta que a função de aptidão deve estar relacionada com a função objetivo a otimizar, o seja que por enquanto melhor seja o indivíduo para a função objetivo, maior será sua aptidão de sobrevivência ante o entorno. A diferencia entre estes conceitos, função objetivo e função de aptidão, é mais bem conceptual a fim

de diferencias que as funções objetivos são uma característica intrínseca do problema a otimizar, sem embargo a função de aptidão é própria do algoritmo evolutivo.

Para o caso de otimização mono-objetivo estas duas funções coincidem e em outras ocasiões a função de aptidão é o resultado de escalar proporcionalmente a função objetivo. Para o caso de otimização multi-objetivo a função de aptidão representa precisamente o compromisso procurado entre todas as funções objetivo do problema.

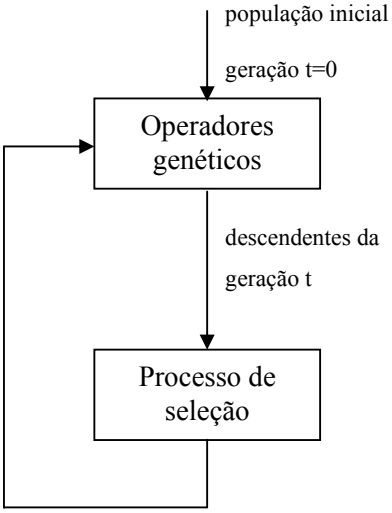


Figura B.1: Base do ciclo evolutivo

B.2 Algoritmos Genéticos

Os algoritmos genéticos (AGs) foram introduzidos por Holland a princípios dos anos 60 e são a técnica evolutiva mais popular da atualidade. O objetivo de Holland foi o estudo formal dos processos de adaptação natural e o traslado destes mecanismos à computação.

Definição [KOZA, J. R., 1992] : É um algoritmo matemático altamente paralelo que transforma um conjunto de objetos matemáticos individuais com respeito ao tempo usando operações modeladas de acordo ao principio Darwiniano de reprodução e sobrevivência do mais apto, e trás haver-se apresentado de forma natural uma série de operações genéticas de entre as que destaca a recombinação sexual. Cada um destes objetos matemáticos freqüentemente é uma cadeia de caráteres (letras ou números) de comprimento fixo que se ajusta ao modelo de cadeias de cromossomos, e associa-se a eles uma certa função matemática que reflexa sua aptidão.

Um algoritmo genético (AG), deve ter os seguintes seis componentes:

1. Uma representação genética para soluções potenciais ao problema,
2. Uma forma de criar uma população inicial de soluções potenciais,
3. Uma função de aptidão que joga o rol do ambiente, avalia as soluções em termos de seu "fitness",
4. Um mecanismo de seleção que permita selecionar ao indivíduos de acordo a sua aptidão. Nos algoritmos genéticos pode-se levar o processo de seleção de diversas formas, seja determinística ou probabilisticamente (dando oportunidade aos menos aptos para reprodução). As técnicas de seleção podem-se agrupar em três grupos:
   - *Seleção proporcional*: Em esta seleção os indivíduos escolhem-se de acordo a sua aptidão com respeito à população. Subdivide-se em:
     - Roleta
     - Resto estocástico
     - Universal estocástico.
     - Mostra determinístico.
   - *Seleção por torneio*: Escolhe-se os indivíduos em base a comparações diretas entre indivíduos. Divide-se em:
     - Torneio determinístico.
     - Torneio probabilístico.
   - *Seleção uniforme*: substitui aos indivíduos menos aptos da geração pelos mais aptos da nova geração.
5. Operadores genéticos que alteram a composição dos descendentes (filhos) que se produziram para as seguintes gerações, podem ser cruzamento e mutação.

   O AG enfatiza a importância da cruza sexual (operador principal) sobre o da mutação (operador secundário) e necessita elitismo para poder convergir ao ótimo.

   Existem três tipos principais de cruzamento:

- *Cruza de um ponto*: seleciona-se um ponto de forma aleatória dentro do cromossomo de cada pai e a partir de este intercambiam-se os materiais genéticos para dar origem a novos indivíduos .
- *Cruza de dois pontos*: igual ao anterior exceto que se geram dois pontos de cruza por cada pai .
- *Cruza uniforme*: cruza de n pontos .

6. Valores para vários parâmetros que o algoritmo genético usa (tamanho da população, probabilidades de aplicar operadores genéticos, etc.)

O algoritmo geral de um algoritmo genético simples se dá a continuação:

Algoritmo B.1: Algoritmo genético simples.

```
Inicio
        Gerar aleatoriamente a população inicial, G(0)
        t=0
        Repetir
                t=t+1
                Calcular a aptidão de cada indivíduo de G(t)
                Selecionar G(t) em base à aptidão de G(t)
                Aplicar os operadores genéticos a G(t) para gerar G(t+1)
        Até (condição de parada)
Fim
```

APENDICE

C

PARALELIZAÇÃO

Apresentamos a seguir uma breve introdução aos conceitos de processamento paralelo utilizados neste trabalho.

C.1 Arquitetura de máquinas paralelas

Existem hoje diversas arquiteturas de máquinas paralelas e a classificação mais utilizada foi proposta por Flynn [FLYNN M. J., 1972], conhecida como *taxonomia de Flynn*. Segundo Flynn, existem quatro categorias básicas de máquinas:

1. *Single Instruction, Single Data stream* (SISD) – é o modelo de Von Neumann, nesta categoria encaixam-se os microcomputadores pessoais e máquinas com apenas um processador.
2. *Single Instruction, Multiple Data stream* (SIMD) – inclui máquinas que suportam paralelismo em *arrays*, que consiste em máquinas com vários processadores conectados em forma de *grid*. Estas máquinas possuem uma unidade de controle que recebe e interpreta instruções, para em seguida enviá-las aos processadores conectados em grid para processamento paralelo.
3. *Multiple Instruction, Single Data stream* (MISD) – poucas máquinas foram construídas nesta categoria e nenhuma teve sucesso comercial ou algum impacto na computação científica.
4. *Multiple Instruction, Multiple Data stream* (MIMD) – é a categoria mais abrangente, cobrindo máquinas com vários processadores e que suportam paralelismo de processos.

Em uma máquina com múltiplos processadores, cada processador é identificado por um número, iniciando em 0 e terminando em $p-1$, em que p é o número de

processadores existentes na máquina. As máquinas que possuem múltiplos processadores pertencem, segundo a taxonomia de Flynn, à categoria MIMD.

As máquinas de múltiplos processadores, de acordo com a forma de acesso à memória, podem ser classificadas de três formas

- Sistemas de memória compartilhada – a memória é global e acessível a todos os processadores.
- Sistemas de memória distribuída ou de memória local – a memória é local, cada processador possui uma memória separada e não é acessível aos demais processadores.
- Sistemas híbridos – possuem memória distribuída e compartilhada. Nestes sistemas, os processadores são divididos em grupos e, cada grupo de processadores possui seu espaço de memória, que não é acessível aos demais grupos.

Como exemplos de sistemas de memória compartilhada podemos citar o SGI Altix 350, o Cray SV1. Como exemplos de máquinas de memória distribuída, encontram-se o IBM SP2.

Software

O paralelismo em máquinas paralelas é comumente gerenciado através de APIs (*Application Programming Interfaces*) específicas. Dependendo da arquitetura da máquina, o paralelismo pode ser obtido através da criação de processos ou threads paralelos, definidos como segue:

- Um processo consiste de um conjunto de instruções que executam uma tarefa, e não compartilham recursos alocados pelo sistema operacional, como o espaço de memória. Um processador é o dispositivo físico de hardware que executa os processos.
- *Threads* são partes de um processo, cada processo possui pelo menos um *thread*, que é chamado de thread mestre e diversos *threads* podem existir em um processo. Os *threads* podem compartilhar entre eles os recursos alocados pelo sistema operacional para o processo, como o espaço

memória, o que os torna adequados à programação paralela em sistemas de memória compartilhada.

Em sistemas de memória distribuída, atualmente as APIs mais utilizadas são as de passagem de mensagem, como MPI (*Message Passing Interface*) e PVM (*Parallel Virtual Machine*). Estas APIs criam, em cada processador, processos que executam as tarefas de forma independente. A API de passagem de mensagens é responsável por gerenciar a comunicação entre os processos criados e também por finalizá-los.

Em sistemas de memória compartilhada, o paralelismo pode ser gerenciado através do OpenMP (*Open Multi Processing*), que consiste de um conjunto de especificações e interfaces para paralelizar um programa. O OpenMP funciona criando um único processo com vários threads paralelos, o número de threads é definido pelo usuário e é recomendável que seja igual ao número de processadores disponíveis, pois assim cada thread é executado por um processador, evitando a execução de mais de um thread por algum processador. O OpenMP é destinado exclusivamente a sistemas de memória compartilhada e a comunicação entre os threads é feita através da leitura direta da área de memória em que está armazenada a informação desejada. As APIs de passagem de mensagens também podem ser utilizadas em máquinas de memória compartilhada, o que torna os programas que as utilizam mais portáveis.

Medidas de desempenho

O objetivo do paralelismo é reduzir o tempo necessário para completar uma determinada tarefa computacional. O desempenho de um algoritmo paralelo é medido pela aceleração e pela eficiência, definidos a seguir.

Definição C.1. Seja t_p o tempo de execução de um programa paralelo em p processadores e t_s o tempo de execução do algoritmo serial mais rápido utilizando 1 processador, a aceleração em p processadores é definida pela relação

$$S_p = \frac{t_s}{t_p}$$

A aceleração, mede o ganho obtido em utilizar um programa paralelo ao invés de um programa serial executado em apenas um processador. Outra definição de aceleração é dada em função da versão seqüencial do algoritmo paralelo.

Definição C.2: Seja t_p o tempo de execução de um algoritmo paralelo em p processadores e t_1 o tempo de execução do mesmo algoritmo em 1 processador, a aceleração em p processadores é definida por

$$S_p = \frac{t_1}{t_p}$$

De acordo com a Definição C.2, a aceleração é uma medida do ganho obtido pela paralelização do algoritmo, e mede diretamente os efeitos provocados pela comunicação entre processos e sincronização.

Idealmente, a aceleração deve crescer linearmente, com $S_p = p$, no entanto isto raramente é verificado na prática, pois a medida que o número de processadores p é aumentado, aumenta também o tempo gasto com sincronização e comunicação entre os processos. Outra medida de desempenho associada à aceleração é a eficiência, definida a seguir.

Definição C.3: (Eficiência). A eficiência em p processadores é determinada pela relação

$$E_p = 100 \times \frac{S_p}{p}$$

A eficiência mede o percentual, a fração de tempo em que os recursos totais de computação são utilizados, sem contar o tempo total gasto com comunicação e sincronização. Idealmente, quando a aceleração é linear, a eficiência é igual a 100% para todo p, o que raramente é verificado na prática. Em implementações reais, o que se verifica é um decréscimo da eficiência a medida que o número de processadores utilizados é incrementado, provocado pelo aumento tempo gasto em operações de sincronização e comunicação entre os processos. No desenvolvimento de algoritmos paralelos, deseja-se que a aceleração seja mantida o mais próximo possível a p e a eficiência o mais próximo possível a 100%.

As medidas de desempenho apresentadas acima são dadas em função do número de processadores utilizados no processamento e são os parâmetros que serão utilizados para avaliar o desempenho dos programas paralelos desenvolvidos neste trabalho.

C.2 Paralelização dos algoritmos evolutivos

Aplicam-se técnicas de processamento paralelo e distribuído aos modelos clássicos de EAs com o propósito de obter melhoras na eficiência computacional e proporcionar um mecanismo diferente de exploração do espaço de busca [CANTÚ & PAZ, 2001]. Dividindo a população em vários elementos de processamento, os Algoritmos Evolutivos Paralelos (pEAs) permitem afrontar a lentidão da convergência para problemas complexos que motivam o uso de populações numerosas ou múltiplas avaliações de funções objetivo que exigem o custo computacional elevado. Conjuntamente, os pEAs introduzem um modelo de evolução diferente ao modelo panmíctico seqüencial, que possibilita explorar o trabalho simultâneo sobre populações geograficamente distribuídas o localizadas de acordo a uma estrutura de organização espacial subjacente.

A organização da população constitui o principal critério utilizado pelos investigadores para classificar os modelos de pEAs [ALBA & TOMASSINI, 2002]. Deste modo, destacam-se três grandes famílias de peãs:

onde cada indivíduo tem a possibilidade de interagir –competir ou cruzarse– com qualquer outro

- pEAs baseados no modelo mestre-escravo, onde se distribui a avaliação da função de fitness e se mantém o mecanismo de evolução com interação panmíctica (onde cada indivíduo tem a possibilidade de interagir –competir ou cruzar– com qualquer outro), que é característico dos modelos seqüenciais. A figura 8 apresenta graficamente o modelo mestre-escravo:

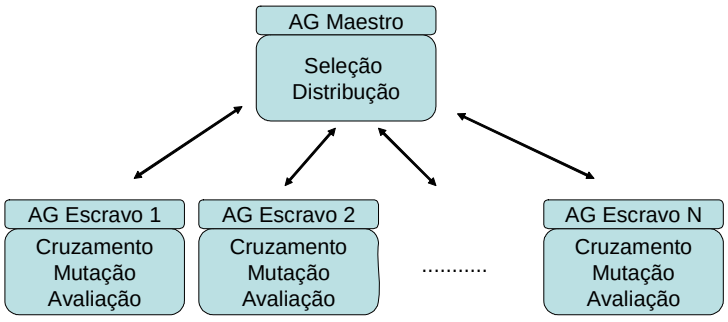


Figura C.1: Algoritmo Evolutivo Paralelo Modelo Mestre-Escravo

- pEAs da população distribuída, trabalha com um conjunto de sub-populações independentes (ilhas) com a limitação de que as interações somente são possíveis entre dois indivíduos da mesma ilha. Um operador adicional chamado migração possibilita intercâmbios ocasionais de indivíduos entre ilhas, introduzindo uma nova fonte de diversidade. A figura 9 apresenta graficamente o modelo de população distribuída.

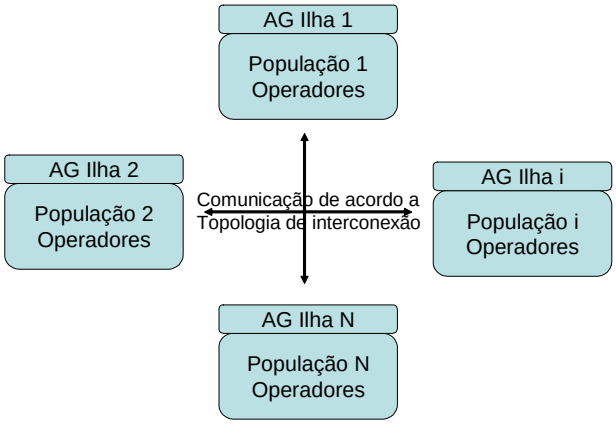


Figura C.2: Algoritmo Evolutivo Paralelo Modelo de População Distribuída.

- pEAs celulares, caracterizados por possuir uma estrutura espacial subjacente à população e por seu modelo especial de propagação de características de indivíduos, denominado difusão, que segue os endereços definidos pela topologia de interconexão de elementos de processamento. A figura 10 apresenta graficamente o modelo celular:

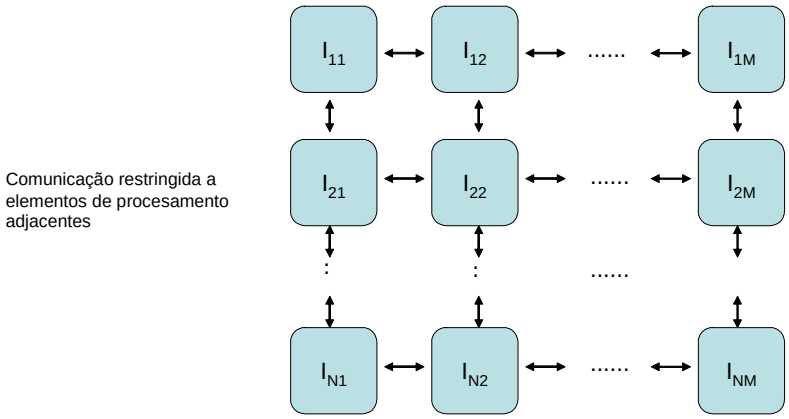


Figura C.3: Algoritmo Evolutivo Paralelo Modelo Celular

Estes modelos têm diferentes variantes, e existem modelos híbridos que combinam as características de dois o mais dos modelos da categorização geral apresentada.

Na última década, os modelos paralelos de EAs foram popularizados por sua eficiência computacional e aplicabilidade para a resolução de variados problemas em diversas áreas como indústria, economia, telecomunicações, bioinformática e outras [ALBA & TOMASSINI, 2002].

C.3 Referências de Algoritmos Evolutivos Paralelos para Otimização Multi-objetivo

De acordo ao texto de referência [COELLO et al., 2002] o número de aplicações de paralelismo no campo de MOEAs é muito baixo, e o alcance das propostas é bastante restringido. A referência imediata sobre paralelismo aplicado a MOEAs a constitui o artículo de Veldhuizen et al. (2003) onde os autores expõem as principais considerações do desenho, implementação e teste de algoritmos evolutivos paralelos para otimização Multi-objetivo.

A continuação comenta-se três referências a propostas de aplicação de paralelismo a variantes do algoritmo NSGA. Dois de elas desenham versões paralelas dentro do modelo mestre-escravo na área de fluidodinâmica, concretamente em problemas de desenho de perfiles aerodinâmicos. A terceira proposta corresponde ao próprio criador do algoritmo quem propõe um modelo de populações distribuídas.

A primeira referência corresponde a Mäkinen et al. [MÄKINEN et al., 1997] eles propõem um modelo paralelo mestre-escravo a fim de avaliar custosas funções de fitness em um problema de desenho de perfiles aerodinâmicos em dois dimensões. Os autores modificaram o algoritmo NSGA original, substituindo a seleção mediante roleta pela seleção mediante torneio e modificando o mecanismo tradicional de *sharing* por uma variante denominada *tournament slot sharing*, que fixa ao parâmetro σ um valor relativo a um slot do torneio. Adicionalmente, introduziram um mecanismo de evolução elitista, perpetuando os indivíduos não dominados em cada geração. O algoritmo foi implementado utilizando a biblioteca MPICH de passo de mensagens. As rodadas reportadas sobre um IBM SP2 de 9 processadores modelo 390 apresentam

bons valores de eficiência, a pesar do custo total de execução de uma otimização resulta, ainda para o modelo paralelo, excessivamente alto.

Outra proposta foi aplicar uma estratégia de paralelismo em dois níveis para abordar um problema similar, de desenho de perfiles aerodinâmicos. A estratégia de paralelismo aplicada ao algoritmo NSGA corresponde à distribuição da avaliação de funções de fitness (cálculo de fluxos Eulerianos). Os autores reportaram utilizar um equipo SGI Origin 2000 com processadores R10000 a 195 Mhz, e trabalhando com 8 processadores obter um tempo razoável para a resolução do problema de desenho abordado, mas não se realiza comparações de eficiência contra modelos seriais.

Um último enfoque constituiu a proposta de dominância guiada por distribuição [DEB et al., 2002], que se baseia uma busca concorrente mediante divisão de domínio. Os autores propõem métodos para guiar a busca a diferentes seções do frente de Pareto introduzindo uma transformação dos objetivos através de uma soma ponderada, que modifica o conceito de dominância possibilitando que diferentes processos se enfoquem em diferentes regiões de busca. Para possibilitar a cooperação entre processos, se baseia na necessidade de introduzir um operador de migração. Reportam-se bons resultados no referente à cobertura do frente Pareto, e muito bons resultados respeito à eficiência, logrando speedup lineal no conjunto de problemas de teste estudados.

Referências Bibliográficas

AART, E.H.L. AND VAN LAARHOVEN, P.J.M., 1987, *Simulated Annealing:a Review of Theory and Applications*, Amsterdam, Kluwer Academic Publishers.

ALBA E., 2002, "Parallel Evolutionary Algorithms can Achieve Super-Linear Performance", Information Processing Letters, Elsevier, 82(1):7-13, abril 2002.

ALBA E., TOMASSINI M., 2002. "Parallelism and Genetic Algorithms", *IEEE Transactions on Evolutionary Computation*, v. 6, n. 5, pp. 443-462.

ALTSCHUL, S.F., ERICKSON, B.W., 1986, "Optimal Sequence Alignment Using Affine Gap Costs", *Bull. Math. Biol.*, v. 48, pp. 603-609.

ALTSCHUL, S.F., 1989, "Gap Costs for Multiple Sequence Alignment", *J. Theor. Biol.*, v.138, pp. 297-309.

ALTSCHUL, S.F., 1991, "Amino acid substitution matrices from an information theoretic perspective", *J. Mol. Biol.*, v.219, pp. 555-565.

ANG, K. H., LI, Y., 2001, "Multi-Objective Benchmark Studies for Evolutionary Computation". In: *2001 Genetic and Evolutionary Computation Conference. Workshop Program*, pp. 393–396, San Francisco, California, Julio 2001.

AZARM, S., REYNOLDS, B. J., NARAYANAN, S, 1999, "Comparison of Two Multiobjective Optimization Techniques With and Within Genetic Algorithms", In: *CD-ROM Proceedings of the 25th ASMEDesign Automation Conference*, v. Paper No. DETC99/DAC-8584, Las Vegas, Nevada, Setembro 1999.

BALDI, P., CHAUVIN, Y., HUNKAPILLAR, T. & MCCLURE, M., 1994, "Hidden Markov models of biological primary sequence information", *Proc. Natl. Acad. Sci. USA*, v. 91, pp. 1059-1063.

BARAN, B., DUARTE, S., 2001, "Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms", *XXVII Conferencia Latinoamericana de Estudios en Informática CLEI*, Mérida, Venezuela.

BARTONT, G. J., STERNBERG, M. J. E., 1987, "A Strategy for the Rapid Multiple Alignment of Protein Sequence", *J. Mol. Biol.*, v.198, pp. 327-337.

BREMERMANN, H., 1962, "Optimization through evolution and recombination", *Self-organizing Systems,* En *M. C. Yovits, G. T. Jacobi, y G. D. Goldstine*, editors, Washington, Spartan Books pp. 93-106.

CANCINO, W.G, 2003, *Aplicação de Algoritmos Genéticos Multi-Objetivo para Alinhamento de Seqüências Biológicas*, MSc. Dissertação, Instituto de Ciências Matemática e de Computação – ICMC-USP, São Carlos, São Paulo, Brasil.

http://www.lania.mx/~ccoello/EMOO/

CANTÚ-PAZ E., 2001, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publisher.

CARRILLO, H., LIPMAN, D., 1988, "The multiple sequence alignment problem in biology", *SIAM J. Appl. Math.*, v. 48, pp. 1073-1082.

CHELLAPILLA, K., FOGEL, GB, 1999, "Multiple Sequence Alignment Using Evolutionary Programming", *Proc. 1999 Congress on Evolutionary Computation IEEE*, Washington DC, pp. 463-469.

CHEN, L., WU, L., WANG, R., WANG, Y., ZHANG, S., ZHANG, X., 2005, "Comparison of protein structures by multi-objective optimization", *Genome Informatics*, v. 16, n. 2, pp. 114-124.

COELLO COELLO C. A. www.lania.mx/ ccoello/EMOO/EMOObib.html.

COELLO, C. A., 1999, "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques", *Knowledge and Information Systems*, v. 1, n. 3, pp. 269- 308.

COELLO, C. A., 2001, "A Short Tutorial on Evolutionary Multiobjective Optimization". En: E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, y D. Corne, editors, *First International Conference on Evolutionary Multi-Critérion Optimization*, Lecture Notes in Computer Science, n. 1993, pp. 21–40, Springer-Verlag.

DAYHOFF, M.O., SCHWARTZ, R.M., ORCUTT, B.C, 1978, "A model of evolutionary change in proteins", *Atlas of Protein Sequence and Structure*, NBRF Washington, v. 5, supplement 3.

http://www.cs.ucsb.edu/~ambuj/Courses/bioinformatics/dayhoffetal1978.pdf

DEB, K., AGRAWAL, S., PRATAP, A., MEYARIVAN, T., 2000, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II", *PPSN*, pp. 849-858.

DEB, K., 2001, *Multi-objective optimization using evolutionary algorithms*, New York, John Wiley & Sons.

DELLA VEDOVA, G, BONIZZONI, P, 2001, "The complexity of multiple sequence alignment with SP-score that is a metric", *Theoretical Computer Science*, v. 259, n. 1-2, pp.63-79.

DORIGO, M., 1992, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italia.

http://iridia.ulb.ac.be/~mdorigo/ACO/publications.html

EDDY, S.R., 1995, "Multiple alignment using hidden Markov models", *ISMB*, v.3, pp. 114–120.

FITCH, V.M., MARGOLIASH, E., 1966, "Construction of Phylogenetic Trees", *J.Mol.Biol.*, v. 16, pp. 9-17.

FLYNN, M. J., 1972, "Some computer organizations and their effectiveness". *IEEE Transactions on Computers*, v. C-21, n. 9, pp. 948–960.

FOGEL, L. F., 1964, *On the Organization of intellect*, PhD thesis, University of California, Los Angeles, California.

FONSECA C.M., FLEMING P.J., 1995, "An Overview of Evolutionary Algorithms in MultiobjectiveOptimization", *Evolutionary Computation*, v.3, n.1, pp. 1-16.

GLOVER, F., 1989, "Tabu Search — Part I", *ORSA Journal on Computing*, v.1, n.3, pp.190-206

GLOVER, F., 1990, "Tabu Search — Part II", *ORSA Journal on Computing*, v.2, n.1, pp. 4-32.

GOLDBERG, D. E., & RICHARDSON, J., 1987, "Genetic algorithms with sharing for multimodal function optimization". In: Grefenstette, J. J. (Ed.), *Genetic algorithms and their applications : Proc. of the second Int. Conf. on Genetic Algorithms*, pp. 41–49, Hillsdale, NJ. Lawrence Erlbaum Associates.

GOLDBERG D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachusetts.

GOTOH O., 1996, "Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments", *J. Mol. Biol.*, v. 264, pp. 823–838.

GUPTA, S.K., KECECIOGLU, J. and SCHAFFER, A.A., 1996, "Improving the practical space and time efficiency of the shortest-paths Approach to sum-of-pairs multiple sequence alignment.", *J. Comput. Biol.*, v.2, pp. 459–472.

HAJELA, P. and LIN, C.-Y., 1992, "Genetic search strategies in multicritérion optimal design", *Structural Optimization*, v.4, n.2, pp. 99-107.

HANADA, Y., YOKOYAMA, T., SHIMIZU, T., 2000, "Multiple Sequence Alignment by Genetic Algorithm", *Genome Informatics*, v.11, pp. 317-318.

HENIKOFF, S., HENIKOFF, J.G., 1992, "Amino acid substitution matrices from protein blocks", *Proc. Natl. Acad. Sci. USA*, v. 89, pp. 10915-10919.

http://amber.cs.umd.edu/class/838-s04/papers/henikoffandhenikoff1992.pdf

HOLLAND, J. H., 1962, "Outline for a Logical Theory of Adaptive System", *Journal of the ACM*, v. 9, n. 3, pp.297-314.

HOLLAND, J. H., 1975, *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor.

HORNG, J., WU, L., LIN, C., YANG B., 2004, "A genetic algorithm for multiple sequence alignment", *Soft Comput*, v. 9, pp. 407–420.

ISHIKAWA, M., TOYA, T., HOSHIDA, M., NITTA, K., OGIWARA, A., KANEHISA, M., 1993, "Multiple sequence alignment by parallel simulated annealing", *Comput. Appl. Biosci.*, v.9, pp. 267-273.

ISOKAWA, M., TAKAHASHI, K., AND SHIMIZU, T., 1996, "Multiple sequence alignment using a genetic algorithm", *Genome Informatics*, v.7, pp.176–177.

KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P., 1983, Optimization by Simulated Annealing, *Science*, v. 220, n. 4598, pp. 671-680.

KOZA, R.J., 1992, *Genetic Programming: On the programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts.

KROGH, A., BROWN, M., MIAN, S., SJOLANDER, K. AND HAUSSLER, D., 1994, "Hidden Markov Models in Computational Biology: Applications to Protein Modeling", *J. Mol. Biol.*, v. 235, pp.1501-1531.

KROGH, A., MIAN, I. S., HAUSSLER, D., 1994, "A hidden Markov model that finds genes in E. coli DNA", *Nucleic Acids Res.*, v. 22, n. 22, pp. 4768–4778.

LAWRENCE J. FOGEL, 1966, *Artificial inteligence through simulated evolution*, John Wiley, New York.

LIPMAN,D.J., ALTSCHUL,S.F. and KECECIOGLU,J.D., 1989, "A tool for multiple sequence alignment", *Proc. Natl. Acad.Sci. USA*, v. 86, pp. 4412–4415.

MÄKINEN, R.A.E., NEITTAANMÄKI, P., PERIAUX, J., SEFRIOUI, M., TOIVANEN, J., 1997, "Parallel Genetic Solution for Multiobjective MDO", In A. Schiano, A. Ecer, J. Périaux, and N. Satofuka, editors, *Parallel CFD'96 Conference*, pages 352-359, Capri, 1996. Elsevier.

MIETTINEN, K.M., 1999, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, Massachusetts.

McCLURE, M.A., VASI, T.K., FITCH, W.M., 1994, "Comparative analysis of multiple protein-sequence alignment methods", *Mol. Biol. Evol.*, v.11, n.4, pp. 571-592.

NEEDLEMAN, S. B., BLAIR, T. H., 1969, "Homology of Pseudomonas Cytochrome c-551 with Eukaryotic c-cytochromes", *Proc. Nat. Acad. Sci.*, v.63, n.4, pp. 1227-1233.

NEEDLEMAN, S., WUNSCH, C., 1970, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *Journal of Molecular Biology*, v. 48, pp. 443–453.

NOTREDAME, C., HIGGINS, D. G., 1996, "SAGA: sequence alignment by genetic algorithm", *Nucleic Acids Res.*, v. 24, pp. 1515–1524.

NOTREDAME, C., HOLM, L. & HIGGINS, D. G., 1998, "COFFEE: an objective function for multiple sequence alignments", *Bioinformatics*, v. 14, pp. 407-422.

NOTREDAME, C., 2002, "Recent progrésé in multiple sequence alignment: a survey", *Pharmacogenomics*, v.3.

OSYCZKA, A., 1985, "Multicriteria optimization foe engineering design". In: Gero, J. S. (Ed.), *Design Optimization*, pp. 193-227. Academy Press.

RECHENBERG, I., 1973, *Evolutionsstrategie: Optimierung technischer systemenach prinzipien de biologishen evolution*, Technical report, Stuttgart: Frommann-Holzboog.

SAID, Y.H., 2005, "On Genetic Algorithms and their Applications", *Handbock of Estatistics*, V. 24, pp. 359-390

SANKOFF, D., 1975, "Minimal mutation trees of sequences", *SIAM Journal on Applied Mathematics*, v. 28, pp. 35–42.

SANKOFF,D. AND CEDERGREN,R.J., 1983, Simultaneous comparison of three or more sequences related by a tree. In: Sankoff, D. and Kruskal, J.B. (eds) *Time Warps, String Edits, and Macromolecules:The Theory and Practice of Sequence Comparison*, pp. 253–263, Addison-Wesley, Reading, MA.

SCHAFFER, J.D., 1985, "Multiple objective optimization with vector evaluated genetic algorithm", In: J.J. Grefensttete, Ed. Hillsdale, *Proceedings of the First International Conference on Genetic Algorithm*, pp. 93-100, NJ: Lawrence Erlbaum.

SCHWEFEL, H.P., 1995, *Evolution and Optimum Seeking*, Wiley Inter-science.

SEELUANGSAWAT, P., CHONGSTITVATANA, P., 2005, "A Multiple Objective Evolutionary Algorithm for Multiple Sequence Alignment", In: *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 477-478, Washington DC, USA.

SETUBAL, J., MEIDANIS, J., 1997, *Introduction to Computational Molecular Biology*, Brooks, United States.

SMITH, T., WATERMAN, M., 1981, "Identification of common molecular subsequences", *Journal of Molecular Biology*, v.147, pp. 195–197.

SNEATH, SNOKAL, 1973, "UPGMA Unweighted Pair Group Method with Arithmetic Mean", *Numerical Taxonomy*, W.H. Freeman and Company, San Francisco, pp. 230-234.

TAYLOR, WR, 1988, "A flexible method to align large numbers of biological sequences", *J. Mol. Evol.*, v. 28, pp. 161–169.

THOMPSON, J. D., HIGGINS, D. G., GIBSON, T. J., 1994, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence

weighting, position-specific gap penalties and weight matrix choice", *Nucleic Acids Res.*, v.22, n.22, pp. 4673–4680.

THOMPSON, J. D., GIBSON, T. J., PLEWNIAK, F., JEANMOUGIN, F., HIGGINS, D. G., 1997, *Nucleic Acids Res.*, v.25, n.24, pp. 4876–4882.

THOMPSON, J.D, PLEWNIAK, F., POCH, O., 1999, "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs", *Bioinformatics Application Note*, v.15, n.1, pp.1999.

VAN VELDHUIZEN D., LAMONT G., 1999, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD. Dissertation, Air Force Institute of Technology Air University, Montgomery, Alabama, USA.

www.lania.mx/~**ccoello**/EMOO/

VAN VELDHUIZEN, D. A., LAMONT, G. B., 2000, On Measuring Multiobjective Evolutionary Algorithm Performance. In: *2000 Congress on Evolutionary Computation*, v. 1, pp. 204–211, Piscataway, New Jersey, Julio 2000. IEEE Service Center.

VAN VELDHUIZEN D., ZYDALLIS J., LAMONT G., 2003, "Considerations in engineering parallel multiobjective evolutionary algorithms", *IEEE Trans. Evolutionary Computation*, v.7, n.2, pp. 144-173.

WANG, L, JIANG, T, 1994, "On the Complexity of Multiple Sequence Alignment", *Journal of Computational Biology*, v.1, n.4, pp. 337-348.

ZADEH, L.A., 1963, "Optimality and Nonscalar-Valued Performance Criteria", *IEEE Transactions on Automatic Control*, v.AC-8, n.1, pp. 59-60.

ZHANG, C., AND WONG, A. K. C., 1997, "A Genetic Algorithm for Molecular Sequence Comparison", *Comput. Applic. Biosci.*, v. 13, n. 6, pp. 565-581.

ZHANG, C., WONG, A. K. C., 1997, "Toward Efficient Multiple Molecular Sequence Alignment: A System of Genetic Algorithm and Dynamic Programming", *IEEE Transactions on systems, man, and cybernetics—part b: Cybernetics*, v. 27, n. 6.

ZITZLER E., DEB, K., and THIELE, L., 1999, "Comparison of multiobjective evolutionary algorithms: empirical results", *Evolutionary Computation*, v.8, n.2, pp. 173-195.

ZITZLER E., THIELE L., 1998, “An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach”, Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Mayo 1998.

ZITZLER, E., THIELE, L., 1999, “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach”, *IEEE Transactions on Evolutionary Computation*, v.3, n.4, pp.257–271.

ZWIR, I, ROMERO-ZÁLIZ, R, 2002, “Automated Biological Sequence Décription and Recognition by a Localized Multiobjective Genetic Algorithm”, *JCIS* , pp. 586-589.