SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura : _____

# Aplicação de Algoritmos Genéticos Multi-Objetivo para Alinhamento de Seqüências Biológicas

***Waldo Gonzalo Cancino Ticona***

**Orientador:** ***Prof. Dr. Zhao Liang***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências, na área de Ciências de Computação e Matemática Computacional.

USP-São Carlos

**Fevereiro de 2003**

# Aplicação de Algoritmos Genéticos Multi-Objetivo para Alinhamento de Seqüências Biológicas

*Waldo Gonzalo Cancino Ticona*

# Agradecimentos

A meu orientador Prof. Dr. Zhao Liang, por tudo o que com ele aprendi, em ambiente propício para a pesquisa de excelência.

Ao professor André Ponce de Leon Ferreira de Carvalho por sua amizade e por ter acompanhado com muito interesse minha pesquisa.

A todos os meus amigos do mestrado do ICMC-USP pela acolhida, abertura, e disponibilidade em me ajudar

Ao CNPQ, pelo fundamental suporte financeiro dispensado à execução desta pesquisa.

*À minha família e aos meus amigos.*

# Resumo

O alinhamento de seqüências biológicas é uma operação básica em Bioinformática, já que serve como base para outros processos como, por exemplo, a determinação da estrutura tridimensional das proteínas. Dada a grande quantidade de dados presentes nas seqüencias, são usadas técnicas matemáticas e de computação para realizar esta tarefa. Tradicionalmente, o Problema de Alinhamento de Seqüências Biológicas é formulado como um problema de otimização de objetivo simples, onde alinhamento de maior semelhança, conforme um esquema de pontuação, é procurado.

A Otimização Multi-Objetivo aborda os problemas de otimização que possuem vários critérios a serem atingidos. Para este tipo de problema, existe um conjunto de soluções que representam um "compromiso"entre os objetivos. Uma técnica que se aplica com sucesso neste contexto são os Algoritmos Evolutivos, inspirados na Teoria da Evolução de Darwin, que trabalham com uma população de soluções que vão evoluindo até atingirem um critério de convergência ou de parada.

Este trabalho formula o Problema de Alinhamento de Seqüências Biológicas como um Problema de Otimização Multi-Objetivo, para encontrar um conjunto de soluções que representem um compromisso entre a extensão e a qualidade das soluções. Aplicou-se vários modelos de Algoritmos Evolutivos para Otimização Multi-Objetivo. O desempenho de cada modelo foi avaliado por métricas de performance encontradas na literatura.

# Abstract

The Biological Sequence Alignment is a basic operation in Bioinformatics since it serves as a basis for other processes, i.e. determination of the protein's three-dimensional structure. Due to the large amount of data involved, mathematical and computational methods have been used to solve this problem. Traditionally, the Biological Alignment Sequence Problem is formulated as a single optimization problem. Each solution has a score that reflects the similarity between sequences. Then, the optimization process looks for the best score solution.

The Multi-Objective Optimization solves problems with multiple objectives that must be reached. Frequently, there is a solution set that represents a trade-off between the objectives. Evolutionary Algorithms, which are inspired by Darwin's Evolution Theory, have been applied with success in solving this kind of problems.

This work formulates the Biological Sequence Alignment as a Multi-Objective Optimization Problem in order to find a set of solutions that represent a trade-off between the extension and the quality of the solutions. Several models of Evolutionary Algorithms for Multi-Objetive Optimization have been applied and were evaluated using several performance metrics found on the literature.

# Sumário

# Lista de Figuras

# Lista de Tabelas

Capítulo

1

# Introdução

## 1.1 Contexto

A Biologia Molecular é uma ciência que apresentou avanços muito significativos nas últimas décadas. Os biólogos 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 tratar os problemas da biologia molecular. Um dos problemas principais dentro da biologia molecular é a comparação de seqüências. A técnica mais freqüentemente usada para comparar duas ou mais seqüências de nucleotídeos ou aminoácidos consiste em sobrepor uma cadeia sobre outra e buscar semelhanças e diferenças, a qual é formalmente denominada alinhamento.

A comparação de seqüências tem um papel central na era pós-genômica. Lecompte e colaboradores [Lecompte et al., 2001] apontam algumas informações obtidas a partir da comparação de seqüências:

1. Determinação da função biológica mediante homologia de seqüências.

2. Busca de padrões conservados ao longo da evolução, que pode servir para buscar estruturas conservadas, sinais de localização ou resíduos funcionais que podem descrever uma família ou sub-família de proteínas.

3. Estudos evolutivos para definir relações filogenéticas entre seqüências.

4. Organização dos domínios dentro de uma família protéica.

5. Construção da estrutura molecular a partir do alinhamento de nucleotídeos com uma proteína de estrutura conhecida e, conseqüentemente, determinar sua função biológica.

Os primeiros algoritmos usados para comparação de seqüências apareceram na década do 70, usando a técnica de programação dinâmica. Posteriormente, surgiram modelos heurísticos e probabilísticos. Técnicas de aprendizado de máquina como redes neurais, algoritmos evolutivos e *simulated annealing* foram também desenvolvidas [Lecompte et al., 2001].

## 1.2 Motivação

Os Algoritmos Evolutivos (AEs) são inspirados na teoria da evolução de Darwin. Para cada problema, inicialmente, gera-se aleatoriamente várias soluções que são agrupadas em um conjunto chamado de população inicial. Posteriormente, aplicando operadores genéticos de reprodução é possível gerar uma nova população de soluções melhor *adaptadas*. Em outros termos, os indivíduos da nova população gerada possuem uma melhor *aptidão* que os da população anterior. Após várias aplicações desse processo de geração de novas populações é possível chegar a uma população de soluções ótimas para um problema.

Dentro dos Algoritmos Evolutivos destacam-se os Algoritmos Genéticos que são utilizados com muito sucesso na atualidade [Deb, 2001]. O funcionamento de um AG é muito simples: dado um problema, gera-se um conjunto de soluções iniciais chamado população inicial. Logo as melhores soluções, avaliadas por uma medida de aptidão, são selecionadas e combinadas entre si para gerar uma nova população. Este procedimento é iterativo até chegar à solução ótima ou atingir um critério de parada. Em problemas reais, entretanto, existem múltiplos objetivos possivelmente conflitantes, a serem atingidos, o que deu lugar ao surgimento de técnicas de Algoritmos Genéticos Multi-Objetivo. A idéia é encontrar um conjunto de soluções que sejam as soluções ótimas para todos os objetivos. Este fato implica que nenhuma solução dentro desse conjunto é melhor que a outra; essas soluções representam na verdade um "compromisso" entre vários objetivos.

O problema de alinhamento de seqüências biológicas pode ser descrito como a descoberta de sobreposições de seqüências cuja pontuação, dada por uma métrica de semelhança, seja a ótima. Na literatura, é possível encontrar aplicações de Algoritmos Genéticos para o alinhamento de seqüências [Notredame e Higgins, 1996, Zhang e Wong, 1997]. É particularmente interessante o trabalho de Romero Zaliz [Romero Zaliz, 2001] que usa os Algoritmos

Genéticos Multi-Objetivo para encontrar um conjunto de alinhamentos ótimos, tendo como objetivos a extensão e qualidade das soluções. O presente trabalho usa essas mesmas idéias, porém aplicadas ao alinhamento de proteínas. Além disso, compara os modelos de Algoritmos Genéticos Multi-Objetivo mais recentes da literatura.

## 1.3 Objetivo

O objetivo deste trabalho é apresentar três modelos de Algoritmos Evolutivos Multi-Objetivo para o alinhamento de seqüências protéicas. Os resultados fornecidos por cada algoritmo serão avaliados conforme as métricas de desempenho achadas na literatura.

## 1.4 Estrutura do trabalho

O presente trabalho está dividido em cinco capítulos. O Capítulo 2 apresenta o problema de Alinhamento de Seqüências Biológicas e uma descrição das principais técnicas usadas neste problema. O Capítulo 3 introduz os conceitos básicos de Otimização Multi-Objetivo e ilustra as principais técnicas tradicionais aplicadas a este tipo de problema. O Capítulo 4 apresenta os Algoritmos Evolutivos enfatizando os Algoritmos Genéticos para estudar profundamente os principais modelos de Algoritmos Genéticos para Otimização Multi-Objetivo. O Capítulo 5 formula o problema de Alinhamento de Seqüências Biológicas como um Problema de Otimização Multi-Objetivo. Ainda neste capítulo, três Algoritmos Evolutivos para Otimização Multi-Objetivo são propostos para a solução do problema. Finalmente, são mostrados os resultados dos experimentos realizados para cada modelo, que foram avaliados conforme métricas de desempenho mais freqüentes da literatura sendo, então, formuladas as conclusões e as possíveis extensões do presente trabalho.

Capítulo

2

# Alinhamento de Seqüências Biológicas

Este capítulo trata sobre os principais tópicos do problema de Alinhamento de Seqüências Biológicas. A primeira seção fornece os conhecimentos necessários sobre Biologia Molecular. Na segunda seção é definido o problema de Alinhamento de Seqüências Biológicas na área de Bioinformática. A terceira seção apresenta as técnicas mais usadas para o alinhamento de seqüências. Finalmente, na quarta seção, são resumidas as principais conclusões deste capítulo.

## 2.1 Conceitos Básicos de Biologia Molecular

Nesta seção são apresentados conceitos básicos de Biologia Molecular, necessários para a área de Bioinformática. Esta seção está fortemente baseada no Capítulo 1 dos livros Introduction to Computational Molecular Biology [Setubal e Meidanis, 1997] e Introduction to Computational Biology [Waterman, 1996], tendo ambos conceitos fundamentais em biologia molecular. As informações obtidas foram complementadas com o livro Biologia Molecular Básica [Zaha, 2000] que apresenta os temas aqui tratados, de forma mais detalhada.

### 2.1.1 Vida

Na natureza tem-se objetos vivos e inanimados. Os seres vivos podem se mover, reproduzir, crescer e comer, ou seja, têm uma participação ativa no meio ambiente enquanto que os objetos inanimados não interagem deste jeito.

A vida de um ser deve-se ao complexo arranjo de reações químicas que ocorrem dentro dele. Essas reações não param, sendo que o produto de uma reação é consumido pela outra, mantendo ativo o sistema. Porém, os organismos vivos estão constantemente trocando matéria e energia com seu meio.

A vida surgiu por volta de 3,5 bilhões de anos atrás; as primeiras formas eram muito simples e a evolução gerou formas mais complexas com o decorrer dos anos, até as que se conhece atualmente.

Tanto as formas complexas quanto as simples possuem a mesma bioquímica. Os principais atores da química da vida são as moléculas chamadas **proteínas** e **ácidos nucléicos**. A grosso modo, as proteínas são responsáveis pelo que é a vida em um sentido físico enquanto os ácidos nucléicos codificam a informação necessária para produzir proteínas e são responsáveis por passar esta informação às gerações seguintes.

Dado que tais moléculas são fundamentais ao estudo da biologia molecular elas serão tratadas com mais detalhe neste capítulo.

### 2.1.2 Proteínas

Muitas substâncias do nosso organismo são proteínas. Elas possuem diversas funções, podendo ser divididas em grupos. Por exemplo um tipo de proteínas denominadas *enzimas* atuam como catalizadores (aceleradores) de reações químicas. As enzimas são muito importantes devido ao fato de que muitas reações bioquímicas podem tornar-se muito lentas ou então não ocorrer sem elas, sendo, portanto, fundamentais para a vida.

As proteínas são uma cadeia de *aminoácidos* (moléculas mais simples). A Figura 2.1 mostra a estrutura básica dos aminoácidos formada de um átomo de carbono central, chamado carbono $\alpha$, ligado ao grupo amino (-$H_2N$), ao grupo carboxila (-COOH), a um átomo de hidrogênio e a um agrupamento variável, denominado *grupo R* ou *cadeia lateral*.

A cadeia lateral confere ao aminoácido suas propriedades e diferencia um aminoácido do outro. A Tabela 2.1 mostra os 20 diferentes aminoácidos na natureza, os quais são comuns em proteínas.

As proteínas podem se dobrar para formar estruturas em três dimensões. As ligações entre o $C_\alpha$ com o nitrogênio e o carbono podem girar em ângulos $\phi$ e $\psi$ respectivamente como mostrado na Figura 2.2. As cadeias laterais também podem girar, mas este movimento

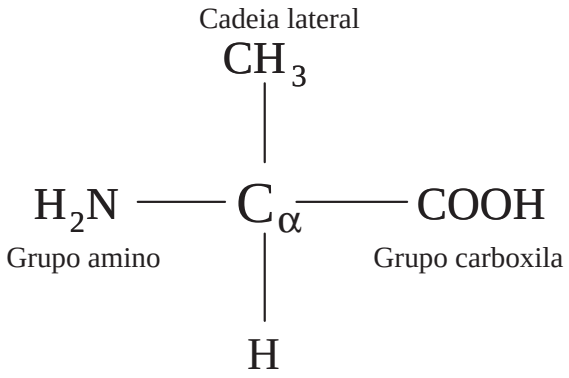



**Figura 2.1:** Aminoácido

| Nro | Letra | Abreviatura | Nome |
|---|---|---|---|
| 1 | A | Ala | Alanina |
| 2 | C | Cys | Cisteína |
| 3 | D | Asp | Aspartato |
| 4 | E | Glu | Glutamato |
| 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 | Aspartato |
| 13 | P | Pro | Prolina |
| 14 | Q | Gln | Glutamato |
| 15 | R | Arg | Arginina |
| 16 | S | Ser | Serina |
| 17 | T | Thr | Treonina |
| 18 | V | Val | Valina |
| 19 | W | Trp | Triptofano |
| 20 | Z | Tyr | Tirosina |

**Tabela 2.1:** Os 20 aminoácidos freqüentemente encontrados em proteínas

é secundário em relação à rotação da espinha dorsal. Encontrando os valores para $\phi$ e $\psi$, pode-se saber como a proteína exatamente se dobra e, também, qual é sua estrutura tridimensional.

As proteínas são produzidas dentro da célula em uma estrutura celular chamada ribossomo, por meio da informação contida em uma molécula, chamada **RNA mensageiro**

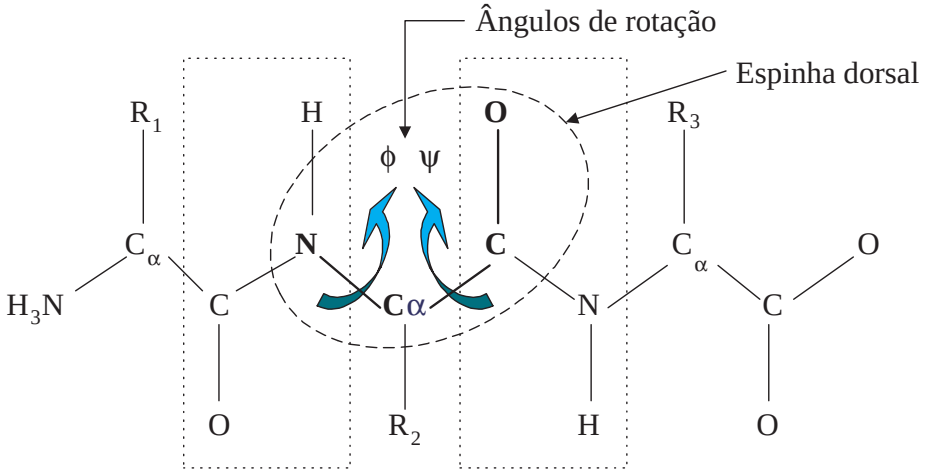



**Figura 2.2: Cadeia polipeptídica** A $R_i$ identifica o componente aminoácido. Os átomos dentro de cada retângulo estão no mesmo plano, os quais giram. Retirado de [Setubal e Meidanis, 1997]

### 2.1.3 Ácidos Nucléicos

Os ácidos nucléicos são macromoléculas de extrema importância biológica em todos os organismos vivos. Eles dão instruções sobre quais proteínas serão sintetizadas e em que quantidade, e também estocam e transmitem a informação genética na célula. Existem dois tipos de ácidos nucléicos: o DNA e o RNA.

#### Ácido Desoxirribonucléico (DNA)

Uma molécula de DNA é uma *cadeia dupla* de moléculas simples. Cada cadeia (fita) tem uma espinha dorsal que consiste em repetições da mesma unidade básica. Esta unidade é formada por uma molécula de açúcar chamada 2'-desoxirribosa ligada a um resíduo fosfato. A molécula de açúcar é formada por 5 carbonos numerados de 1' a 5' (Figura 2.3). A ligação entre as unidades faz-se entre o carbono 3' de uma e os resíduo fosfato e carbono 5' da seguinte. Por esse fato, a molécula de DNA tem uma *orientação* que, por convenção, começa no 5' e acaba no 3'.

Em cada carbono 1' da espinha dorsal encontram-se ligadas moléculas chamadas *bases*. Existem 4 tipos de bases: adenina (A), guanina (G), citosina (C) e timina (T). Ao conjunto formado pelo grupo fosfato, açúcar e base é dado o nome *nucleotídeo*.

As moléculas de DNA estão formadas por duas fitas (fita dupla). Estas duas fitas estão ligadas formando uma dupla hélice, fato este descoberto por Watson e Crick em 1953. As orientações das fitas são 5'→3' e 3'→5' respectivamente, sendo fitas *antiparalelas*. Cada base de uma fita está pareada com uma base na outra. A base A sempre está pareada com a base T e a base C com a base G. A Figura 2.4 mostra um esquema da estrutura do DNA.

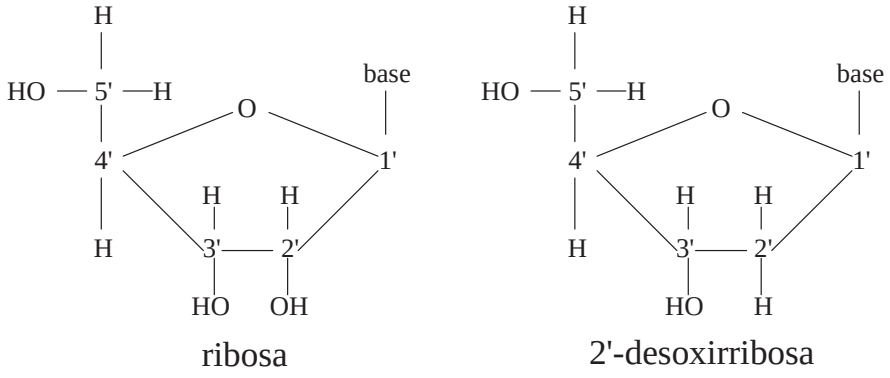



**Figura 2.3:** Açúcares presentes em ácidos nucléicos. Os símbolos 1' até 5' representam os carbonos. A ribosa está presente no RNA e a 2'-desoxirribosa no DNA

As bases A e T, assim como as C e G são chamadas *bases complementares*. Os pares de bases (bp) são a unidade de longitude do DNA. O comprimento de uma molécula de DNA normalmente é muito grande, como por exemplo a célula humana cujas moléculas de DNA têm centenas de milhões de bp.

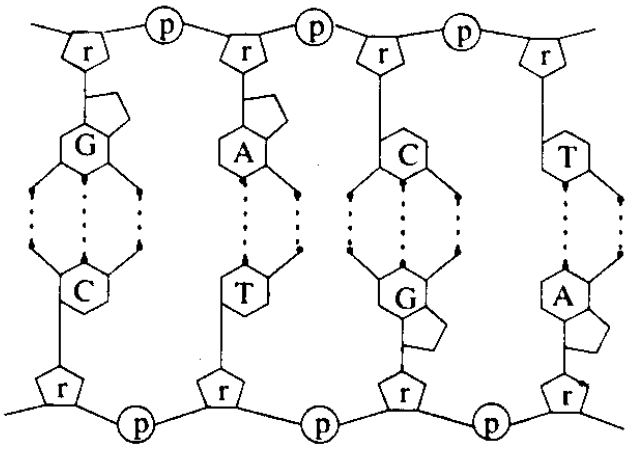



**Figura 2.4:** Vista esquemática da estrutura de dupla fita do DNA

### Ácido Ribonucléico (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'-desoxirribosa é substituída pela ribosa 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.

### 2.1.4 Dogma Central

Pode-se resumir o Dogma Central como o fluxo da informação a partir do DNA até chegar a proteína. A Figura 2.5 mostra o esquema para o Dogma Central.

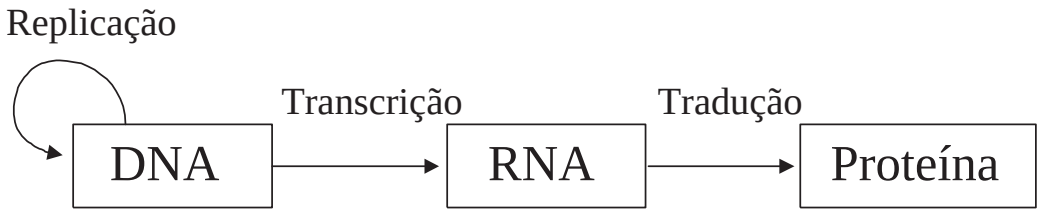



**Figura 2.5:** Esquema do Dogma Central.

O ciclo do DNA para ele mesmo, significa que a molécula pode ser copiada. Isto é chamado *replicação*. O passo do DNA ao RNA é chamado *transcrição* e a formação de proteína é chamada *tradução*.

A idéia geral é que uma macromolécula pode ser usada como modelo para construir outra.

### 2.1.5 Genes e Código Genético

Cada célula de um organismo tem algumas moléculas de DNA muito longas. Cada uma dessas moléculas é chamada cromossomo. Uma coisa importante dentro do DNA é que alguns trechos contíguos codificam informação para construir proteínas, enquanto outros trechos não. Outro detalhe importante é que usualmente cada classe diferente de proteína em um organismo corresponde a um único trecho contíguo ao longo do DNA. Este trecho é conhecido como *gen*. O tamanho do gen varia e, no caso humano, a longitude geralmente é de 10.000 bp. As células têm mecanismos para reconhecer os pontos precisos dentro do DNA onde começa e onde termina um gen.

Como já dito, as proteínas são cadeias de aminoácidos e, portanto, para especificar uma proteína, deve-se especificar quais aminoácidos ela contém. Isso é precisamente o que o gen faz, usando trios (*triplets*) de nucleotídeos chamados *códons*. Na Tabela 2.2 é mostrada a correspondência entre cada trio de nucleotídeos e seu correspondente aminoácido, que é também conhecido como *código genético*. Nota-se que os trios de nucleotídeos são dados com as bases de RNA, em lugar do DNA. Isto acontece porque as moléculas de RNA são o vínculo entre o DNA e as proteínas.

Existem 64 possíveis triplets, mas somente 20 aminoácidos. Portanto diferentes triplets codificam o mesmo aminoácido. Três códons são usados como sinais de parada do gen. Estes estão representados com a palavra STOP (Tabela 2.2). Este código genético é usado para a maioria dos organismos vivos, salvo algumas modificações.

| Primera Posicão | Segunda Posição | | | | Terçeira Posição |
|---|---|---|---|---|---|
| | G | A | C | U | |
| G | Gly | Glu | Ala | Val | G |
| | Gly | Glu | Ala | Val | A |
| | Gly | Asp | Ala | Val | C |
| | Gly | Asp | Ala | Val | U |
| A | Arg | Lys | Thr | Met | G |
| | Arg | Lys | Thr | Ile | A |
| | Ser | Asn | Thr | Ile | C |
| | Ser | Asn | Thr | Ile | U |
| C | Arg | Gln | Pro | Leu | G |
| | Arg | Gln | Pro | Leu | A |
| | Arg | His | Pro | Leu | C |
| | Arg | His | Pro | Leu | U |
| U | Trp | STOP | Ser | Leu | G |
| | STOP | STOP | Ser | Leu | A |
| | Cys | Tyr | Ser | Phe | C |
| | Cys | Tyr | Ser | Phe | U |

**Tabela 2.2:** O Código Genético mapeando códons para aminoácidos

## 2.2 Comparação de Seqüências

A comparação de seqüências é a operação primitiva mais importante em Bioinformática, pois serve como base para outras operações mais complexas. Comparar seqüências significa encontrar quais partes das seqüências são parecidas e quais são diferentes.

Uma das utilidades da comparação de seqüências é determinar um ancestral comum a partir de um conjunto de seqüências. Por exemplo, a partir das seqüências de seres humanos e de alguns primatas é possível saber a seqüência hipotética de uma espécie ancestral de ambos extinta. Se duas seqüências têm um ancestral comuns são ditas *homólogas*. É muito provável que duas seqüências muito parecidas sejam homólogas. Também, que duas seqüências não muito parecidas o sejam. Ainda mais, é possível que duas seqüências sejam parecidas embora não sejam homólogas.

O problema de Comparação de Seqüências começou a ser estudado na década do setenta, por Needleman e Wunsh [Needleman e Wunsch, 1970], Sankoff [Sankoff, 1975], Smith e Waterman [Smith e Waterman, 1981], que basearam-se na programação dinâmica. Recentemente, técnicas de aprendizado de máquina estão sendo desenvolvidas, para casos de alinhamentos mais complexos, como *Simulated Annealing* [Ishikawa et al., 1993, Kim e Cole,

1993] apud [Lecompte et al., 2001], Algoritmos Genéticos [Notredame e Higgins, 1996, Zhang e Wong, 1997, Romero Zaliz, 2001] e outras técnicas iterativas. Um bom resumo dos principais avanços em alinhamento de seqüências pode ser obtido na revisão de Lecompte e colaboradores [Lecompte et al., 2001].

Os principais métodos para comparação de seqüencias são apresentados a seguir.

### 2.2.1 Alinhamento de Seqüências

Uma forma de fazer a comparação de seqüências é mediante o alinhamento. Alinhar seqüências é colocar uma seqüência sobre a outra de forma que a correspondência entre elas fique clara. Exemplo:

GAACGGATTAG
GATCGGAATAG

Nota-se neste exemplo que ambas as seqüências têm o mesmo comprimento. Quando se tem seqüências de comprimentos diferentes, iguala-se as mesmas mediante o uso de caracter nulo "-"(conhecido como gap ou buraco). Se no exemplo anterior tira-se um caracter, obtém-se:

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

**Definição 1** *Dado um alfabeto finito $\mathcal{A}$, sejam $s$ e $t$ duas cadeias sobre $\mathcal{A}$ de comprimento $m$ e $n$, respectivamente. Seja $\mathcal{A}^* = \mathcal{A} \cup \{-\}$ um alfabeto finito onde "-"representa o caracter de buraco (gap). A tupla $(s^*, t^*)$ representa um alinhamento de $s$ e $t$ se as seguintes propriedades são satisfeitas:*

- *As cadeias $s^*$ e $t^*$ possuem o mesmo comprimento, $|s^*| = |t^*|$.*
- *$s_i^*$ e $t_i^* \neq$ -, para $0 \leq i \leq |s^*|$. Ou seja, não existem dois buracos na mesma posição em $s^*$ e $t^*$,*
- *Eliminando os buracos, $s^*$ é reduzido para $s$.*
- *Eliminando os buracos, $t^*$ é reduzido para $t$.*

Para as seqüências de DNA o alfabeto é $\mathcal{A}^* = \{A, G, C, T, -\}$ dado que existem apenas 4 nucleotídeos além do caracter de buraco. O alfabeto $\mathcal{A}^*$ para proteínas tem 21 elementos (incluindo o buraco).

### 2.2.2 Esquemas de Pontuação

Para cada alinhamento é associada uma pontuação. Os esquemas de pontuação para seqüências de DNA e proteínas são apresentados a seguir.

A semelhança entre duas seqüências está definida como a maior pontuação para um alinhamento. Um esquema de pontuação é dado pela tupla $(p, g)$ onde a função $p : \mathcal{A}^* \times \mathcal{A}^* \mapsto \mathbb{R}$ determina a pontuação de cada par de caracteres alinhados e $g$ é usado para penalizar buracos. Normalmente $g < 0$. O esquema de pontuação fornece um valor numérico a cada alinhamento possível.

Dado um par de seqüências $s$ e $t$ e o alinhamento $(s^*, t^*)$ se adiciona $p(a, b)$ cada vez que um caracter $a$ de $s*$ está pareado com um caracter $b$ de $t^*$. Quando um caracter de $s^*$ ou $t^*$ está alinhado a um buraco agregamos $g$ à pontuação. A pontuação total denotada como $score(s^*, t^*)$ é a soma de todas as pontuações em todas as posições do alinhamento $(s^*, t^*)$. Finalmente, a semelhança entre as seqüências $s$ e $t$ é dada por:

$$sim(s, t) = \max_{(s^*,t^*)\in\mathcal{B}} score(s^*, t^*) \tag{2.1}$$

onde $\mathcal{B}$ é o conjunto de todos os possíveis alinhamentos entre $s$ e $t$.

Para seqüências de DNA usa-se normalmente o seguinte esquema de pontuação:

- (a,a) denota que o caracter $s^*$ e $t^*$ na mesma posição são iguais, freqüentemente isto é denominado emparelhamento *(match)*. Para cada match soma-se 1 na pontuação.
- (a,b) denota caracteres distintos na mesma posição em $s^*$ e $t^*$. Também é conhecido como *mismatch*. Cada mismatch soma -1 na pontuação.
- (a,-) ou (-,b) denota a presença de uma letra de $s^*$ alinhada com um buraco em $t^*$ e vice versa. Geralmente cada buraco soma -2 na pontuação.

Este esquema de pontuação com $p(a, a) = 1, p(a, b) = -1$ *e* $g = -2$ é conhecido como *Modelo de Custo Unitário* [Setubal e Meidanis, 1997]. Um exemplo de cálculo é apresentado a seguir.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | - | G | G | A | T | T | A | G |
| G | A | T | C | G | G | A | - | - | A | G |

Valor de um match : 1

Valor de um mismatch : -1

Valor de um gap : -2

Pontuação : 7×1 + 1×-1 + 3×-2 = 0

Para alinhamento de proteínas, o método de pontuação simples aplicado ao DNA não é suficiente. Os aminoácidos que compõem as proteínas possuem propriedades bioquímicas que determinam como eles são substituídos durante a evolução. Por exemplo, existe uma maior probabilidade de que um aminoácido seja substituído por um outro de igual tamanho em lugar de um aminoácido maior. Dado que a comparação de proteínas é feita freqüentemente com critérios evolutivos, é necessário um esquema de pontuação que leve em conta estas probabilidades [Setubal e Meidanis, 1997].

Uma forma de pontuação muito usada envolve as chamadas *matrizes de substituição*. A seguir serão brevemente descritas as duas famílias de matrizes de substituição: PAM [Dayhoff et al., 1978] e BLOSUM [Henikoff e Henikoff, 1991] que são as mais usadas.

### Matriz PAM

As matrizes PAM (*Point Accepted Mutations* ou *Percent of Accepted Mutations*) fornecem uma medida de evolução produzida por, em média, uma mutação por cada 100 aminoácidos. Cada matriz PAM foi desenvolvida para comparar duas seqüências, as quais estão separadas por uma distância especificada em unidades PAM. Por exemplo a matriz PAM120 serve para comparar seqüências que estão separadas por 120 unidades PAM.

Diz-se que duas seqüências $s_1$ e $s_2$ divergem em uma unidade PAM, se a série de mutações aceitas para converter $s_1$ em $s_2$ estão na ordem de uma mutação para cada 100 aminoácidos. Uma mutação diz-se aceita quando não altera o funcionamento da proteína ou quando a mudança na proteína é benéfica para o organismo.

Cada valor $i, j$ da matriz PAM representa a pontuação para substituir um aminoácido $A_i$ por um $A_j$. A Tabela 2.3 mostra a matriz PAM120.

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

**Tabela 2.3:** Matriz de Substituição PAM120

### Matriz BLOSUM

BLOSUM vem do inglês *Block Substitution Matrix*. Um bloco é uma região de um alinhamento local de múltiplas seqüências. Os blocos podem ser obtidos a partir de grupos de seqüências de proteínas relacionadas entre elas. Cada bloco representa as regiões melhor conservadas dentro de uma familia de proteínas [Pietrokovski e Rubin, 2002].

As matrizes BLOSUM estão baseadas nas mudanças dentro de cada bloco. As seqüências de cada bloco são agrupadas colocando duas seqüências no mesmo grupo se a sua porcentagem de identidade é maior que um valor $L\%$. Por exemplo, a matriz BLOSUM50 é obtida a partir de um alinhamento de seqüências com pelo menos 50% de identidade. A Tabela 2.4 mostra a matriz BLOSUM50. Cada valor $(i, j)$ da matriz representa a pontuação por substituir um aminoácido $A_i$ por um $A_j$.

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X | * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 5 | -2 | -1 | -2 | -1 | -1 | -1 | 0 | -2 | -1 | -2 | -1 | -1 | -3 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 | -1 | -5 |
| R | -2 | 7 | -1 | -2 | -4 | 1 | 0 | -3 | 0 | -4 | -3 | 3 | -2 | -3 | -3 | -1 | -1 | -3 | -1 | -3 | -1 | 0 | -1 | -5 |
| N | -1 | -1 | 7 | 2 | -2 | 0 | 0 | 0 | 1 | -3 | -4 | 0 | -2 | -4 | -2 | 1 | 0 | -4 | -2 | -3 | 4 | 0 | -1 | -5 |
| D | -2 | -2 | 2 | 8 | -4 | 0 | 2 | -1 | -1 | -4 | -4 | -1 | -4 | -5 | -1 | 0 | -1 | -5 | -3 | -4 | 5 | 1 | -1 | -5 |
| C | -1 | -4 | -2 | -4 | 13 | -3 | -3 | -3 | -3 | -2 | -2 | -3 | -2 | -2 | -4 | -1 | -1 | -5 | -3 | -1 | -3 | -3 | -2 | -5 |
| Q | -1 | 1 | 0 | 0 | -3 | 7 | 2 | -2 | 1 | -3 | -2 | 2 | 0 | -4 | -1 | 0 | -1 | -1 | -1 | -3 | 0 | 4 | -1 | -5 |
| E | -1 | 0 | 0 | 2 | -3 | 2 | 6 | -3 | 0 | -4 | -3 | 1 | -2 | -3 | -1 | -1 | -1 | -3 | -2 | -3 | 1 | 5 | -1 | -5 |
| G | 0 | -3 | 0 | -1 | -3 | -2 | -3 | 8 | -2 | -4 | -4 | -2 | -3 | -4 | -2 | 0 | -2 | -3 | -3 | -4 | -1 | -2 | -2 | -5 |
| H | -2 | 0 | 1 | -1 | -3 | 1 | 0 | -2 | 10 | -4 | -3 | 0 | -1 | -1 | -2 | -1 | -2 | -3 | 2 | -4 | 0 | 0 | -1 | -5 |
| I | -1 | -4 | -3 | -4 | -2 | -3 | -4 | -4 | -4 | 5 | 2 | -3 | 2 | 0 | -3 | -3 | -1 | -3 | -1 | 4 | -4 | -3 | -1 | -5 |
| L | -2 | -3 | -4 | -4 | -2 | -2 | -3 | -4 | -3 | 2 | 5 | -3 | 3 | 1 | -4 | -3 | -1 | -2 | -1 | 1 | -4 | -3 | -1 | -5 |
| K | -1 | 3 | 0 | -1 | -3 | 2 | 1 | -2 | 0 | -3 | -3 | 6 | -2 | -4 | -1 | 0 | -1 | -3 | -2 | -3 | 0 | 1 | -1 | -5 |
| M | -1 | -2 | -2 | -4 | -2 | 0 | -2 | -3 | -1 | 2 | 3 | -2 | 7 | 0 | -3 | -2 | -1 | -1 | 0 | 1 | -3 | -1 | -1 | -5 |
| F | -3 | -3 | -4 | -5 | -2 | -4 | -3 | -4 | -1 | 0 | 1 | -4 | 0 | 8 | -4 | -3 | -2 | 1 | 4 | -1 | -4 | -4 | -2 | -5 |
| P | -1 | -3 | -2 | -1 | -4 | -1 | -1 | -2 | -2 | -3 | -4 | -1 | -3 | -4 | 10 | -1 | -1 | -4 | -3 | -3 | -2 | -1 | -2 | -5 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | -1 | 0 | -1 | -3 | -3 | 0 | -2 | -3 | -1 | 5 | 2 | -4 | -2 | -2 | 0 | 0 | -1 | -5 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 2 | 5 | -3 | -2 | 0 | 0 | -1 | 0 | -5 |
| W | -3 | -3 | -4 | -5 | -5 | -1 | -3 | -3 | -3 | -3 | -2 | -3 | -1 | 1 | -4 | -4 | -3 | 15 | 2 | -3 | -5 | -2 | -3 | -5 |
| Y | -2 | -1 | -2 | -3 | -3 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | 0 | 4 | -3 | -2 | -2 | 2 | 8 | -1 | -3 | -2 | -1 | -5 |
| V | 0 | -3 | -3 | -4 | -1 | -3 | -3 | -4 | -4 | 4 | 1 | -3 | 1 | -1 | -3 | -2 | 0 | -3 | -1 | 5 | -4 | -3 | -1 | -5 |
| B | -2 | -1 | 4 | 5 | -3 | 0 | 1 | -1 | 0 | -4 | -4 | 0 | -3 | -4 | -2 | 0 | 0 | -5 | -3 | -4 | 5 | 2 | -1 | -5 |
| Z | -1 | 0 | 0 | 1 | -3 | 4 | 5 | -2 | 0 | -3 | -3 | 1 | -1 | -4 | -1 | 0 | -1 | -2 | -2 | -3 | 2 | 5 | -1 | -5 |
| X | -1 | -1 | -1 | -1 | -2 | -1 | -1 | -2 | -1 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | 0 | -3 | -1 | -1 | -1 | -1 | -1 | -5 |
| | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | 1 |

**Tabela 2.4:** Matriz de Substituição BLOSUM50

## 2.2.3 Tipos de Alinhamento

Na literatura encontram-se basicamente três formas diferentes de alinhamentos que são apresentados a seguir.

### Alinhamento Global

Para o Alinhamento Global, o total das posições de um alinhamento $(s^*, t^*)$ é considerado. Por exemplo:

Dadas as seqüências
s = G A A G G A T T A G
t = G A T C G A A G
Tem-se o seguinte alinhamento :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | - | G | G | A | T | T | A | G |
| G | A | T | C | G | G | A | - | - | A | G |

Pontuação : 0

### Alinhamento Local

Um alinhamento local entre duas seqüências $s$ e $t$ acontece entre uma subcadeia de $s$ e uma subcadeia de $t$. A pontuação é calculada apenas sobre a região onde as subcadeias estão alinhadas. Por exemplo:

Dadas as seqüências
s = A A G A C G G
t = G A T C G A A G
Tem-se um possível alinhamento :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | **A** | **A** | **G** | C | G | G |
| G | A | T | C | G | G | **A** | **A** | **G** | | | |

Pontuação : 3

Este tipo de alinhamento é usado quando compara-se grandes seqüências de DNA ou quando duas proteínas compartilham um domínio comum. Apresenta maior sensibilidade para comparar duas seqüências divergentes (menos relacionadas) incluindo aquelas que têm uma origem de evolução comum.

### Alinhamento Semi-Global

Um Alinhamento Semi-Global é aquele onde a pontuação de um alinhamento é calculada ignorando os *buracos terminais* nas seqüências.

Os *buracos terminais* de uma seqüência são aqueles que estão localizados antes do primeiro caracter ou depois do último caracter de uma seqüência.

Com o exemplo pode-se observar, a seguir, a diferença com relação ao alinhamento global:

Dadas as seqüências
s = G A C G A C T T T C C A T T
t = G A T C G T C C

Alinhamento global:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | A | C | G | A | C | T | T | T | C | C | A | T | T |
| G | A | - | - | T | C | - | G | T | C | C | - | - | - |

Pontuação: -8

Alinhamento semi-global:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | A | C | G | A | - | C | T | T | T | C | C | A | T | T |
| - | - | - | G | A | T | C | G | T | - | C | C | - | - | - |

Pontuação: 2

## 2.3 Algoritmos de Comparação de Seqüências

### 2.3.1 Algoritmos de Programação Dinâmica

A idéia básica da programação dinâmica consiste em obter a solução de um problema a partir de soluções menores de sub-problemas análogos. Em outras palavras, pode-se resolver uma instância de um problema usando-se soluções previamente calculadas para o mesmo problema [Setubal e Meidanis, 1997].

#### Alinhamento Global

No caso do alinhamento global de seqüências, usa-se o algoritmo de Needleman-Wunsh [Needleman e Wunsch, 1970]. Dadas duas seqüências $s$ e $t$, de comprimento $i, j$ respectivamente, forma-se uma matriz $A[0 \ldots i, 0 \ldots j]$. Começa-se preenchendo $A[0,0] = 0$, ou em outros termos, a pontuação obtida ao alinhar duas seqüências vazias é zero.

Assim, calcula-se os valores para a primeira coluna $A[m,0] = -gm, 1 \leq m \leq i$ e para a primeira linha $A[n,0] = -gn, 1 \leq n \leq j$. Isso significa alinhar as subcadeias $s_{1\ldots m}$ e $t_{1\ldots n}$ com a seqüência nula, o que faz com que a pontuação seja o comprimento da subcadeia multiplicado pelo custo de um buraco (g). Por exemplo, dadas as seqüências $s = PAWHEAE$, $t = HEAGAWGHEE$ e um custo de buraco $g = 8$, tem-se a seguinte matriz $A$:

| s/t | | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -8 | -16 | -24 | -32 | -40 | -48 | -56 | -64 | -72 | -80 |
| P | -8 | | | | | | | | | | |
| A | -16 | | | | | | | | | | |
| W | -24 | | | | | | | | | | |
| H | -32 | | | | | | | | | | |
| E | -40 | | | | | | | | | | |
| A | -48 | | | | | | | | | | |
| E | -56 | | | | | | | | | | |

Os valores restantes são preenchidos de cima para baixo e da esquerda para a direita. O valor para $A[m, n]$ pode ser calculado a partir dos valores de $A[m-1, n-1]$, $A[m-1, n]$ e $A[m, n-1]$. Existem três possibilidades para obter o alinhamento ótimo entre $s_{1...m}$ e $t_{1...n}$ :

1. Alinhar $s_{1...m}$ e $t_{1...n-1}$ e um buraco com $t_n$.
2. Alinhar $s_{1...m-1}$ e $t_{1...n-1}$ e $s_m$ com $t_n$.
3. Alinhar $s_{1...m-1}$ e $t_{1...n}$ e $s_m$ com um buraco.

Cada uma destas possibilidades tem uma pontuação associada. Seja $p(s_m, t_n)$ o valor associado a um match ou mismatch, e $g$ o valor dado a um gap. Conseqüentemente, o valor de $A[m, n]$ pode ser calculado com a seguinte relação:

$$A[m,n] = \max \begin{cases} A[m-1, n-1] + p(s_m, t_n) \\ A[m-1, n] - g \\ A[m, n-1] - g. \end{cases} \tag{2.2}$$

Usando a matriz BLOSUM50 para calcular as pontuações $p$ e $g = 8$, obtém-se a seguinte matriz:

| s/t | | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -8 | -16 | -24 | -32 | -40 | -48 | -56 | -64 | -72 | -80 |
| P | -8 | -2 | -9 | -17 | -25 | -33 | -42 | -49 | -57 | -65 | -73 |
| A | -16 | -10 | -3 | -4 | -12 | -20 | -28 | -36 | -44 | -52 | -60 |
| W | -24 | -18 | -11 | -6 | -7 | -15 | -5 | -13 | -21 | -29 | -37 |
| H | -32 | -14 | -18 | -13 | -8 | -9 | -13 | -7 | -3 | -11 | -19 |
| E | -40 | -22 | -8 | -16 | -16 | -9 | -12 | -15 | -7 | 3 | -5 |
| A | -48 | -30 | -16 | -3 | -11 | -11 | -12 | -12 | -15 | -5 | 2 |
| E | -56 | -38 | -24 | -11 | -6 | -12 | -14 | -15 | -12 | 9 | 1 |

À medida que os valores $A[m, n]$ são preenchidos, mantém-se um apontador à célula da qual este foi derivado ($A[m-1, n-1]$, $A[m-1, n]$ ou $A[m, n-1]$). Logo, para obter o alinhamento ótimo percorre-se os apontadores desde a última posição $A[i, j]$ até o início da matriz em $A[0, 0]$. Para o exemplo, a pontuação final é de 1 e o alinhamento resultante é:

```
HEAGAWGHE-E
--P-AW-HEAE
```

### Alinhamento Local

O algoritmo usado para alinhamentos locais é conhecido como algoritmos de Smith-Waterman [Smith e Waterman, 1981]. Ao invés de procurar um alinhamento global, busca-se os melhores alinhamentos entre subseqüências de $s$ e $t$. Este algoritmo é simplesmente uma modificação do algoritmo para alinhamento global. A primeira diferença é que existe uma opção a mais para o preenchimento da matriz $A[m, n]$ que é calculado usando:

$$A[m,n] = \max \begin{cases} 0, \\ A[m-1, n-1] + p(s_m, t_n) \\ A[m-1, n] - g \\ A[m, n-1] - g. \end{cases} \tag{2.3}$$

Isso significa, também, que os valores para $A[m, 0]$ e $A[0, n]$ são inicializados com zeros. Para obter o alinhamento ótimo procura-se a célula que tenha o maior valor e, a partir dela, percorre-se os apontadores até chegar a um $A[m, n]$ cujo valor seja 0. O exemplo seguinte ilustra a matriz $A$ para o alinhamento local utilizando as mesmas seqüências que no caso do alinhamento global.

| s/t | | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 5 | 0 | 5 | 0 | 0 | 0 | 0 | 0 |
| W | 0 | 0 | 0 | 0 | 2 | 0 | 20 | 12 | 4 | 0 | 0 |
| H | 0 | 10 | 2 | 0 | 0 | 0 | 12 | 18 | 22 | 14 | 6 |
| E | 0 | 2 | 16 | 8 | 0 | 0 | 4 | 10 | 18 | 28 | 20 |
| A | 0 | 0 | 8 | 21 | 13 | 5 | 0 | 4 | 10 | 20 | 27 |
| E | 0 | 0 | 6 | 13 | 18 | 12 | 4 | 0 | 4 | 16 | 26 |

A pontuação final do alinhamento é 28. O alinhamento obtido é o seguinte:

```
AWGHE
AW-HE
```

### Alinhamento Semi-Global

O alinhamento semi-global é usado para alinhar uma seqüência contida em uma outra ou quando duas seqüências estão sobrepostas. Para obter a pontuação do alinhamento não são levados em conta os buracos terminais. Para preencher a matriz $A[m, n]$ é usada a equação 2.2 do alinhamento global.

Porém, tem-se a opção de não contar para a pontuação os buracos terminais antes do primeiro caracter da primeira ou da segunda seqüência ou ambas. Para o primeiro caso, preenche-se a primeira linha da matriz $A$ com zeros. No segundo caso, preenche-se a primeira coluna de $A$ com zeros. Caso contrário, tanto a primeira linha quanto a primeira coluna são inicializadas com zero.

Para recuperar o alinhamento existem três casos: levar em conta os buracos terminais depois do último caracter da primeira seqüência, da segunda ou ambas. No primeiro caso, busca-se o valor máximo da última linha de $A$. No segundo caso, busca-se o valor máximo na última coluna de $A$. Caso contrário é buscado o valor máximo da última linha e coluna de $A$. Uma vez obtido este valor basta percorrer a matriz seguindo os apontadores tal como foi feito no alinhamento global. A Tabela 2.5 resume o procedimento conforme os diferentes casos para o alinhamento semi-global.

| Não considerar ... | Fazer ... |
|---|---|
| Buracos no começo da primeira seqüência | Preencher a primeira linha com zeros |
| Buracos no começo da segunda seqüência | Preencher a primeira coluna com zeros |
| Buracos no final da primeira seqüência | Buscar o valor máximo na última linha |
| Buracos no final da segunda seqüência | Buscar o valor máximo na última coluna |

**Tabela 2.5:** Os diferentes casos do alinhamento semi-global

O exemplo seguinte ilustra a matriz $A$ para o alinhamento semi-global usando as mesmas seqüências que no caso do alinhamento global. Não são considerados os buracos terminais da primeira nem os da segunda seqüência.

| s/t | | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | -2 | -1 | -1 | -2 | -1 | -4 | -2 | -2 | -1 | -1 |
| A | 0 | -2 | -2 | 4 | -1 | 3 | -4 | -4 | -4 | -3 | -2 |
| W | 0 | -3 | -5 | -4 | 1 | -4 | 18 | 10 | 2 | 6 | -6 |
| H | 0 | 10 | 2 | 6 | -6 | -1 | 10 | 16 | 20 | 12 | 4 |
| E | 0 | 2 | 16 | 8 | 0 | 7 | 2 | 8 | 16 | 26 | 18 |
| A | 0 | -2 | 8 | 21 | 13 | 5 | 3 | 2 | 8 | 18 | 25 |
| E | 0 | 0 | 4 | 13 | 18 | 12 | 4 | 4 | 2 | 14 | 24 |

Pode ser visto que a primeira linha e a primeira coluna são zero, o valor máximo da última linha e última coluna é 25, que é a pontuação do alinhamento final:

```
GAAWGHEE
PAW-HEEA
```

Os algoritmos vistos nesta subseção são os mais básicos que existem para programação dinâmica. Existem múltiplas variações destes algoritmos para tarefas específicas. Por exemplo, para incluir o critério de afinamento de buracos (o primeiro buraco de uma série consecutiva recebe uma penalidade maior) o algoritmo é mais complexo e requer duas matrizes auxiliares. Existem versões do algoritmo de programação dinâmica com o critério de afinamento de buracos para o alinhamento global, local e semi-global. Tais tipos de algoritmos não serão detalhados aqui, mas foram usados para calcular as pontuações durante os experimentos do Capítulo 5.

### 2.3.2 Algoritmo BLAST

A familia de algoritmos BLAST (Basic Local Alignment Search Tool) [Altschul et al., 1990] retorna os alinhamentos locais de maior pontuação entre uma seqüência de consulta e uma base de dados de seqüências. Dentro dos algoritmos BLAST, destacam-se o BLASTP usado para o alinhamento de proteínas e o BLASTA para alinhar seqüências de DNA. Todos os algoritmos BLAST funcionam de maneira muito semelhante.

A idéia básica do algoritmos BLAST é que os melhores alinhamentos contêm provavelmente dentro deles segmentos pareados de alta pontuação (High-scoring Segment Pairs, HSP). Um *segmento* é uma sub-cadeia de uma seqüência. Dadas duas seqüências, um *segmento pareado* entre elas é um par de segmentos do mesmo comprimento, um de cada seqüência. Um exemplo de segmento pareado e sua pontuação usando a matriz BLOSUM50 é mostrado na Tabela 2.6.

| | | | | |
|---|---|---|---|---|
| A | W | G | H | E |
| A | W | H | H | E |
| 5 | 15 | -2 | 10 | 6 |
| Total : 34 | | | | |

**Tabela 2.6:** Exemplo de um segmento pareado.

Um segmento pareado máximo (Maximun Segment Pair, MSP) entre duas seqüências é obviamente aquele segmento pareado de maior pontuação. O Algoritmo BLAST consiste dos seguintes três passos:

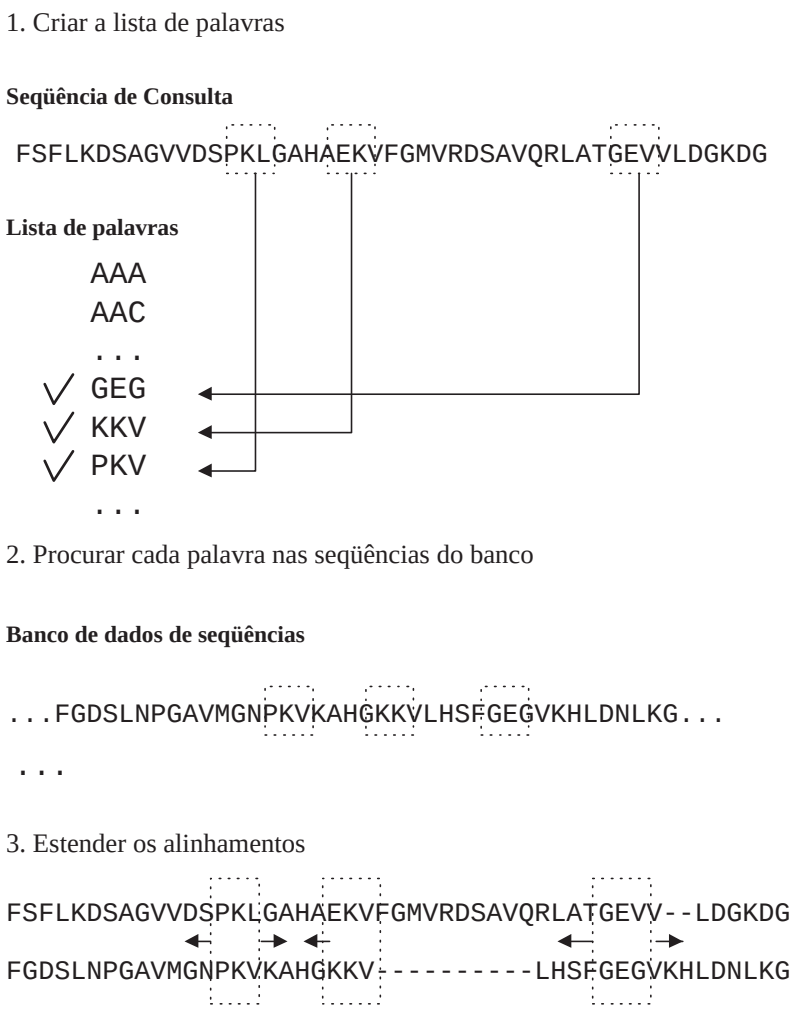



**Figura 2.6:** Esquema do Algoritmo BLAST

1. Dada uma longitude de palavra $w$ e uma matriz de pontuação, cria-se uma lista de todas as palavras ($w$-mers) de tamanho $w$, cuja pontuação seja maior que um $T$ (*threshold*); quando são alinhadas com as palavras da seqüência de consulta. Estas palavras são conhecidas também como sementes.

2. Cada semente é procurada em cada seqüência do banco de dados.

3. Caso a semente seja encontrada em uma seqüência do banco de dados, tenta-se extender o alinhamento (sem usar buracos) com a seqüência de consulta em ambas as direções, enquanto a pontuação deste não seja menor que um limite $S$. Isto permite que os melhores segmentos pareados de alta pontuação sejam mantidos, embora exista a probabilidade de perder algumas extensões importantes [Setubal e Meidanis, 1997].

Normalmente o tamanho da semente é fixado em 3 para seqüências de proteínas e 11 para o caso de seqüências de DNA. Finalmente, BLAST retorna como resultado todos aque-

les alinhamentos contendo os segmentos pareados de alta pontuação entre a seqüência de consulta e o banco de dados. A Figura 2.6 mostra o esquema de funcionamento do algoritmo BLAST.

### 2.3.3 Algoritmo FAST

A família de algoritmos FAST [Pearson e Lipman, 1978] representa outra técnica heurística muito usada para busca em banco de dados de seqüências. Dentro dos algoritmos FAST, destaca-se o FASTA para alinhar seqüências de DNA e proteínas.

A idéia básica do algoritmo FAST é comparar cada seqüência do banco de dados com a seqüência procurada e reportar aquelas que tenham semelhança significativa [Setubal e Meidanis, 1997].

Dadas duas seqüências $s$ e $t$ o algoritmo FAST começa determinando as *k-tuplas* comuns entre ambas seqüências. Uma *k-tupla* é simplesmente uma palavra de tamanho $k$, sendo este parâmetro conhecido como *ktup*. Além disso, é necessário um valor de deslocamento (*offset*) de uma tupla comuns para ambas as seqüências. Por exemplo, se existe uma tupla em comum na posição $i$ da primeira seqüência e na posição $j$ da segunda, o deslocamento será de $i - j$. Se $|s| = m$ e $|t| = n$, o valor de deslocamento pode estar entre $-n+1$ e $m+1$. A Tabela 2.7 ilustra as possíveis k-tuplas de tamanho 1 para a seqüência $s$ =`HARFYAAQIVL`

| ktup | posições |
|---|---|
| A | 2,6, 7 |
| F | 4 |
| H | 1 |
| I | 9 |
| L | 11 |
| Q | 8 |
| R | 3 |
| V | 10 |
| Y | 5 |

**Tabela 2.7:** Exemplo de K-tuplas.

Como exemplo é mostrado o cálculo dos deslocamentos em relação à seqüência $t$ = `VDMAAQIA`. O resultado pode ser visto na Tabela 2.8.

| seq | V | D | M | A | A | Q | I | A |
|---|---|---|---|---|---|---|---|---|
| Deslocamentos | 9 | | -2, 2,3 | -3, 1, 2 | 2, | 2, | -6,-2,-1 | |

**Tabela 2.8:** Deslocamentos de $s$ em relação a $t$

Após realizado esse cálculo constrói-se um *vetor de deslocamento* onde são guardadas as freqüências para cada valor de deslocamento. Conforme os dados da Tabela 2.8 gera-se o vetor mostrado na Tabela 2.9.

| valor deslocamento | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Freqüência | 0 | 1 | 0 | 0 | 1 | 2 | 1 | 0 | 1 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**Tabela 2.9:** Vetor de deslocamento

Pode ser observado que no deslocamento 2 se apresenta uma maior freqüência (4), o que significa que a maioria dos *matchs* são encontrados nesse deslocamento. Este método é conhecido também como método da diagonal, dado que pode ser interpretado como uma diagonal em uma matriz de programação dinâmica [Setubal e Meidanis, 1997].

Depois, o algoritmo FAST junta duas ou mais $k-tuplas$ que estão na mesma diagonal formando uma *região*. Posteriormente, as regiões recebem uma pontuação conforme uma matriz como BLOSUM ou PAM. Este processo é repetido para cada seqüência do banco de dados e as regiões com melhor pontuação são processadas com um algoritmo de programação dinâmica restrito a uma banda ao redor das regiões. É no algoritmo FAST que os melhores alinhamentos são encontrados. O esquema do algoritmo FAST pode ser visto na Figura 2.7.

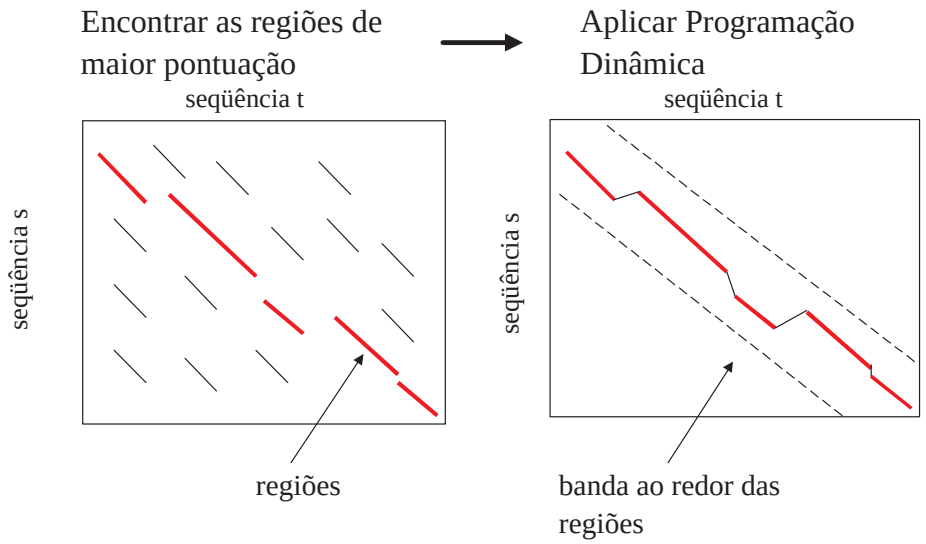



**Figura 2.7:** Esquema do Algoritmo FAST

O parâmetro $k$ (ktup) influi na performance do algoritmo em termos de sensibilidade e seletividade. Sensibilidade está relacionada com a capacidade para reconhecer seqüências relacionadas, embora distantes. Seletividade é a capacidade para descartar falsos positivos - matches entre seqüências não relacionadas. Geralmente sensibilidade e seletividade são metas opostas. Um valor baixo para $k$ incrementa a sensibilidade, enquanto um valor alto de $k$ favorece a seletividade [Setubal e Meidanis, 1997].

### 2.3.4 Algoritmos Probabilísticos

Do ponto de vista probabilístico usa-se os *Modelos Ocultos de Markov (HMMs)* para realizar alinhamentos e buscar semelhanças.

O Modelo Oculto de Markov (HMM) é um modelo estatístico bastante adequado para muitas tarefas em biologia molecular. Este modelo tem sido utilizado para o reconhecimento da fala desde o início dos anos 70 e foi introduzido na biologia molecular por volta de 1980. O uso mais popular do HMM em biologia molecular é como um "perfil probabilístico"de uma família de proteínas ou de padrões de DNA, chamado $profileHMM$. Este modelo foi inicialmente proposto por Krogh e colaboradores [Krogh et al., 1993] e atualmente é usado para procurar em bases de dados por proteínas homólogas e para gerar alinhamento múltiplo de seqüências [Rodrigues, 2001].

Existem vários softwares que implementam este método como o HMMER, PFTOOLS, HMMpro, SAM, entre outros. Os modelos HMMs baseiam-se em dados probabilísticos. Por exemplo, para criar um HMM a partir de um conjunto de seqüências treina-se o modelo, de modo a fazer-se contagem da freqüência para cada letra nas seqüências para obter probabilidades associadas a cada estado do modelo. Este processo é repetido até chegar a um alinhamento com a melhor pontuação possível.

## 2.4 Comentários Finais

Neste capítulo foi tratado o problema de Alinhamento de Seqüências Biológicas e as principais técnicas usadas para resolvê-lo. Todas estas técnicas têm suas vantagens e desvantagens, que são comentadas a seguir.

O algoritmo de Programação Dinâmica avalia todos os possíveis alinhamentos entre duas seqüências e fornece o alinhamento ótimo. Porém, o número possível de alinhamentos entre duas seqüências aumenta exponencialmente quanto maiores forem os comprimentos das seqüências. Por exemplo, para duas seqüências de 1000 nucleotídeos existem aproximadamente $10^{767.4}$ alinhamentos possíveis. [Waterman, 1996]. Outro problema muito comum em programação dinâmica é a dimensionalidade [Goldberg, 1989], é necessária uma capacidade de memoria da ordem $O(n^2)$ para alinhar duas seqüências de comprimento $n$. Na prática, o algoritmo de Programação Dinâmica não é usado para alinhar mais de dez seqüências [Lecompte et al., 2001]. Embora existam melhoras em relação aos requisitos de memória, estes ainda continuam elevados.

Os algoritmos heurísticos (BLAST, FAST) apresentam a vantagem de serem muito rápidos e poderem fazer comparações com grandes bancos de seqüências. O espaço de busca é percorrido filtrando aquelas soluções que sejam relevantes do ponto de vista estatístico.

A desvantagem é que eles possuem vários parâmetros (nas seções 2.3.2 e 2.3.3 foram vistos apenas os principais) que influenciam na qualidade de resultados obtidos. Os algoritmos heurísticos garantem boas soluções, porém não a solução ótima, ou seja, existe um compromisso entre rapidez e otimalidade. Os modelos probabilísticos dependem fortemente do conjunto de treinamento fornecido. Assim, se não existe um bom conjunto de exemplos, os modelos probabilísticos não são aplicáveis.

Conforme exposto neste capítulo, existem diversos tipos de alinhamentos de seqüências mas nenhum algoritmo aplicado permite encontrar todos os tipos de alinhamento. No caso da programação dinâmica, existem variantes para cada caso. Os algoritmos heurísticos somente encontram alinhamentos locais. Nos capítulos seguintes serão apresentados os conceitos de Otimização Multi-Objetivo e Computação Evolutiva, que fornecem uma nova formulação do problema de Alinhamento de Seqüências e que procuram resolver os problemas mencionados anteriormente mediante novos critérios de avaliação para um alinhamento.

Capítulo

3

# Otimização Multi-Objetivo

Neste capítulo serão apresentadas noções básicas de Otimização Multi-Objetivo. O conteúdo está baseado fortemente nos capítulos 2 e 3 do livro de K. Deb [Deb, 2001]. Este capítulo está dividido em quatro seções. A primeira seção apresenta os principais conceitos de Otimização Multi-Objetivo. A segunda seção detalha o conceito de Dominância de Pareto. A terceira seção descreve as principais técnicas matemáticas usadas nestes problemas. Finalmente na quarta seção são apresentados os comentários finais.

## 3.1 Conceitos Básicos

Um Problema de Otimização Multi-Objetivo (MOOP, do inglês *Multi-Objective Optimization Problem*) trabalha com mais de uma função objetivo. Muitos problemas de tomada de decisões implicam vários critérios que devem ser balanceados. Técnicas tradicionais de otimização têm sido usadas para solucionar estes problemas no passado. Estas técnicas originalmente foram formuladas para trabalhar com uma única função objetivo e encontrar uma solução ótima. Assim, os vários objetivos são combinados em uma função. Porém, existem diferenças fundamentais entre a otimização multi-objetivo e de objetivo simples, que serão detalhadas neste capítulo.

### 3.1.1 Formulação

Um MOOP possui um número de funções objetivo a serem otimizadas (maximizar ou minimizar). Além disso, possui restrições que devem de ser satisfeitas por qualquer solução factível. O enunciado geral para um MOOP é o seguinte [Deb, 2001] :

$$\left.\begin{array}{rll} \text{maximizar/minimizar} & f_m(\mathbf{x}), & m = 1, 2, \ldots, M \\ \text{restrita a} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}\right\} \tag{3.1}$$

onde $\mathbf{x}$ é o vetor de $n$ variáveis de decisão $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. Os valores $x_i^{(L)}$ e $x_i^{(U)}$, representam o mínimo e máximo valor respectivamente para a variável $x_i$. Estes limites definem o *espaço de variáveis de decisão* ou *espaço de decisão* $D$. Além, o vetor $\mathbf{x}$ será referido também como *solução*.

As $J$ desigualdades ($g_j$) e as $K$ igualdades ($h_k$) são chamadas de funções de restrição. Uma solução $\mathbf{x}$ factível será aquela que satisfaça as $J+K$ funções de restrição e os $2n$ limites. Caso contrário a solução será não factível. O conjunto de todas as soluções factíveis formam a *região factível* ou *espaço de busca* $S$.

Cada uma das $M$ funções objetivo $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}$ pode ser maximizada ou minimizada. Porém, para trabalhar com os algoritmos de otimização, é necessário converter todas para serem maximizadas ou minimizadas. O vetor funções objetivo $\mathbf{f}(\mathbf{x})$ conforma um espaço multi-dimensional chamado *espaço de objetivos* $Z$. Então para cada solução $\mathbf{x}$ no espaço de decisão, existe um $\mathbf{f}(\mathbf{x})$ 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 $\mathbf{x}$ ($n$-dimensional) e um vetor $\mathbf{f}(\mathbf{x})$ ($M$-dimensional). Por exemplo, se cada elemento de $\mathbf{x}$ e $\mathbf{f}(\mathbf{x})$ são números reais, então $\mathbf{f}(\mathbf{x})$ estaria mapeada como $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^M$.

### 3.1.2 Soluções Pareto-ótimas

Tomar decisões implica um processo que consiste em vários fatores, com o objetivo de encontrar a melhor solução. Em alguns casos, podem aparecer várias soluções boas, das quais nenhuma é quantitavamente melhor que a outra. Por exemplo, na hora de comprar um carro suponha-se que se está procurando o preço e conforto. A Figura 3.1 ilustra várias alternativas de escolha.

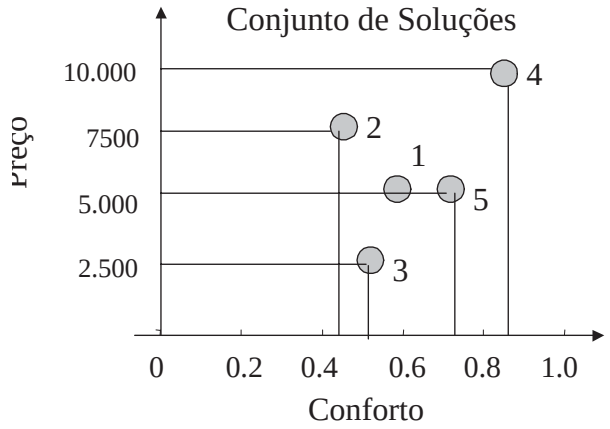



**Figura 3.1:** Exemplo que ilustra várias opções de compra de carro (1-5), considerando o seu custo e conforto.

O objetivo é minimizar o custo e maximizar o conforto. Neste casso 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. Tem-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*. Estes conjuntos têm as seguintes propriedades:

1. Qualquer par de soluções do conjunto não dominado devem de ser não dominadas a uma em relação a outra.

2. Quaisquer das soluções não contidas no conjunto não dominado, devem de ser dominadas por no mínimo uma solução no conjunto não dominado.

Seja os pontos não dominados estão em um espaço contínuo, pode-se desenhar uma curva. Todos os pontos contidos na curva formam a *Frente de Pareto* ou *Fronteira de Pareto*.

### 3.1.3 Metas em Otimização Multi-Objetivo

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

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 3.2a mostra-se uma boa distribuição de soluções na fronteira de Pareto, entanto na Figura 3.2b as soluções estão distribuídas apenas em algumas regiões. É necessário assegurar a maior cobertura possível da fronteira, já que implica que tem-se um bom conjunto de soluções "comprometidas"com os objetivos desejados. Como em MOOP 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.

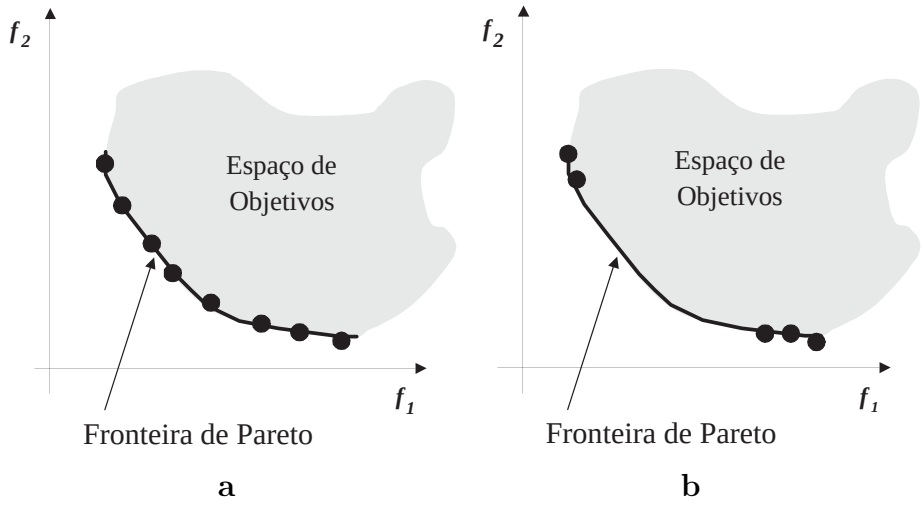



**Figura 3.2:** Distribuição de Soluções na Fronteira de Pareto

### 3.1.4 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 MOOP, 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 MOOP 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 MOOP pode ser convertido mediante algumas técnicas, em um problema de otimização simples.

## 3.2 Operador de Dominância de Pareto

A maioria dos algoritmos para otimização multi-objetivo usam o conceito de dominância. Se existem $M$ funções objetivo $f_j, j = 1, ldots, M$, o operador $\lhd$ entre duas soluções, $x \lhd y$, significa que a solução $x$ é melhor que $y$ em um objetivo em particular. Reciprocamente $x \rhd y$ denota que a solução $x$ é pior que $y$ para algum objetivo. Este operador é usado na seguinte definição [Deb, 2001]:.

**Definição 2** *Uma solução* $\mathbf{x^{(1)}}$ *domina uma outra solução* $\mathbf{x^{(2)}}$ *(representado como* $\mathbf{x^{(1)}} \preceq \mathbf{x^{(2)}}$*) se as condições seguintes são satisfeitas :*

1. *A solução* $\mathbf{x^{(1)}}$ *não é pior que* $\mathbf{x^{(2)}}$ *em todos os objetivos, ou seja* $f_j(\mathbf{x^{(1)}}) \not\rhd f_j(\mathbf{x^{(2)}})$ *para todo* $j = 1, 2, \ldots, M$.

2. *A solução* $\mathbf{x^{(1)}}$ *é estritamente melhor que* $\mathbf{x^{(2)}}$ *pelo menos em um objetivo, ou seja* $f_j(\mathbf{x^{(1)}}) \lhd f_j(\mathbf{x^{(2)}})$ *pelo menos em um* $j = 1, 2, \ldots, M$.

Se ambas as condições são satisfeitas, pode-se dizer que:

1. $\mathbf{x^{(2)}}$ é dominada por $\mathbf{x^{(1)}}$

2. $\mathbf{x^{(1)}}$ é não dominada por $\mathbf{x^{(2)}}$

3. $\mathbf{x^{(1)}}$ é não inferior que $\mathbf{x^{(2)}}$

Na Figura 3.1, a solução 5 domina à solução 1 ($5 \preceq 1$), e a solução 3 domina à solução 2 ($3 \preceq 2$). Portanto o conceito de dominância permite comparar soluções com múltiplos objetivos.

### 3.2.1 Propriedades da Relação de Dominância

A relação de dominância satisfaz seguintes propriedades:

1. Não é reflexiva. Conforme a definição 2 uma solução não pode ser dominada por si mesma.
2. Não é simétrica, porque se $p \preceq q$ não implica que $q \preceq p$.
3. Transitiva, dado que se $p \preceq q$ e $q \preceq r$ então $p \preceq r$.

Estas propriedades caracterizam a relação de dominância como una relação de ordem parcial estrita.

### 3.2.2 Otimalidade de Pareto

Quando o conjunto de soluções é finito, é possível fazer comparação das soluções duas a duas e pode-se dividir o conjunto em soluções dominadas e não dominadas. Portanto tem-se um conjunto não dominado e um conjunto dominado.

**Definição 3** *Dado conjunto de soluções $P$, o conjunto não dominado $P'$ é formado por aquelas soluções que são não dominadas por qualquer elemento de $P$.*

Quando o conjunto $P$ é o espaço completo de busca ($P = S$), o conjunto não dominado $P'$ é chamado de *conjunto Pareto-ótimo*. A Figura 3.3 mostra vários exemplos de conjuntos Pareto-ótimos, conforme várias combinações de objetivos para as funções $f_1$ e $f_2$. A curva indica onde o conjunto está localizado. Então, é possível ter conjuntos Pareto-ótimos formando por uma região contínua ou pela união de regiões descontínuas.

**Definição 4** *O conjunto não dominado para a totalidade do espaço de busca factível $S$, é chamado de* conjunto Pareto-ótimo global.

Portanto as soluções contidas neste conjunto são as soluções ótimas do MOOP. Ótimos locais são definidos como:

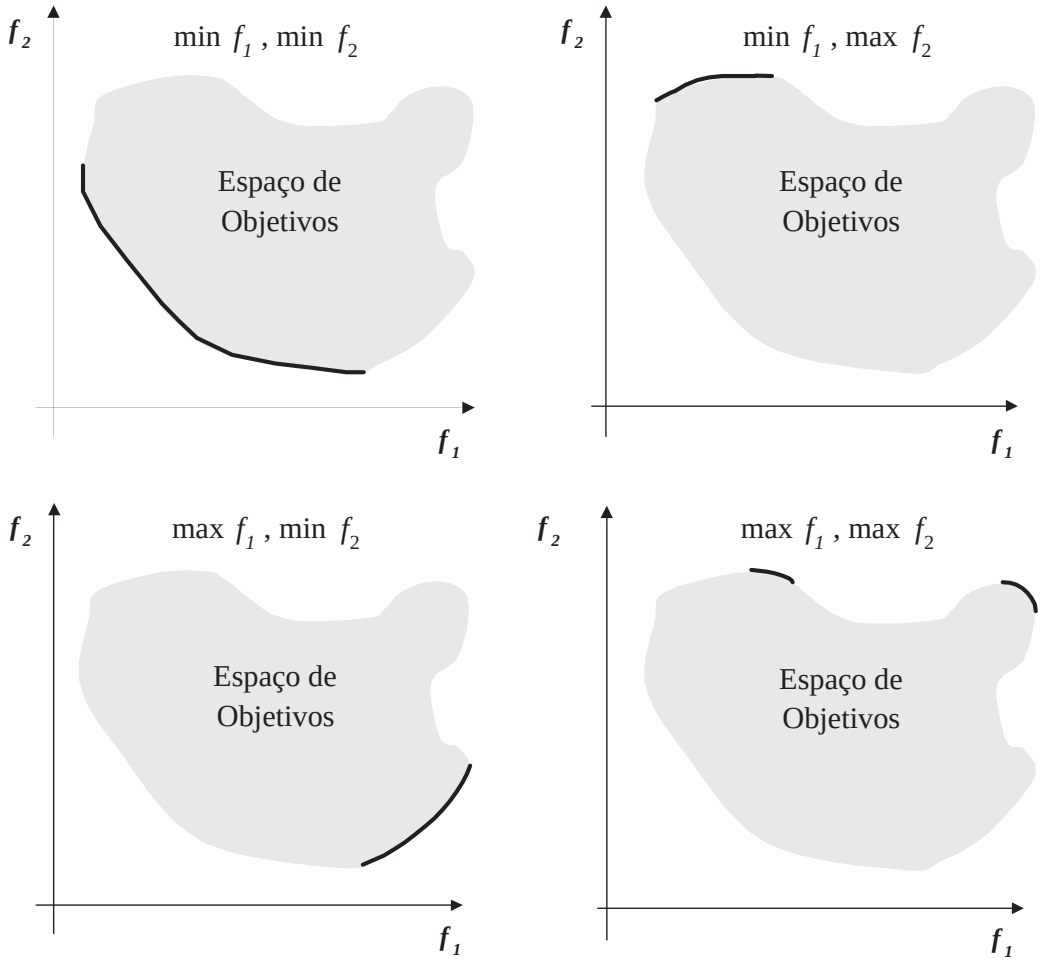



**Figura 3.3:** Vários exemplos de conjuntos Pareto-Ótimos

**Definição 5** *Se cada elemento* $\mathbf{x}$ *do conjunto* $P$ *não é dominado por alguma solução* $\mathbf{y}$ *na vizinhança de* $\mathbf{x}$ *tal que* $\|\mathbf{y}-\mathbf{x}\|_\infty \leq \epsilon$*, onde* $\epsilon$ *é um número positivo arbitrariamente pequeno, então o conjunto* $P$ *é chamado de* conjunto Pareto-ótimo local.

A Figura 3.4 mostra dois conjuntos Pareto-ótimos locais que são não dominados, mostrando a sua vizinhança no seu espaço de objetivos e no espaço de variáveis (à direita).

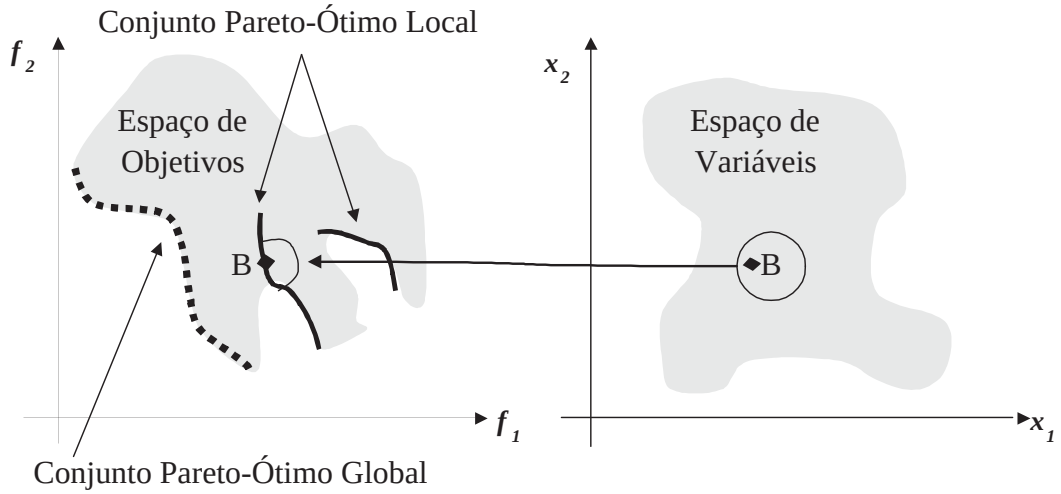



**Figura 3.4:** Soluções Pareto-ótimas locais e globais

**Definição 6** *A Fronteira de Pareto está formada pelo conjunto de vetores de funções objetivo* $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))^T$*, para cada solução* $\mathbf{x}$ *que está no conjunto Pareto-ótimo.*

A Fronteira de Pareto está formada então por os 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 7** *A solução* $\mathbf{x^{(1)}}$ *domina* fortemente *a solução* $\mathbf{x^{(2)}}$ *(representado como* $\mathbf{x^{(1)}} \prec \mathbf{x^{(2)}}$*) se é estritamente melhor à solução* $\mathbf{x^{(2)}}$ *em todos os* $M$ *objetivos*

## 3.3 Técnicas Tradicionais para MOOP

Nesta seção serão descritas as principais técnicas clássicas usadas em MOOP.

### 3.3.1 Somatório de Pesos (Weighted Sum)

O Método de Somatório dos Pesos consiste em criar uma função objetivo somando cada objetivo multiplicado por um peso. Este peso é fornecido como parâmetro. 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$/.4000 a R$/.30000, enquanto o conforto pode estar entre 0% e 100%.

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

$$\left.\begin{array}{lll} \text{minimizar} & F(\mathbf{x}) = \sum_{m=1}^{M} w_m f_m(\mathbf{x}), & \\ \text{restrita a} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}\right\} \tag{3.2}$$

Onde $w_m \in [0, 1]$ é o peso para cada função objetivo $f_m$. Alguns resultados matemáticos são descritas a seguir/

**Teorema 1** *A solução do problema na equação 3.2 é Pareto-ótima se os pesos são positivos para todos os objetivos.*

**Teorema 2** *Se* $\mathbf{x}$ *é uma solução Pareto-ótima para um MOOP convexo, existe um vetor de pesos positivos* $\mathbf{w}$ *tal que* $\mathbf{x}$ *é uma solução para o problema da Equação 3.2*

O Teorema 3.2 garante que quando MOOP é convexo, qualquer solução Pareto-ótima pode ser achada usando o Método de Somatório dos Pesos.

Seja um MOOP com dois objetivos. O espaço de objetivos é a Fronteira de Pareto são mostrados na Figura 3.5. Tem-se um vetor de pesos $\mathbf{w} = (w_1, w_2)$ para cada objetivo.

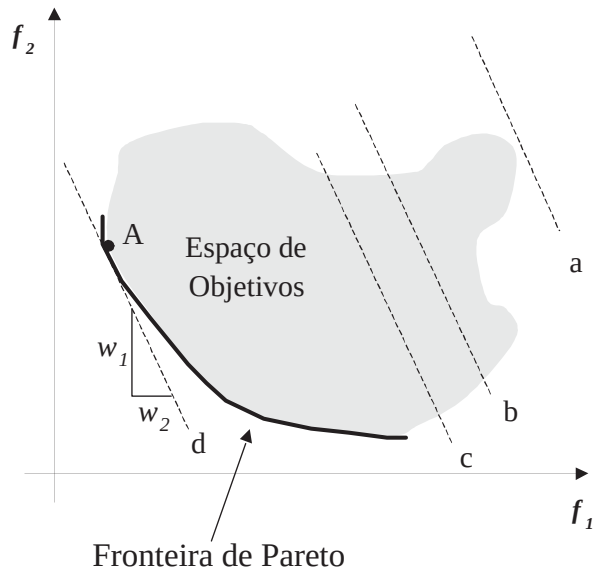



**Figura 3.5:** O Método do Somatório de Pesos

Dado um vetor de pesos $\mathbf{w}$ é possível plotar o contorno de $F$ no espaço de objetivos. Dado que $F$ é uma combinação linear dos objetivos, obtém-se uma linha reta. Cada linha de contorno possui o menor valor para $F$. Encontrar o mínimo valor da equação 3.2 é equivalente a achar uma linha de contorno com um valor mínimo para $F$.

A Figura 3.5 mostra várias linhas de contorno para $F$, sendo que a linha $d$ é *tangencial* a um ponto do espaço de objetivos ($A$). Esse ponto se encontra na Fronteira de Pareto e conseqüentemente é uma solução Pareto-ótima. Modificando os valores para $w_1$ e $w_2$ encontra-se uma outra solução Pareto-ótima. A seguir são mostrados detalhes do procedimento, usando um exemplo.

### Exemplo 1: MOOP convexo

Dado o seguinte MOOP :

$$\left.\begin{array}{rl} \text{minimizar} & f_1(\mathbf{x}) = x_1 \\ & f_2(\mathbf{x}) = 1 + x_2^2 - x_1 - asen(b\pi x_1), \\ \text{restrita a} & 0 \leq x_1 \leq 1, -2 \leq x_2 \leq 2. \end{array}\right\} \tag{3.3}$$

Os parâmetros $a$ e $b$ controlam a convexidade do espaço de busca. A Figura 3.6 mostra a Fronteira de Pareto e a sua vizinhança. Os parâmetros usados são $a = 0.2$, $b = 1$, e o peso $w_1$ foi uniformemente distribuído no intervalo $[0.271, 0.620]$

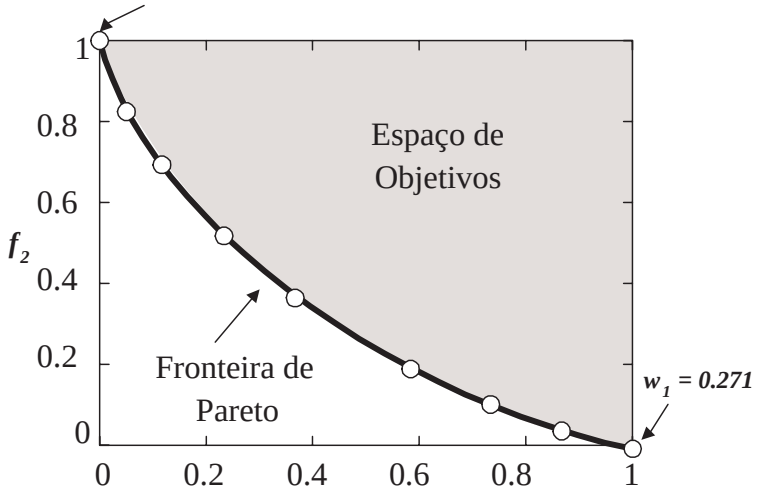



**Figura 3.6:** Soluções para $F$ ao longo da Fronteira de Pareto para $F$.

A função composta $F$ é dada por:

$$F(\mathbf{x}) = w_1 x_1 + w_2[1 + x_2^2 - x_1 - asen(b\pi x_1)]$$

Usando a primeira condição de otimalidade (ver Apêndice A) tem-se que:

$$\frac{\partial F}{\partial x_1} = w_1 + w_2\left[-1 - ab\pi cos\left(b\pi x_1\right)\right], \tag{3.4}$$

$$\frac{\partial F}{\partial x_2} = 2w_2x_2. \tag{3.5}$$

Igualando cada fórmula a 0, obtém-se:

$$x_1^* = \frac{1}{b\pi}\arccos\left[\frac{1}{ab\pi}\left(\frac{w_1}{w_2}-1\right)\right], \tag{3.6}$$

$$x_2^* = 0 \tag{3.7}$$

Logo, para satisfazer as segunda condição de otimalidade (Apêndice A):

$$sen(b\pi x_1^*) \geq 0 \tag{3.8}$$

A Figura 3.7 mostra os valores do $x_1^*$ que satisfazem a condição 3.8. Existem o vários intervalos para os quais a condição é satisfeita, dados pelas regiões $2i/b \leq x_1^* \leq (2i+1)/b$, para todo $i = 0, 1, 2, \ldots$ e os limites para $x_1^*$ são 0 e 1, então apenas quando $i = 0$ a solução é factível. Substituindo $a = 0.2$ e $b = 1$ na Equação 3.6 resulta em:

$$x_1^* = \frac{1}{\pi}\arccos\left[\frac{5}{\pi}\left(\frac{w_1}{w_2}-1\right)\right], \quad x_2^* = 0$$

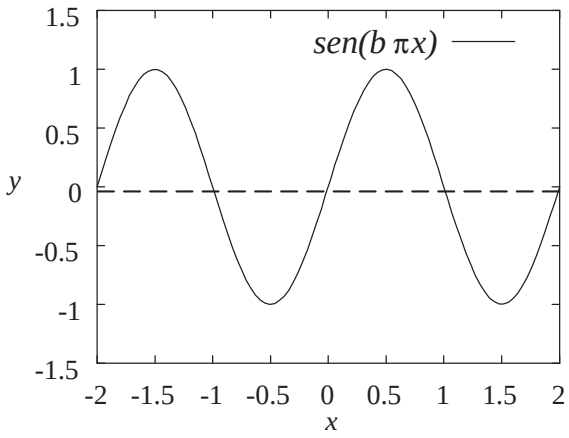



**Figura 3.7:** Gráfico para a desigualdade da equação 3.8

Esta última equação permite calcular os valores de $x_1^*$ é $x_2^*$ para usando diferentes pesos $w_1$ e $w_2$. Dado que a função arco cosseno está definida no intervalo [-1, 1], $w_1$ pode ser calculada para $0.271 \leq w_1 \leq 0.620$. Algumas soluções obtidas são:

$$x_1^* = 0: \quad w_1 = 0.620, \quad w_2 = 0.380$$
$$x_1^* = 1: \quad w_1 = 0.271, \quad w_2 = 0.729$$

Para encontrar uma boa quantidade de soluções Pareto-ótimas é necessário aplicar várias vezes este procedimento com diferentes valores para $w_1$ e $w_2$.

### Exemplo 2: MOOP não convexo

Para um MOOP não convexo, o método da soma dos pesos não encontrará algumas soluções Pareto-ótimas. A Figura 3.8 mostra o problema Exemplo 1 com os parâmetros $a = 0.1$ e $b = 3$. A Fronteira de Pareto está dividida em 4 regiões ($AB$, $BC$, e $CD$). Para qualquer reta tangente a um ponto na região $BC$, existirá uma outra reta tangente às regiões $AB$ ou $CD$ com um valor menor para $F$.

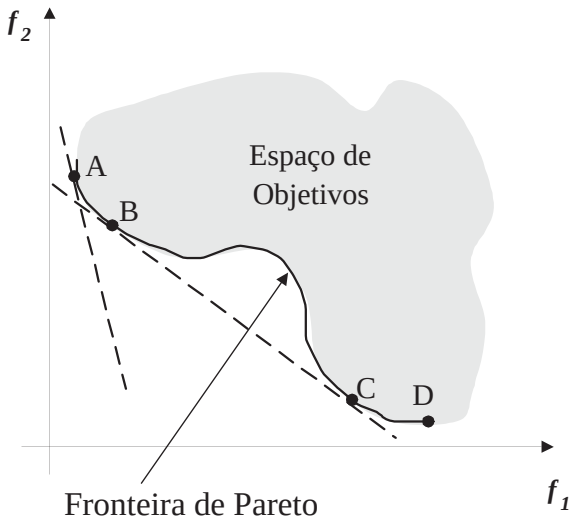



**Figura 3.8:** Fronteira de Pareto não convexa

Conforme a equação 3.8 o intervalo para $x_1^*$ possui agora duas regiões, $0 \leq x_1^* \leq 1/3$ e $2/3 \leq x_1^* \leq 1$. Isso significa que em cada região existe um mínimo. Substituindo os valores de $a$ e $b$ na equação 3.6, e considerando os dois intervalos para $x_1^*$ tem-se:

$$\begin{aligned} x_1^* &= \frac{1}{3\pi} \arccos\left[\frac{1}{0.3\pi}\left(\frac{w_1}{w_2} - 1\right)\right] \qquad x_2^* = 0 \\ x_1^* &= \frac{2}{3} + \frac{1}{3\pi} \arccos\left[\frac{1}{0.3\pi}\left(\frac{w_1}{w_2} - 1\right)\right] \qquad x_2^* = 0 \end{aligned} \tag{3.9}$$

O intervalo de $w_1$ está dado para $[0.054, 0.660]$. Para cada $w_1$ neste intervalo, calcula-se os valores para $x_1^*$ conforme à equação 3.9. Será escolhido o valor de $x_1^*$ tal que $F(x_1^*, x_2^*)$ seja menor o menor valor de $F$ . Os valores $0.054 \leq w_1 \leq 0.500$ encontram soluções na região $CD$, enquanto para $0.500 \leq w_1 \leq 0.660$ são achadas soluções em $AB$. Quando $w_1 = 0.500$

a reta é tangente aos pontos $B$ e $C$, portanto as soluções $B$ e $C$ são encontradas, e nenhum valor de $w_1$ permite achar soluções na região $BC$.

Uma conclusão é que este método, embora seja simples, precisa de várias iterações para atingir toda a Fronteira de Pareto. No caso de um MOOP não convexo, este método não é capaz de achar todas as soluções. A aplicação de vetores de pesos uniformemente distribuídos não garante achar um conjunto de soluções uniformemente distribuídas.

### 3.3.2 Método de restrições $\epsilon$ ($\epsilon$-Constraint)

Haimes e colaboradores [Haimes et al., 1971] apud [Deb, 2001], sugeriram reformular um MOOP considerando qualquer objetivo, e mantendo restritos os demais objetivos com valores definidos pelo usuário. A formulação é a seguinte:

$$\left.\begin{array}{lll} \text{minimizar} & f_u(\mathbf{x}), & \\ \text{restrita a} & f_m(\mathbf{x}) \leq \epsilon_m, & m = 1, 2, \ldots, M \text{ e } m \neq u; \\ & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N. \end{array}\right\} \tag{3.10}$$

Onde cada $\epsilon_m$ definida pelo usuário representa um limite máximo para o valor de $f_m$. Seja um MOOP não convexo de dois objetivos $f_1$ e $f_2$. Escolhe-se $f_2$ e mantém-se $f_1$ com a restrição $f_1 \leq \epsilon_1$.

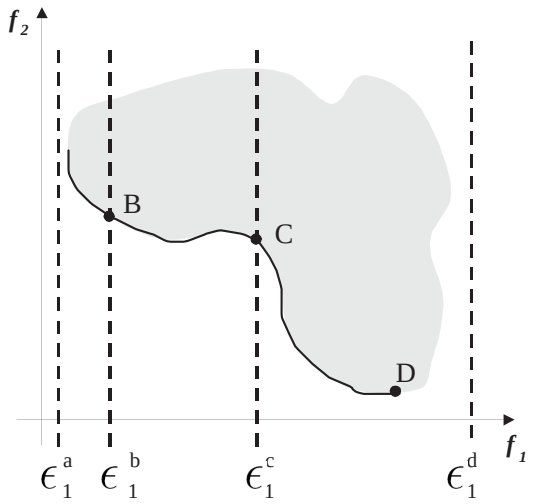



**Figura 3.9:** Método de $\epsilon$-Constrain.

A Figura 3.9 apresenta o espaço de objetivos e vários valores para $\epsilon_1$. O mínimo para $f_2$ depende da escolha do $\epsilon$. Por exemplo usando $\epsilon_1^c$, o valor mínimo para $f_2$ é ponto C. Então

iterando valores diferentes de $\epsilon$ achamos diferentes soluções Pareto-ótimas. Este método resolve o problema do método do Somatório dos Pesos para MOOP não convexos. Os resultados da presente técnica estão garantidos pelo seguinte teorema.

**Teorema 3** *A solução para o problema formulado na Equação 3.10 é Pareto-ótima para qualquer vetor* $\epsilon = (\epsilon_1, \ldots, \epsilon_{u-1}, \epsilon_{u+1}, \ldots, \epsilon_M)$.

Seja o problema do Exemplo 2 com parâmetros $a = 0.2$ e $b = 3$. Escolhe-se o objetivo $f_2$, e mantém-se $f_1$ como uma restrição da forma $f_1(\mathbf{x}) \leq \epsilon_1$, tem-se que:

$$\left.\begin{array}{rl} \text{Minimizar} & f_2(\mathbf{x}) = 1 + x_2^2 - x_1 - 0.2sen(3\pi x_1), \\ \text{restrita a} & f_1(\mathbf{x}) = x_1 \leq \epsilon_1 \\ & 0 \leq x_1 \leq 1, -2 \leq x_2 \leq 2. \end{array}\right\} \tag{3.11}$$

Seja $g_1 \equiv \epsilon_1 - x_1 \geq 0$, conforme as condições de otimalidade de Kuhn-Tucker (veja Apêndice A), dado um multiplicador o vetor $u_1$ para $g_1$, tem-se:

$$\begin{aligned} \nabla f_2 - u_1 \nabla g_1 &= 0; \\ g_1 &\geq 0, \\ u_1 g_1 &= 0, \\ u_1 &\geq 0 \end{aligned}$$

Substituindo:

$$\begin{aligned} -1 - 0.6\pi \cos(3\pi\ x_1) + u_1 &= 0; \\ 2x_2 &= 0, \\ \epsilon_1 - x_1 &\geq 0, \\ u_1(\epsilon_1 - x_1) &= 0, \\ u_1 &\geq 0 \end{aligned} \tag{3.12}$$

A Figura 3.10 mostra a Fronteira de Pareto para o problema formulado na Equação 3.11. Esta apresenta duas regiões descontínuas ($AB$ e $DE$).

Pela primeira igualdade no conjunto de equações 3.12, se $u_1 = 0$, tem-se que $cos(3\pi x_1^*) = -1/0.6\pi$. Os três valores para $x_1^*$ são $0.226, 0.441, 0.893$ correspondentes aos pontos $B$, $C$, e $E$. O ponto $C$ é um máximo em $f_2$, então é descartado. Dentro do intervalo para $\epsilon_1 \in [0, 1]$ tem-se os seguintes valores de $x_1^*$ ($x_2^* = 0$) :

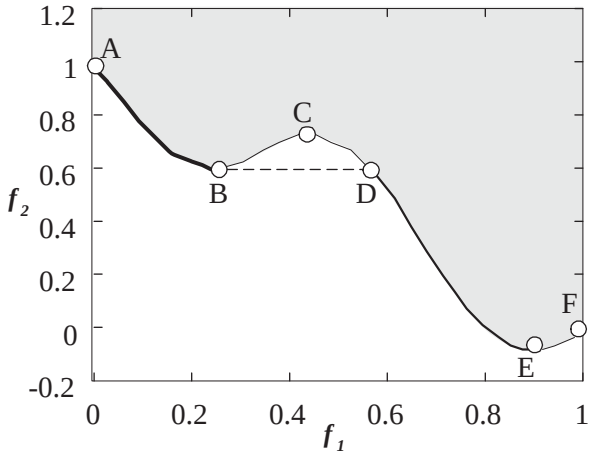



**Figura 3.10:** Fronteira de Pareto descontínua do problema 3.11.

$$x_1^* = \begin{cases} \epsilon_1, & \text{se } 0 \leq \epsilon_1 \leq 0.226 \text{ (região } AB); \\ 0.226, & \text{se } 0.226 < \epsilon_1 < 0.441 \ (BC); \\ 0.226, & \text{se } f_2(\epsilon_1, 0) > f_2(0.226, 0) \text{ e } 0.441 \leq \epsilon_1 \leq 0.893 \ (CD); \\ \epsilon_1, & \text{se } f_2(\epsilon_1, 0) \leq f_2(0.226, 0) \text{ e } 0.441 \leq \epsilon_1 \leq 0.893 \ (DE); \\ 0.893, & \text{se } 0.893 < \epsilon_1 < 1 \ (EF); \end{cases}$$

Desta forma, o método de restrições $\epsilon$ pode ser usado para gerar as soluções Pareto-ótimas independentemente se o espaço de objetivos for convexo, não convexo ou discreto. Este método depende também da escolha do vetor $\epsilon$ que esteja em uma região factível para cada objetivo. Por exemplo na Figura 3.9 se for escolhido $\epsilon_1^a$, então nenhuma solução será achada.

### 3.3.3 Programação de Metas (Goal Programming)

Esta técnica tenta achar soluções que possam atingir uma meta predeterminada para uma ou mais funções objetivo. Caso não exista uma solução factível que alcance as metas para todos os objetivos, esta minimiza os *desvios* em relação às metas.

Considere uma função $f(\mathbf{x})$ para ser minimizada dentro do espaço de busca $S$. Para cada objetivo é escolhido um valor meta $t$ pelo usuário. Então o problema é formulado para encontrar uma solução cujo valor em $f$ seja igual a $t$. Formalmente:

$$\begin{aligned} meta \quad & (f(\mathbf{x}) = t), \\ & \mathbf{x} \in S \end{aligned}$$

Onde $S$ é os espaço de busca. Seja $f(\mathbf{x}^*)$ o valor mínimo para $f$. Conforme o valor $t$, podem acontecer o seguintes casos:

1. O valor de $t$ é menor que $f(\mathbf{x}^*)$. Portanto não existe uma solução que alcance $t$ dentro do espaço factível. Então, a solução ao problema seria $f(\mathbf{x}^*)$ já que possui a menor distância em relação a $t$.

2. O valor de $t$ é maior que o máximo para $f$ ($f_{max}$). A solução seria $f_{max}$ porque minimiza a distância em relação a $t$.

3. A meta $t$ está no intervalo $[f(\mathbf{x}^*), f_{max}]$. A solução seria um $\mathbf{x}$ tal que $f(\mathbf{x}) = t$, onde o desvio é zero.

Existem quatro tipos de critérios para atingir uma meta,

1. $f(\mathbf{x}) \leq t$, onde o objetivo é minimizar o desvio $p$ para que $f(\mathbf{x}) - p \leq t$. O desvio $p$ representa a quantidade pela qual $f$ supera a $t$, caso $f(\mathbf{x}) > t$. Caso contrário $p$ é zero.

2. $f(\mathbf{x}) \geq t$, onde o objetivo é minimizar o desvio $n$ para que $f(\mathbf{x}) + n \geq t$. O desvio $n$ representa a quantidade para $f$ alcance $t$, caso $f(\mathbf{x}) < t$. Caso contrário $n$ é zero.

3. $f(\mathbf{x}) = t$, onde o objetivo é minimizar a soma dos desvios $p$ e $n$ para que $f(\mathbf{x}) - p + n = t$. Se $f(\mathbf{x}) > t$, $p$ é positivo e $n$ é zero. Se $f(\mathbf{x}) < t$, $n$ é positivo e $p$ é zero. Ambos desvios $p$ é $n$ são zero caso $f(\mathbf{x}) = t$

4. $f(\mathbf{x}) \in [t^l, t^u]$. Para este caso, as metas $t^l$ e $t^u$ são tratadas usando as restrições $f(\mathbf{x}) - p \leq t^l$ e $f(\mathbf{x}) + n \geq t^u$. A meta é minimizar a soma dos desvios $p$ e $n$.

Os casos acima são resumidos em uma restrição genérica :

$$f(\mathbf{x}) - p + n = t$$

Para resolver um problema de programação de metas, cada meta é convertida em uma restrição de igualdade, e procura-se minimizar todos os desvios. Existem várias formas de trabalhar com estes problemas as quais serão descritas a seguir:

### Programação de Metas com Pesos

Para um problema com $M$ objetivos, formula-se uma função somando os desvios para cada um dos $M$ objetivos. A forma geral é:

$$\left.\begin{array}{ll} \text{minimizar} & \sum_{j=1}^{M}(\alpha_j p_j + \beta_j n_j) \\ \text{restrita a} & f_j(\mathbf{x}) - p_j + n_j = t_j, \quad j = 1, 2, \ldots, M \\ & \mathbf{x} \in S, \\ & n_j, p_j \geq 0, \qquad\qquad j = 1, 2, \ldots, M. \end{array}\right\} \tag{3.13}$$

Onde $\alpha_j$ e $\beta_j$ são os pesos dos desvios $p$ e $n$ para o j-ésimo objetivo. $S$ é o espaço de decisão factível. As soluções achadas por este método dependem da escolha dos valores $\alpha_j$ e $\beta_j$. Este método compartilha as mesmas dificuldades que o Método do Somatório dos Pesos [Deb, 2001].

### Programação de Metas Lexicográficas

Neste método, as metas são organizadas em vários níveis de prioridade. Resolve-se seqüencialmente vários problemas de programação de metas. As metas de primeiro ordem de prioridade são consideradas na formulação do problema, e logo este problema é resolvido. Caso existam múltiplas soluções, as metas de segundo ordem de prioridade são consideradas, e formula-se outro problema para minimizar apenas os desvios para as metas de segundo ordem. As metas de primeiro ordem de prioridade são usadas como restrições. O processo continua com os demais níveis de prioridade até que seja achada uma única solução. Mediante este método, é achada freqüentemente uma solução Pareto-ótima.

A Figura 3.11 mostra um espaço de objetivo para as funções $f_1$ e $f_2$. Se $f_1$ é mais importante, minimiza-se $f_1$ primeiro e se obtém as soluções das regiões $AB$ e $CD$ nas quais $f_1$ é mínima. Dado que existem multiples soluções, minimiza-se $f_2$ somente nas regiões $AB$ e $CD$ encontradas na iteração anterior. A solução é o ponto $D$ sendo o mínimo para $f_2$. Então, $D$ é a solução para todo o problema de programação de metas Lexicográficas.

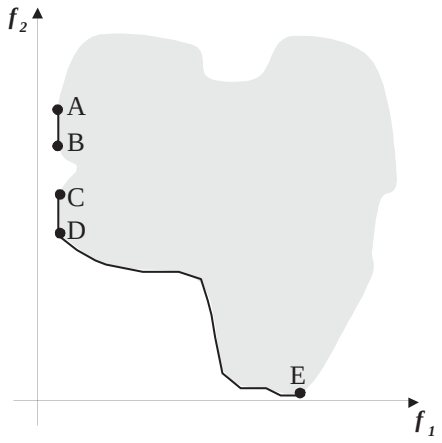



**Figura 3.11:** Método da Programação de Metas Lexicográfica

**Programação de Metas Min-Max**

Neste método é minimizado o máximo desvio em relação às metas. A formulação é a seguinte.

$$\left.\begin{array}{lll} \text{Minimizar} & d & \\ \text{restrita a} & \alpha_j p_j + \beta_j n_j \leq d, & j = 1, 2, \ldots, M \\ & f_j(\mathbf{x}) - p_j + n_j = t_j, & j = 1, 2, \ldots, M \\ & \mathbf{x} \in S, & \\ & n_j, p_j \geq 0, & j = 1, 2, \ldots, M. \end{array}\right\} \tag{3.14}$$

Onde $d$ é o maximo desvio para qualquer meta, $p_j$ e $n_j$ são as desvios para cada objetivo e $\alpha_j$ e $\beta_j$ representam os pesos para cada desvio. Este método precisa a escolha dos pesos $\alpha_j$ e $\beta_j$.

### 3.3.4 Vantagens e Desvantagens das Técnicas Tradicionais

A principal vantagem das técnicas tradicionais é que estas possuem provas de convergência que garantem encontrar as soluções Pareto-ótimas.

Todas as técnicas descritas neste capítulo fazem a conversão de um MOOP para um problema de objetivo simples. Cada técnica usa uma forma diferente de conversão e introduz parâmetros adicionais. A escolha destes parâmetros afeta diretamente os resultados obtidos. Cada vez que os parâmetros são modificados, tem-se que resolver um novo problema de otimização simples. Portanto, para encontrar $N$ soluções Pareto-ótimas, tem-se que solucionar no mínimo $N$ problemas de objetivos simples.

Alguns métodos não garantem achar soluções ao longo de toda a Fronteira de Pareto. Se a Fronteira de Pareto não é convexa, o método do Somatório dos Pesos não encontra certas soluções independentemente dos pesos escolhidos.

Finalmente, todos as técnicas descritas precisam de parâmetros adicionais tais como pesos, metas, e vetores de restrição. A distribuição uniforme destes parâmetros não garante a diversidade das soluções Pareto-ótimas.

## 3.4 Comentários Finais

Neste capítulo foram introduzidas noções básicas de otimização Multi-objetivo. Apresentou-se o modelo geral para um MOOP e as principais diferenças à respeito do problema de otimização de objetivos simples. Finalmente, foram descritas as técnicas tradicionais para resolver um MOOP, verificando as vantagens e desvantagens de cada uma. No próximo

capítulo serão apresentados os Algoritmos Evolutivos, enfatizando na sua utilidade para encontrar um conjunto de soluções Pareto-ótimas simultaneamente.

Capítulo

4

# Algoritmos Evolutivos para Otimização Multi-Objetivo

Durante os últimos anos, tem-se observado uma expansão na utilização de Algoritmos Evolutivos (AEs) [Deb, 2001], especialmente os Algoritmos Genéticos (AGs), em problemas de otimização. Uma das características mais importantes dos AEs é que possibilitam encontrar soluções ótimas ou adequadas para um problema sem usar informação extra, como cálculo de derivadas de funções [Goldberg, 1989].

Este capítulo enfoca a aplicação dos AGs nos problemas de Otimização Multi-Objetivo, descrevendo os diferentes modelos existentes. A primeira seção é dedicada aos conceitos gerais sobre AGs. A segunda seção descreve os principais modelos de AGs para MOOPs. Na terceira seção detalha-se os métodos para comparar a performance dos diferentes modelos de Algoritmos Evolutivos para Otimização Multi-Objetivo (MOEA). Finalmente, na quarta seção são apresentadas as conclusões.

## 4.1 Algoritmos Genéticos

Esta seção tem como objetivo apresentar os AGs como técnica de otimização. Os tópicos tratados neste ponto abordam os conceitos básicos para entender o funcionamento dos AGs, e as principais vantagens dos AGs em relação às técnicas tradicionais de otimização.

### 4.1.1 Definição

Os Algoritmos Genéticos são técnicas de busca inspiradas em mecanismos de seleção e genética natural [Goldberg, 1989]. Dada uma população inicial de soluções, esta evolui até convergir para uma solução, por meio da aplicação de operadores genéticos de seleção, cruzamento e mutação.

Considerando um problema de otimização qualquer, os AGs inicializam a busca da melhor solução a partir de um conjunto inicial de soluções aleatórias. Cada elemento do conjunto inicial de soluções é denominado indivíduo ou cromossomo. Um indivíduo pode ser representado por uma cadeia de símbolos, por exemplo, uma cadeia binária. É importante que cada indivíduo da população seja capaz de representar completamente uma possível solução do problema tratado.

Em seguida, uma nova população ou geração é gerada a partir da população inicial. Para criar os indivíduos da nova população, são utilizados operadores genéticos de *cruzamento e mutação*. Uma população é obtida a partir da anterior, aplicando-se o cruzamento para aqueles indivíduos com um maior valor de *aptidão*, simulando o processo de seleção natural. O valor de aptidão é calculado para cada indivíduo mediante uma função chamada função de aptidão ou função objetivo.

O processo de geração de novas populações é repetido iterativamente até que o AG chegue a uma solução aceitável, ou satisfaça alguma condição de parada. A representação do AG na forma de fluxograma está ilustrada na Figura 4.1.

### 4.1.2 Representação das Soluções

Cada indivíduo de uma população, gerada pelo AG, representa uma possível solução do problema que se deseja resolver. Assim, as variáveis da função objetivo devem ser representadas por um indivíduo. Por exemplo, considere a seguinte função objetivo:

$$f(x_1, x_2) = 20 + x_1 sen(4\pi x_1) + x_2 sen(20\pi x_2), \tag{4.1}$$

Todos os indivíduos de qualquer população devem representar as variáveis $x_1$ e $x_2$. Uma possível representação para as variáveis seria uma cadeia de bits de tamanho $n$ que representa tanto $x_1$ quanto $x_2$. Outra possibilidade seria um vetor de valores contínuos ($x_1$,$x_2$). Várias representações como matrizes e árvores também são possíveis. Escolher uma representação adequada para a solução de um problema é uma area de estudo fundamental para AGs [Gen e Cheng, 1997].

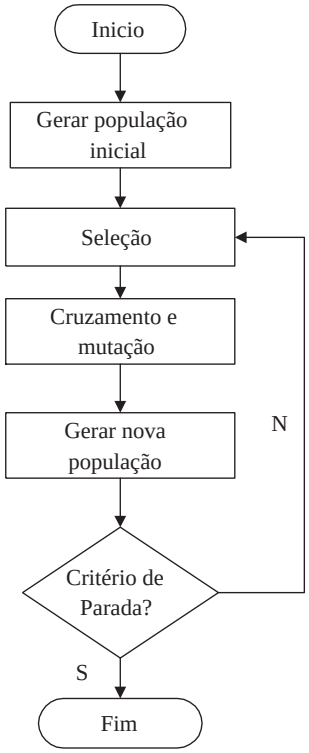



**Figura 4.1:** Fluxograma de um Algoritmo Genético.

### Representação Binária

Uma das representações mais utilizadas nos AGs é a representação binária, ideal para problemas onde as variáveis são discretas. Porém, se as variáveis são contínuas, uma conversão é necessária [Haupt e Haupt, 1998].

Como exemplo, considere um vetor de variáveis contínuas no intervalo [0,1] que deve ser representado por cadeias binárias de tamanho 3. A Tabela 4.1 ilustra uma representação binária de valores contínuos. De acordo com esta tabela, as variáveis [0.55, 0.11, 0.95, 0.63] serão representados pela cadeia 100—000—111—101.

Apesar dos AGs trabalharem com parâmetros binários, a função de aptidão necessita de valores contínuos. A mesma tabela fornece a conversão oposta de parâmetros binários para contínuos. Os números à esquerda e a direita representam os limites superior e inferior para cada trio binário. Por exemplo para 111 os limites são [0.9375, 1.000]. Assim, o mesmo indivíduo, 100000111101, será recuperado como [0.500, 0.000, 0.875, 0.625], [0.625, 0.125, 1.000, 0.750] ou [0.5625, 0.0625, 0.9375, 0.6875] quando se considera os limites inferior, superior ou a média de ambos os limites. Nota-se que a conversão adiciona uma margem de erro em cada variável.

| | | Valor da variável | | | | |
|---|---|---|---|---|---|---|
| | | 0.55 | 0.11 | 0.95 | 0.63 | |
| 1.000 | | | | | | |
| | 111 | | | • | | 0.9375 |
| 0.875 | | | | | | |
| | 110 | | | | | 0.8125 |
| 0.750 | | | | | | |
| | 101 | | | | • | 0.6875 |
| 0.625 | | | | | | |
| | 100 | • | | | | 0.5625 |
| 0.500 | | | | | | |
| | 011 | | | | | 0.4375 |
| 0.375 | | | | | | |
| | 010 | | | | | 0.3125 |
| 0.250 | | | | | | |
| | 001 | | | | | 0.1875 |
| 0.125 | | | | | | |
| | 000 | | • | | | 0.0625 |
| 0.000 | | | | | | |

**Tabela 4.1:** Tabela de conversão de parâmetros contínuos para binário.

#### Representação Contínua

Para casos onde o erro ocasionado pela representação binária é crítico, os AGs podem utilizar diretamente os valores contínuos. Nesse caso, nenhuma conversão é necessária. No entanto deve-se implementar operadores genéticos de cruzamento e mutação adequados à representação.

### 4.1.3 Operadores Genéticos

Esta seção aborda os principais operadores genéticos e suas variações mais utilizadas.

#### Seleção para Reprodução

Seleção é o processo onde os indivíduos com melhor valor de aptidão têm a maior probabilidade de gerar um ou mais descendentes para a geração seguinte. Este operador é a versão artificial da seleção das espécies do Darwinismo, que estabelece que os seres mais aptos têm maiores chances de sobreviver, ou seja, os mais fortes e menos vulneráveis aos predadores e doenças.

O objetivo principal do operador de seleção é copiar boas soluções, eliminando soluções de baixa aptidão, enquanto o tamanho da população é constante [Deb, 2001]. Isto é realizado seguindo os seguintes passos:

1. Identificar boas soluções na população;
2. Realizar múltiplas cópias das boas soluções;
3. Eliminar soluções de baixa aptidão da população o que permite que várias copias de boas soluções possam ser inseridas na população.

As melhores soluções são guardadas em uma *lista de soluções escolhidas* ou *lista de soluções*, que será utilizada para realizar as operações de cruzamento e mutação.

Existe um grande número de estratégias de seleção. As mais comuns são seleção pelo torneio, seleção proporcional, e seleção por ranking.

Na seleção por torneio, são realizados várias competições entre duas soluções, e a melhor solução é copiada na lista de soluções. Este processo é repetido até preencher a lista. Goldberg e Deb mostraram que este método possui uma convergência igual ou melhor que outras estratégias de seleção, além de possuir uma complexidade computacional menor [Deb, 2001].

Na estratégia de seleção proporcional, o número de cópias de uma solução na lista de soluções escolhidas é proporcional ao seu valor de aptidão. Para calcular o número de cópias esperado, é necessário obter a probabilidade de cada solução :

$$p_i = \frac{F_i}{\sum_{i=1}^{N} F_i} \tag{4.2}$$

sendo que $F_i$ é a aptidão da solução $i$. $N$ é o tamanho da população. O número de cópias na lista de soluções é calculado por $C_i = p_i N$. Ou seja, as soluções com melhor valor de aptidão terão mais copias na lista de soluções. O escalonamento é um problema associado com esta estratégia. Quando existe uma solução com um valor de aptidão muito maior comparado com o resto da população, esta super-solução terá uma probabilidade de escolha perto de 1, e terá muitas cópias na lista de soluções. Caso todas as soluções possuam valores similares de aptidão, terão a mesma probabilidade de serem escolhidas, e cada uma será copiada na lista de soluções. Isto é equivalente a não realizar operação de seleção.

A estratégia de seleção pelo *ranking* ordena as soluções população conforme ao seu valor de aptidão, desde a pior solução (*ranking* 1) até a melhor (*ranking* $N$). Depois, o número de copias das soluções é proporcional ao valor de *ranking*.

### Cruzamento

Este operador gera novas soluções (filhas) a partir de soluções escolhidas da lista de soluções (pais). O operador de cruzamento possui diferentes variações, muitas delas específicas a um determinado problema. A forma mais simples de cruzamento é conhecida como *cruzamento em um ponto*, ilustrada na Figura 4.2. Este cruzamento consiste em:

1. Escolher arbitrariamente dois indivíduos na lista de soluções;
2. Escolher dentro da cadeia do indivíduo uma posição $k$ chamada *posição de cruzamento*;
3. Criar novos descendentes trocando as cadeias parciais de cada um dos indivíduos.

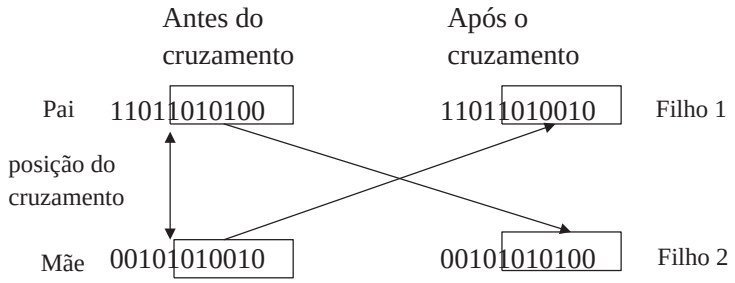



**Figura 4.2:** Operador de Cruzamento em um ponto.

É importante notar que o cruzamento possui uma *probabilidade associada*, geralmente um valor próximo de 1.

Apesar das operações de seleção e cruzamento parecerem muito simples, é nelas que se concentra o poder dos AGs [Goldberg, 1989] .

### Mutação

A mutação é um operador que produz uma alteração aleatória em uma posição de um pequeno número de indivíduos. A mutação é a segunda maneira dos AGs explorem o espaço de busca. Esta pequena alteração impede que o algoritmo tenha convergência muito rápida, evitando sua estabilização em regiões de mínimos locais.

### Elitismo

O operador de elitismo mantém as melhores soluções encontradas previamente nas gerações seguintes. Isto implica que a melhor solução não se deteriora nas gerações seguintes.

Uma forma de implementar elitismo é copiar diretamente as $n\%$ das soluções da população atual à população seguinte. O resto das $(100-n)\%$ das soluções é gerada usando os operadores genéticos usuais sobre a população atual. Desta forma, as melhores soluções passam diretamente para população seguinte, além de participar da criação do resto de soluções da população seguinte.

Outra forma de elitismo consiste em criar a população seguinte a partir da população atual usando os operadores genéticos usuais, e escolher as melhores $N$ soluções de ambas populações.

### 4.1.4 Passos para Implementar um Algoritmo Genético

Para resolver um determinado problema utilizando AGs, os seguintes passos devem ser seguidos:

1. Definir uma representação a ser utilizada para indivíduo de maneira que uma solução completa possa ser representada por ele;
2. Definir as estratégias de substituição, seleção, cruzamento e mutação;
3. Definir a função de aptidão;
4. Ajustar os seguintes parâmetros: tamanho da população, probabilidade de cruzamento, probabilidade da mutação e número de gerações.

Geralmente, estes últimos parâmetros variam de acordo com o problema e são ajustados manualmente.

### 4.1.5 Exemplo do Uso de um Algoritmo Genético

Para exemplificar o funcionamento dos AGs, considere encontrar o máximo da função:

$$f(x, y) = 21.5 + xsen(4\pi x) + ysen(20\pi y) \tag{4.3}$$

Supondo que a solução que se busca tem coordenadas $x$ e $y$. Estas coordenadas representam o ponto de máximo desta função. A superfície gerada por esta função possui vários picos e vales, caracterizando assim um problema com possibilidades de convergência para pontos de máximos locais. A Figura 4.3 ilustra uma projeção suavizada desta superfície.

| **Solução** | $x$ | $y$ | **Aptidão** |
|---|---|---|---|
| 1 | 8.55696 | 4.84176 | 29.51089 |
| 2 | -2.14217 | 5.44308 | 25.88302 |
| 3 | 1.96259 | 4.79317 | 18.61883 |
| 4 | -0.40211 | 4.26638 | 17.46505 |
| 5 | -0.53782 | 5.08256 | 17.22745 |
| 6 | 10.91246 | 5.41764 | 16.63308 |
| 7 | -1.36431 | 5.46409 | 15.91625 |
| 8 | 3.44358 | 4.88583 | 15.80166 |
| 9 | 1.88679 | 5.47073 | 14.35801 |
| 10 | 6.35054 | 5.38383 | 10.87223 |

**Tabela 4.2:** População inicial do AG.

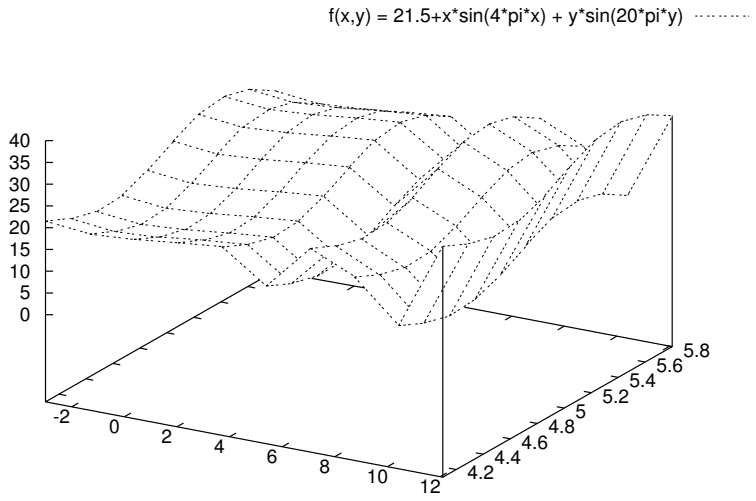



**Figura 4.3:** Superfície suavizada da função $f(x, y)$.

| **Solução** | $x$ | $y$ | **Aptidão** |
|---|---|---|---|
| 1 | 8.55305 | 5.42428 | 32.20755 |
| 2 | 8.55281 | 5.42428 | 32.18789 |
| 3 | 1.00293 | 5.43068 | 26.62459 |
| 4 | 8.55696 | 4.59307 | 25.17936 |
| 5 | 8.55305 | 5.68915 | 23.20583 |
| 6 | 8.55696 | 4.57343 | 22.56428 |
| 7 | -2.14217 | 4.26627 | 19.95104 |
| 8 | -1.78296 | 4.79338 | 18.84688 |
| 9 | -0.40211 | 5.96177 | 18.30942 |
| 10 | -0.40211 | 4.26607 | 17.50904 |

**Tabela 4.3:** Segunda geração do AG.

| **Solução** | $x$ | $y$ | **Aptidão** |
|---|---|---|---|
| 1 | 8.55305 | 5.42428 | 32.20755 |
| 2 | 8.55305 | 5.42428 | 32.20755 |
| 3 | 8.55305 | 5.42428 | 32.20755 |
| 4 | 8.55305 | 5.42428 | 32.20755 |
| 5 | 8.55305 | 5.42428 | 32.20755 |
| 6 | 8.55305 | 5.42407 | 32.20368 |
| 7 | 8.55305 | 5.42407 | 32.20368 |
| 8 | 8.52355 | 5.42428 | 29.40521 |
| 9 | 8.78899 | 5.42428 | 22.78260 |
| 10 | 8.55305 | 4.57426 | 22.21939 |

**Tabela 4.4:** Décima geração do AG.

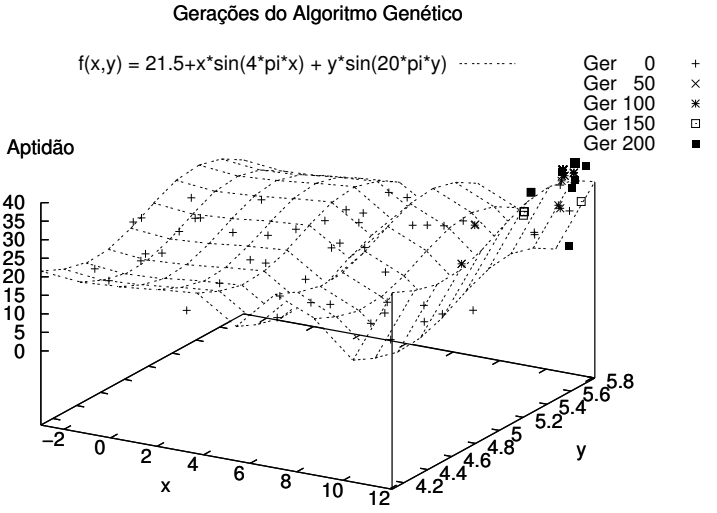



**Figura 4.4:** Indivíduos distribuídos sobre a superfície da função $f(x, y)$.

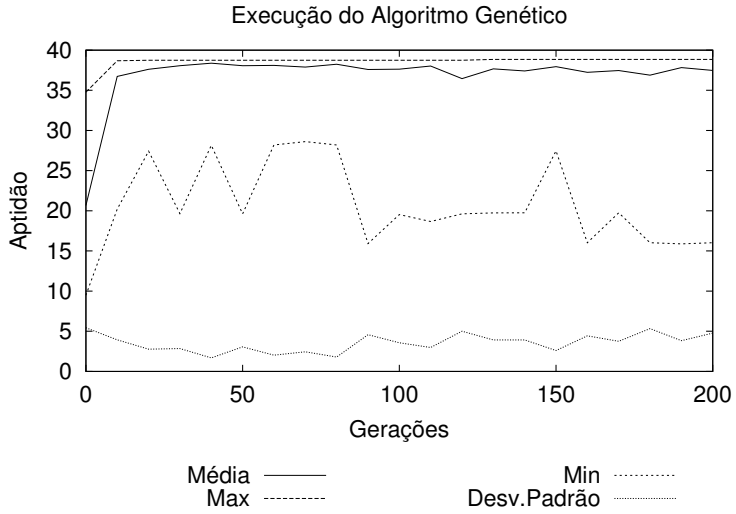



**Figura 4.5:** Gráfico da aptidão calculada em relação as gerações.

O AG começa gerando uma população inicial aleatória. Os indivíduos são ordenados de acordo com o seu valor de aptidão calculado pela Equação 4.3 e alguns entre os melhores são selecionados para o cruzamento. Os indivíduos da nova população também são ordenados. Após esta etapa, os operadores de seleção, cruzamento e mutação são repetidos até a convergência do algoritmo. As Tabelas 4.2, 4.3 e 4.4 apresentam a população inicial e as populações referentes à segunda e décima gerações, respectivamente .

A Figura 4.4 ilustra a distribuição dos indivíduos ao longo da superfície. As gerações 0 (Ger 0), 50 (Ger 50), 100 (Ger 100), 150 (Ger 150) e 200 (Ger 200) são ilustradas nesta figura. A Figura 4.5 apresenta um gráfico do desenvolvimento da função aptidão em relação às gerações, incluindo os valores máximos e mínimos, além da média e desvio padrão.

### 4.1.6 Diferenças entre os Algoritmos Genéticos e os Métodos de Otimização Tradicionais

Visto um exemplo do uso de AGs, pode-se apontar algumas diferenças em relação a técnicas tradicionais: [Goldberg, 1989]

- Os AGs trabalham com uma população de soluções em cada iteração ao invés de uma única solução. Em cada iteração os AGs processam um *conjunto de soluções*, esta característica é denominada de paralelismo implícito.
- Os AGs não precisam de informação adicional a não ser o valor de aptidão das soluções. Isto faz que os AGs sejam aplicáveis a uma grande variedade de problemas, alguns dos quais não se tem informações *a priori*.
- Os AGs empregam regras probabilísticas para guiar a sua busca. Por exemplo, o processo de seleção é baseado na aleatoriedade de duas soluções (seleção pelo torneio), ou na probabilidade de escolha (seleção proporcional) dessas soluções. O operador de mutação permite que os AGs não recaiam nos ótimos locais, mudando a busca para outra região do espaço. Além de que as soluções da população inicial são escolhidas arbitrariamente. Em contrapartida, uma técnica de regras fixas não terá como escapar de ótimos locais, em caso de uma má decisão sobre a direção da busca.

A possibilidade de se trabalhar com várias soluções simultaneamente, de não precisar de informações adicionais e poder escapar de ótimos locais fazem dos AGs uma técnica promissória a ser empregada nos MOOPs. A seguinte seção aborda este tema.

## 4.2 Algoritmos Genéticos para Otimização Multi-Objetivo

A primeira implementação de um MOEA foi proposta por Schaffer em 1985 [Schaffer, 1985] apud [Deb, 2001]. O modelo sugerido foi denominado VEGA (Vector Evaluated Genetic Algorithm). Schaffer fez uma modificação nos AGs para avaliar cada objetivo separadamente. Um dos problemas do VEGA, é que este algoritmo não obtém boa diversidade nas soluções da Fronteira de Pareto.

Goldberg [Goldberg, 1989] crio um procedimento para ordenação de soluções baseado no conceito de dominância. Este método fornece um valor de aptidão para uma solução $i$ proporcional ao número de soluções que $i$ domina. Desta forma, as soluções não dominadas são enfatizadas e terão maior quantidade de cópias na lista de soluções. Para manter a diversidade das soluções, Goldberg sugeriu o emprego de um método de compartilhamento

(apresentado na seção 4.2.1). Com base nas idéias inicias propostas por Goldberg foram propostas vários modelos de MOEAs.

A diferença fundamental dos MOEA em relação aos Algoritmos Evolutivos tradicionais é operador de seleção, dado que a comparação entre duas soluções deve realizar-se conforme o conceito de dominância de Pareto. Em alguns métodos, como MOGA e SPEA, o valor de aptidão é proporcional à dominância da solução. Outros métodos, como NPGA, utilizam apenas a dominância de Pareto e não o calculam um valor de aptidão.

Os modelos de MOEA são classificados por Deb [Deb, 2001] em dois tipos:

1. Não elitistas, são aqueles modelos que como indica o própio nome, não utilizam alguma forma de elitismo nas suas iterações.

2. Elitistas, modelos que usam de alguma forma o elitismo. Alguns modelos como SPEA, PESA, utilizam uma população externa onde são guardadas as soluções não dominadas encontradas até o momento. Outros métodos como NSGA-II combinam a população atual com a população posterior para preservar as melhores soluções de ambas. O estudo realizado por Zitzler [Zitzler et al., 2000] conclui que o elitismo melhora as soluções encontradas por um modelo MOEA. A partir deste trabalho, os novos modelos incorporam alguma estratégia de elitismo.

A Tabela 4.5 lista os principais modelos de MOEAs, seus autores, e a sua classificação conforme Deb [Deb, 2001].

A seguir serão apresentadas quatro dos principais modelos MOEAs, focalizando suas vantagens, desvantagens e principais contribuições de cada um.

### 4.2.1 MOGA (Multi-Objective Genetic Algorithm)

O algoritmo MOGA [Fonseca e Fleming, 1993] foi o primeiro a dar ênfase ao conceito de dominância e à diversidade das soluções. [Deb, 2001].

MOGA se diferença dos AGs clássicos pela forma com que atribui o valor de aptidão as soluções de uma população. A cada solução é associado um valor de ranking que é igual ao número de soluções $n_i$ que a dominam mais um.

$$r_i = 1 + n_i \tag{4.4}$$

Assim, as soluções não dominadas possuem ranking 1. Pelo menos um indivíduo da população possui valor de $r_i = 1$, o valor máximo de $r_i$ não é maior do que o tamanho da população ($N$). A Figura 4.6a mostra um conjunto de soluções e seus valores $r_i$ na Figura

| Sigla | Nome do modelo | Autores |
|---|---|---|
| VEGA | Vector Evaluated Genetic Algorithm | [Schaffer, 1985] |
| WBGA | Weight Based Genetic Algorithm | [Hajela e Lin, 1992] |
| MOGA | Multiple Objective Genetic Algorithm | [Fonseca e Fleming, 1993] |
| NSGA | Non-Dominated Sorting Genetic Algorithm | [Srinivas e Deb, 1994] |
| NPGA | Niched-Pareto Genetic Algorithm | [Horn et al., 1994] |
| PPES | Predator-Prey Evolution Strategy | [Laumanns et al., 1998] |
| REMOEA | Rudolph's Elitist Multi-Objective Evolutionary Algorithm | [Rudolph, 2001] |
| NSGA-II | Elitist Non-Dominated Sorting Genetic Algorithm | [Deb et al., 2000] |
| SPEA, SPEA2 | Strenght Pareto Evolutionary Algorithm 1 e 2 | [Zitzler e Thiele, 1998, Zitzler et al., 2001] |
| TGA | Thermodynamical Genetic Algorithm | [Kita et al., 1996] |
| PAES | Pareto-Archived Evoltionary Strategy | [Knowles e Corne, 1999] |
| MOMGA-I, MOMGA-II | Multi-Objective Messy Genetic Algorithm | [Veldhuizen, 1999] |
| Micro-GA | Multi-Objective Micro-Genetic Algorithm | [Coello Coello e Toscano Pulido, 2001] |
| PESA-I, PESA-II | Pareto Envelope-Base Selection Algorithm | [Corne et al., 2000, 2001] |

**Tabela 4.5:** Diferentes modelos MOEAs

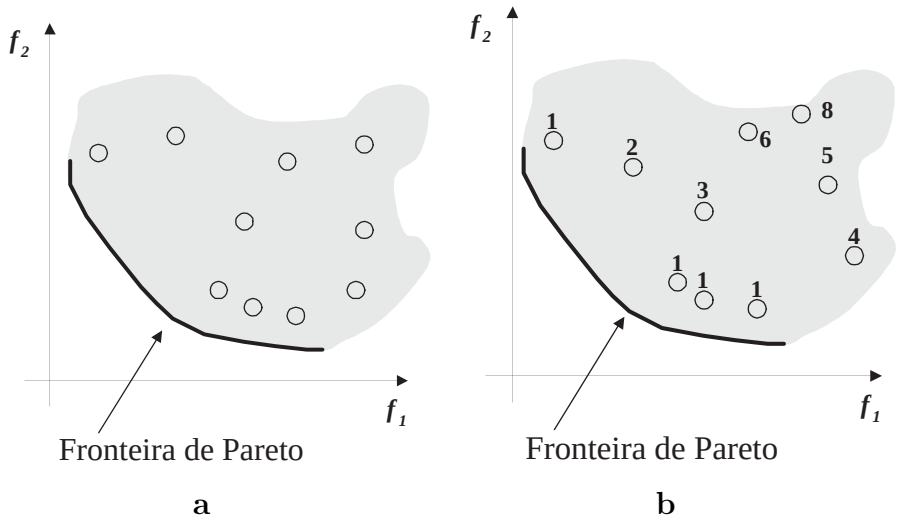



**Figura 4.6:** Cálculo do ranking do algoritmo MOGA [Deb, 2001]

.

4.6b. Pode-se observar que alguns valores de $r_i$ não são usados (7, 9, e 10). Associa-se um contador para $\mu(r_i)$ para cada valor de $r_i$.

A seguir, a população é ordenada conforme $r_i$, e se dá um valor de aptidão preliminar (raw fitness $raw_i$) para cada solução usando uma função linear ou outro tipo de função

[Fonseca e Fleming, 1993]. Posteriormente, o valor de aptidão média para as soluções no mesmo ranking é calculada. Isto permite que, soluções com melhor ranking tenham valores de aptidão mais altos. Desta forma, as soluções não dominadas são destacadas.

$$F_i = \frac{\sum raw_j}{\mu(r_i)}, \tag{4.5}$$

onde $\mu(r_i)$ é o número de soluções no ranking $r_i$. Para manter a diversidade entre soluções não dominadas, os autores propuseram usar nichos para cada ranking. Uma vez obtidos os nichos é calculada a aptidão compartilhada $F_i'$ (apresentada na seção 4.2.1) de cada solução no ranking $r = 1$, depois no ranking $r = 2$ e assim sucessivamente. Finalmente, este valor é multiplicado por um fator de conversão de escala.

$$F_i' = \frac{F_i \mu(r)}{\sum_{j=1}^{\mu(r)} F_j'} F_i' \tag{4.6}$$

Após estes cálculos, o MOGA comporta-se como um AG simples, com operadores de seleção, cruzamento e mutação usuais.

### Compartilhamento da aptidão

O objetivo do compartilhamento da aptidão (*fitness sharing*) é distribuir as soluções sob diferentes regiões ou nichos no espaço de busca. A Figura 4.7 ilustra um conjunto de soluções distribuídas em vários nichos (círculos).

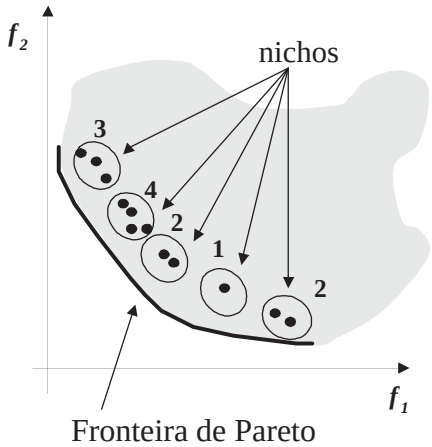



**Figura 4.7:** Conjunto de soluções agrupadas em nichos

Para cada solução $i$ é calculado um valor de contador de nicho $nc_i$ usando a equação 4.7, que representa o grau de multidão da região a que $i$ pertence.

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij}), \tag{4.7}$$

O valor $d_ij$ representa a distância entre duas soluções $i$ e $j$ que possuem o *mesmo valor de ranking* $r_i$. A distância no espaço de objetivos calcula-se como:

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left( \frac{f_k^{(i)} - f_k^{(j)}}{f_k^{max} - f_k^{min}} \right)^2}, \tag{4.8}$$

onde $f_k^{max}$ e $f_k^{min}$ são os valores mínimos e máximos para a k-ésima função objetivo.

A função $Sh$ da equação 4.7 é conhecida como *função de compartilhamento*, que foi proposta por Goldberg e Richardson [Goldberg e Richardson, 1987], sendo definida como:

$$Sh(d) = \begin{cases} 1 - \left( \frac{d}{\sigma_{share}} \right)^{\alpha}, & se \;\; d \leq \sigma_{share} \\ 0, & caso\ contrario \end{cases} \tag{4.9}$$

O parâmetro $d$ é a distância entre duas soluções, $\alpha$ define o comportamento da função $Sh$ e $\sigma_{share}$ é chamado de *raio de nicho*. O valor de raio de nicho define a vizinhança de uma solução. Quando $d > \sigma_{share}$ as soluções estão em nichos separados e $Sh(d) = 0$. Caso contrário, $Sh(d)$ assume um comportamento decrescente em relação a $\frac{d}{\sigma_{share}}$. Desta forma, quanto maior o número de soluções dentro nicho, maior o valor de $nc$.

MOGA usa a função $Sh$ com parâmetros $\alpha = 1$ é o parâmetro $\sigma_{share}$ é atualizado dinamicamente. Os cálculos do $\sigma_{share}$ estão descritos com detalhe no trabalho dos autores [Fonseca e Fleming, 1993]. Após esse procedimento,calcula-se o valor de *aptidão compartilhada* para cada solução $i$:

$$F_i' = \frac{F_i}{nc_i}$$

Desta forma as soluções que residem em um nicho menos ocupado terão melhor aptidão de compartilhamento. Este método permite destacar as soluções pouco representadas em cada ranking.

### Complexidade Computacional

Para obter os valores de $r_i$, cada uma das $N$ soluções deve ser comparada com $N-1$ soluções, em cada um dos $M$ objetivos. O número de comparações necessárias é da ordem $O(MN^2)$

O cálculo da aptidão média, dos nichos e da aptidão final, no pior caso, quando todas as soluções estão no mesmo ranking e no mesmo nicho, é da ordem $O(N^2)$. Portanto, a complexidade de MOGA é de $O(MN^2)$ [Deb, 2001].

**Algoritmo 4.1** Cálculo de Aptidão em MOGA

**Variáveis:**

$P$ : População
$N$ : Tamanho da população $P$.
$\sigma_{share}$ : Parâmetro de compartilhamento

1: Inicializar os contadores $\mu_j = 0$, para todos os possíveis rankings $j$, $j = 1, \ldots, N$.
2: Para cada solução $i$ em $P$.
3: Calcular o número de soluções $n_i$ que dominam à solução $i$.
4: Obter $r_i$ conforme a equação 4.4.
5: Incrementar o contador $\mu_{r_i}$
6: Ordenar descendentemente $P$ por $r_i$.
7: Identificar o valor máximo $r^* = max(r_i)$.
8: designar um valor $raw_i$ para cada solução $i$ em $P$.
9: Calcular o aptidão média para as soluções em $P$ conforme a equação 4.5.
10: Fazer $r = 1$.
11: Para cada solução $i$ com $r_i = r$
12: Calcular o contador de nicho $nc_i$ utilizando a fórmula 4.7.
13: Determinar o aptidão final usando a fórmula 4.6.
14: Se $r < r^*$
15: Fazer $r = r + 1$
16: Ir para o passo 11
17: Senão terminar.

### Vantagens e Desvantagens

A principal vantagem do MOGA é a sua simplicidade mo cálculo do valor de aptidão. Dado que o cálculo de nichos é realizado no espaço de objetivos, este algoritmo é facilmente aplicável em outros problemas [Deb, 2001]. Coello afirma que MOGA tem desempenho melhor quando comparados com outros algoritmos não elitistas [Coello Coello, 2001].

O valor $r_i$ não fornece os mesmos valores para soluções que se encontram na mesma frente, exceto a primeira. Isto pode gerar um ruído não desejado em algumas soluções do espaço de busca. Deb afirma que este algoritmo é sensível à forma da Fronteira de Pareto e à densidade das soluções no espaço de busca [Deb, 2001].

O cálculo do compartilhamento não garante que as soluções com valor $r_i$ alto terão uma pior aptidão que soluções com $r_i$ baixo. Isto acontece quando existem muitas soluções próximas entre si com $r_i$ baixo. O valor de contador de nicho para estas soluções é alto e a

aptidão compartilhada é pequena. Neste caso, não existe uma boa seleção das soluções de melhor $r_i$ e então a convergência do algoritmo é degradada [Deb, 2001].

### 4.2.2 NPGA (Niched Pareto Genetic Algorithm)

Este algoritmo foi proposto por Horn e colaboradores [Horn et al., 1994] e está baseado no conceito de não dominância. Este método se caracteriza porque não precisa de calcular um valor de aptidão que destaque soluções não dominadas. Ao contrário, utiliza um método de seleção por torneio denominado de *Torneio de Pareto*, baseado no conceito de dominância.

Durante o Torneio de Pareto, duas soluções $i$ e $j$ são escolhidas aleatoriamente da população $P$. Estas soluções são comparadas com um subconjunto $T$ de $P$ de tamanho $t_{dom} < |P|$. Podem acontecer os seguintes casos :

1. Se a solução $i$ domina o subconjunto $T$ e a solução $j$ é dominada por algum elemento de $T$, a solução $i$ é a vencedora. Reciprocamente, se a solução $j$ domina o subconjunto $T$ e a solução $i$ é dominada por algum elemento de $T$, a solução $j$ é a vencedora.

2. Caso contrário, se acontecer de ambas soluções dominarem $T$, ou que ao menos uma solução em $T$ domine $i$ e $j$, calcula-se o contador de nicho para escolher a solução vencedora.

O cálculo do nicho para $i$ e $j$ é realizado sobre a nova população gerada, diferente em relação ao cálculo de nicho sobre a nova população que está sendo gerada $(Q)$ . Os autores deste método chamaram de cálculo de compartilhamento dinamicamente atualizado.

Inicialmente, quando $Q$ é vazio, o cálculo de nicho não pode ser realizado e no caso de empate entre soluções candidatas, uma delas é escolhida aleatóriamente. As soluções ganhadoras $p_1$ e $p_2$ geram as soluções filhas $c_1$ e $c_2$ que são incluídas em $Q$.

Nas próximas iterações, o cálculo de nicho para escolher a solução ganhadora é baseada das soluções próximas em $Q$, usando a seguinte medida de distância:

$$d_{ik} = \sqrt{\sum_{m=1}^{M} \left( \frac{f_m^{(i)} - f_m^k}{f_m^{max} - f_m^{min}} \right)^2}$$

onde $f_m^{max}$ e $f_m^{min}$ são os valores mínimos e máximos da *k-ésima* função objetivo.

**Algoritmo 4.2** Modelo NPGA

**Variáveis:**

$P$ : População pai
$N$ : Tamanho da população $P$.
$Q$ : População filha
$\sigma_{share}$ : Parâmetro de compartilhamento
$t_{dom}$ : Tamanho do torneio
$n$ : Número de geração atual
$nMax$ : Número máximo de gerações

1: Criar a população inicial $P_0$ e a população filha $Q_0$ vazia.
2: Fazer $t = 0, i = 0$.
3: Realizar a seleção por torneio com os indivíduos $i$ e $i + 1$, o ganhador será o pai $p_1$.
4: $i = i + 2$.
5: Aplicar a seleção por torneio com os indivíduos $i$ e $i + 1$, o ganhador será o pai $p_2$.
6: Fazer o cruzamento de $p_1$ e $p_2$ para gerar os filhos $c_1$ e $c_2$.
7: Aplicar o operador de mutação.
8: Atualizar a população filha $Q_n = Q_n \cup \{c_1, c_2\}$.
9: $i = i + 1$.
10: Se $i < N$ ir para o passo 3.
11: Se $|Q| = N/2$ então
12: misturar a população $P$
13: $i = 1$
14: Voltar para o passo 3.
15: Se $n < nMax$ terminar.
16: senão fazer $P_{n+1} = Q_n$, $Q_{n+1} = \emptyset$, e voltar para o passo 3.

### Complexidade Computacional

Considerando o pior caso, tem-se que:

- Cada uma das duas soluções candidatas deve ser comparada com um subconjunto $T$ com $|T| = t_{dom}$ em cada objetivo. Supondo que são $M$ objetivos, existem $2Mt_{dom}$ comparações. Para os $N$ elementos da população existem $2NMt_{dom}$ comparações.

- No pior caso, para cada torneio é necessário calcular o valor de nicho, e portanto também as distâncias em relação ao conjunto $Q$, o qual vai crescendo linearmente com $|Q| = 2, 4, 6, \ldots, N - 2$. Para cada solução candidata se deve realizar este processo calculando $2|Q|$ distâncias.

Assumindo que $t_{dom}$ é pequeno comparado a $N$, tem-se que a complexidade total é governada por o cálculo do nicho, com uma complexidade total de $O(N^2)$. Quando $t_{dom}$ é um

valor da mesma ordem de $N$, a complexidade estará determinada pelos torneios realizados, obtendo-se assim uma complexidade de $O(MN^2)$ [Deb, 2001].

#### Vantagens e Desvantagens

A principal vantagem do algoritmo NPGA é que não precisa de um método para calcular o valor de aptidão das soluções. O NPGA está livre da subjetividade da função aptidão, diferente do MOGA.

Outra vantagem deste método é o uso do operador de seleção por torneio. Quando o parâmetro $t_{dom}$ é muito menor que o tamanho da população o algoritmo NPGA é computacionalmente eficiente pois as comparações de dominância se realizam apenas em um subconjunto pequeno.

Como desvantagem, o NPGA usa dois parâmetros adicionais $\sigma_{share}$ e $t_{dom}$. Quando o parâmetro $t_{dom}$ é muito pequeno, as comparações de dominância podem levar ao não destacamento das soluções não dominadas. Porém, quando $t_{dom}$ é muito grande, aumenta-se a complexidade computacional do algoritmo NPGA. Uma escolha adequada dos valores de ambos os parâmetros é necessária.

### 4.2.3 NSGA-II (Non-dominated Sorting Genetic Algorithm

O algoritmo NSGA-II [Deb et al., 2000] está baseado em um ordenamento elitista por não dominância. NSGA-II trabalha com a população pai $P$ para gerar a população filha $Q$ como nos AGs convencionais. Na primeira iteração, gera-se uma população $P_0$, a qual é ordenada por não dominância. Cada solução tem um valor de aptidão igual ao seu nível de não-dominância (1 é o melhor nível, 2 é o seguinte melhor nível e assim por diante). Depois, aplicando os operadores de seleção por torneio, cruzamento e mutação obtém-se a população filha $Q_0$. Tanto $P$ como $Q$ são de tamanho $N$.

Ambas população são unidas em uma população $R_0 = P_0 \cup Q_0$, com $|R| = 2N$. Para as seguintes gerações $n = 1, 2, \ldots$, o algoritmo NSGA-II trabalha com a população $R_n$.

Realiza-se um ordenamento por não dominância sobre $R_t$, obtendo as fronteiras $\mathcal{F}_1, \mathcal{F}_2, \ldots$ e todos estes conjuntos são inseridos na nova população $P_{n+1}$. Dado que apenas $N$ soluções podem ser inseridas, $N$ soluções de $R_n$ são descartadas. Para preencher as $P_{n+1}$ se começa pelas soluções em $\mathcal{F}_1$, logo $\mathcal{F}_2$ e assim por diante. Cada conjunto $\mathcal{F}_i$ deve ser inserido na sua totalidade em $P_{t+1}$, isto acontece enquanto $P_{n+1} + |F_i| \leq N$. Ao inserir um $\mathcal{F}_j$ tal que $|\mathcal{F}_j| > N - P_{n+1}$, o algoritmo NSGA-II escolhe aquelas soluções de $\mathcal{F}_j$ que estejam melhor espalhadas. A Figura 4.8 ilustra uma iteração para o NSGA-II.

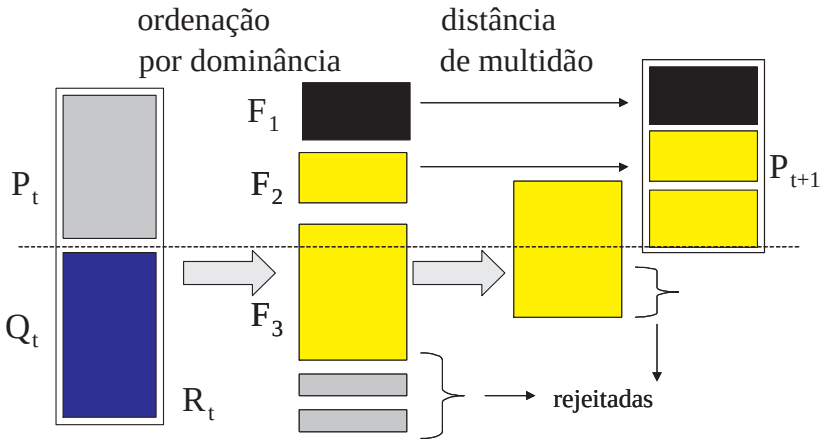



**Figura 4.8:** Esquema do modelo NSGA-II [Deb, 2001].

O algoritmo NSGA-II introduz um método chamado de distancia de multidão (*crowding distance*). Uma vez obtidos as distâncias, os conjuntos $\mathcal{F}_j$ são ordenados decrescentemente em relação as suas distâncias, e copia-se as primeiras $N - |P_{n+1}|$ soluções de $\mathcal{F}_j$ para $P_{n+1}$.

Finalmente, gera-se $|Q_{n+1}|$ a partir de $|P_{n+1}|$ usando o operador de seleção de torneio por multidão, cruzamento e mutação.

### Distância de Multidão

A Distância de Multidão de uma solução $i$, $d_i$, representa uma estimativa do perímetro formado pelo cubóide cujos vértices são os seus vizinhos mais próximos. A Figura 4.9 mostra a distância de multidão para a solução $i$. Quando maior o cubóide de $i$, mais afastada se encontra $i$ dos seus vizinhos. As soluções extremas em cada objetivo terão um cubóide infinito. O procedimento para achar a distância de aptidão está descrito no Algoritmo 4.3

**Algoritmo 4.3** Cálculo da distância de multidão em NSGA-II

**Variáveis:**

$\mathcal{F}_j$ : Conjunto de soluções na fronteira $i$

1: $l$ denota o número de soluções em $\mathcal{F}_j$.
2: Para cada solução em $\mathcal{F}_j$ atribui-se $d_i = 0$
3: Para cada função objetivo $m = 1, 2, \ldots, M$
4: Ordenar decrescentemente as soluções por $f_m$ na lista $I^m$.
5: Para cada solução extrema (mínimo e máximo) em cada um dos $M$ objetivos
6: Fazer $d_{I_1^m} = d_{I_l^m} = \infty$.
7: Para as soluções $i = 2, \ldots, l-1$ calcular :

$$d_{I_i^m} = d_{I_i^m} + \frac{f_m^{(I_{i+1}^m)} - f_m^{(I_{i-1}^m)}}{f_m^{max} - f_m^{min}} \tag{4.10}$$

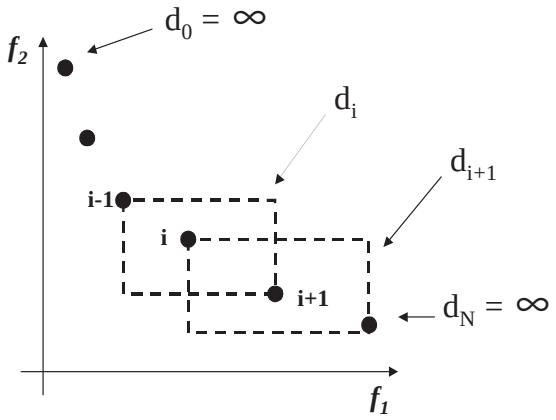



**Figura 4.9:** Cálculo da distância de multidão em NSGA-II [Deb, 2001].

onde $I_i^m$ representa a *i-ésima* solução na lista ordenada pelo objetivo $m$. $I_1^m$ e $I_l^m$ são os elementos da lista com menor e maior valor em um objetivo $m$. $f_m^{I_{i+1}^m}$ e $f_m^{I_{i-1}^m}$ são os valores dos vizinhos de $i$ na m-ésima função objetivo. $f_m^{max}$ e $f_m^{min}$ são parâmetros dos limites máximo e mínimo em cada objetivo. A Equação 4.10 assegura que as soluções mais afastadas tenham $d_i$ maior.

Alternativamente, é possível usar também o operador de compartilhamento para calcular o contador de nicho.

### Operador de Seleção por Torneio de Multidão

NSGA-II incorpora uma pequena modificação no método de seleção por torneio, usando o operador comparativo que leva em conta a multidão duma solução (crowded tournament selection operator) $<_c$. Uma solução $i$ é considerada ganhadora em um torneio contra uma solução $j$, se :

1. A solução $i$ possui um melhor nível de não dominância, $r_i < r_j$.
2. Se ambas soluções estão no mesmo nível, mas $i$ tem uma distância de multidão maior, $d_i > d_j$.

**Algoritmo 4.4** O Modelo NSGA-II

**Variáveis:**

$P$ : População pai
$Q$ : População filha
$N$ : Tamanho fixo para $P$ e $Q$
$F_j$ : Conjunto de soluções na fronteira $j$
$nMax$ : Máximo de gerações para o algoritmo
$n$ : Número de geração atual

1: Gerar a população inicial $P_0$ e $Q_0 = \emptyset$.
2: Atribuir $n = 0$
3: Realizar a seleção, o cruzamento e a mutação para gerar a população filha $Q_0$.
4: Fazer $R_n = P_n \cup Q_n$
5: Realizar a ordenação por não dominância em $R_n$
6: Criar $P_{n+1} = \emptyset$
7: Enquanto $|P_{n+1} + F_j| \leq N$
8: copiar as soluções de $F_j$ em $P_{n+1}$
9: Calcular as distâncias de multidão em $F_j$.
10: Ordenar $F_j$ conforme as distâncias $d_j$.
11: Copiar as primeiras $N - |P_{n+1}|$ soluções de $F_j$ para $P_{n+1}$.
12: Aplicar seleção de torneio por multidão para os indivíduos de $P_{n+1}$.
13: Aplicar os operadores de cruzamento e mutação e gerar a nova população $Q_{n+1}$
14: se $n > nMax$ então pare
15: senão atribuir $n = n + 1$ e voltar ao passo 4.

### Complexidade Computacional

No pior caso, a complexidade de NSGA-II é dada por:

- Para ordenar $R$ por não dominância é necessário comparar cada uma das $2N$ soluções contra $2N - 1$ soluções em cada um dos $M$ objetivos. Isto dá um total comparações da ordem $O(MN^2)$.
- Para obter a distâncias de multidão o pior caso, é quando todas as soluções de $R$ estão em $F_1$. Portanto é necessário ordenar $F_1$ para cada objetivo, obtendo uma ordem de $O(MN \log N)$.
- Finalmente, para passar as $N$ melhores soluções de $F_1$ a $P_{n+1}$, ordenamos conforme o operador $<_c$, o que resulta em $O(N \log N)$

Finalmente, a complexidade total do algoritmo NSGA-II é de $O(MN^2)$ [Deb, 2001].

### Vantagens e Desvantagens

A principal vantagem do NSGA-II é a maneira como mantém a diversidade entre as soluções não dominadas. O método de comparação por multidão é usado para a seleção por torneio e para escolher os elementos da fronteira $F_i$. O cálculo da distância de multidão não requer do parâmetro $\sigma_{share}$, como no MOGA [Deb, 2001].

Se o conjunto $F_1$ tem um tamanho maior que $N$, o processo de escolher apenas $N$ soluções usando a distância de multidão faz que sejam perdidas soluções. Seja um $F_1$ onde existam várias soluções Pareto-ótimas muito próximas e alguma solução distante não Pareto-ótima, mas não dominada no momento. Dado que o cuboide da solução não dominada é maior, esta solução será copiada em $|P_{n+1}|$ enquanto uma solução Pareto-ótima é eliminada. A Figura 4.10a mostra um conjunto de soluções de uma fronteira $F_i$. Depois de aplicar o algoritmo de corte para NSGA-II como mostrado na Figura 4.10b elimina-se uma solução Pareto-ótima e se mantém a solução não dominada (mas não Pareto-ótima). Esta situação faz que o NSGA-II possa cair em um ciclo de gerar soluções Pareto-ótimas e não Pareto-ótimas até convergir finalmente a um conjunto de soluções Pareto-ótimas [Deb, 2001].

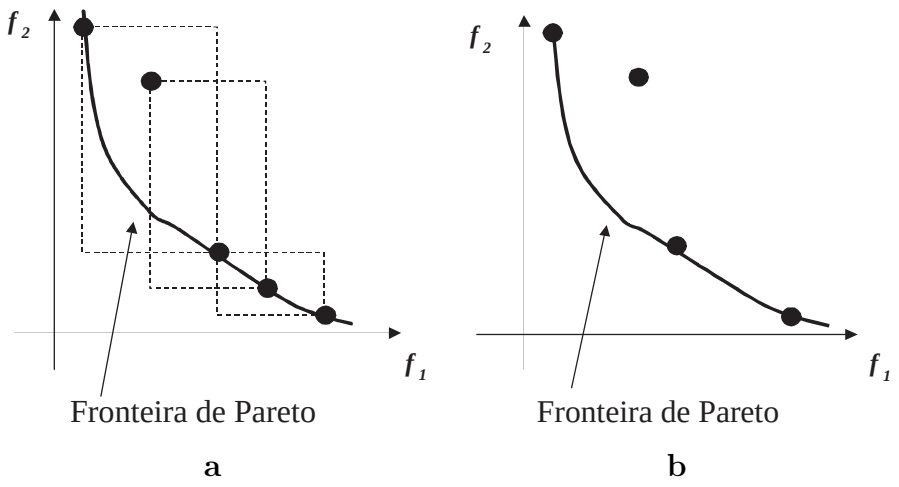



**Figura 4.10:** Situação onde o algoritmo de corte para NSGA-II erra.

## 4.2.4 SPEA2 (Strength Pareto Evolutionary Algorithm

O algoritmo SPEA2 [Zitzler et al., 2001] usa elitismo através de população externa $E$ onde são guardadas as soluções não dominadas. A população $E$ tem tamanho fixo de $N$ fornecido como parâmetro.

O algoritmo começa criando uma população aleatória $P_0$ e uma população externa $E$ inicialmente vazia. O valor de aptidão para as soluções de $Q = P \cup E$ é obtido em várias etapas. Primeiro, um valor de aptidão $s_i$ (*strenght fitness*) é encontrado usando a Equação 4.11.

$$s_i = |\{j, j \in Q, \quad \text{tal que} \quad i \preceq j\}| \tag{4.11}$$

o valor $s_i$ é o número de soluções que $i$ domina em $Q$. As soluções não dominadas têm $s_i = 0$. Depois, calcula-se o valor $r_i$ (raw fitness), conforme à equação:

$$r_i = \sum_{j \in Q, j \preceq i} s_j \tag{4.12}$$

Isto significa que $r_i$ é a soma dos $s_j$ das soluções $j$ que dominam $i$ em $Q$. Para as soluções não dominadas tem-se que $r_i = 0$. Soluções com um $r_i$ alto são dominadas por muitas soluções em $Q$. A Figura 4.11 apresenta um conjunto de soluções e seus valores $(r_i, s_i)$.

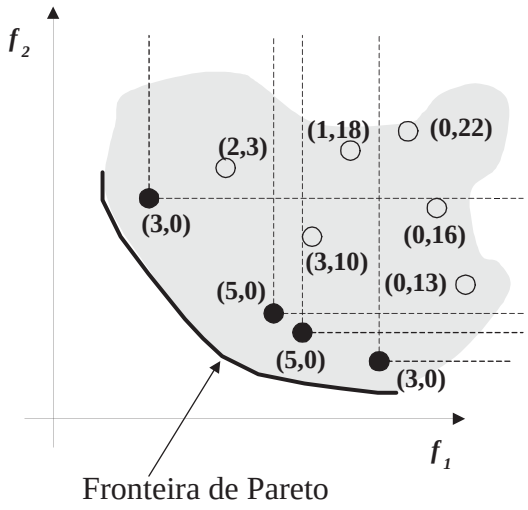



**Figura 4.11:** Esquema para o cálculo do aptidão no algoritmo SPEA2

Conforme os autores, este mecanismo permite uma espécie de ordenação de soluções por dominância, mas pode falhar quando existem muitas soluções não dominadas. Neste caso, existiriam muitas soluções com $r_i = 0$ e não enfatizaria a preferência de uma solução sobre uma outra. Para resolver este problema, SPEA2 usa uma informação de densidade, baseada no método de *k-vizinhos*, onde a densidade em qualquer ponto é uma função decrescente em relação ao *k-ésimo* ponto mais próximo. Para cada solução $i$ em $Q$, obtém-se as distâncias aos $|Q| - 1$ indivíduos de $Q$. Logo, estas distâncias são ordenadas em ordem ascendente. A densidade $d_i$ é formulada como:

$$d_i = \frac{1}{\sigma_i^k + 2} \tag{4.13}$$

a densidade de $i$ é inversamente proporcional à distância a seu *k-vizinho* mais próximo $\sigma_i^k$, onde $k = \sqrt{|Q|}$. A distância $d_i$ está dentro do intervalo aberto $(0, 1)$. Finalmente a aptidão para $i$ é :

$$F_i = r_i + d_i \tag{4.14}$$

Com esta fórmula, as soluções não dominadas terão $F_i < 1$, e as demais soluções $F_i \geq 1$.

Uma vez calculado o valor de aptidão copiamos as soluções não dominadas de Q para a nova população externa $E_{t+1}$. Depois disto, existem 3 possíveis situações:

1. $|E_{n+1}| = N$, e não se fazem modificações sobre $|E_{t+1}|$.
2. $|E_{n+1}| < N$, então ordena-se $Q$ por $F_i$ e copia-se as primeiras $N - |E_{n+1}|$ soluções $i$ de $Q$ tal que $F_i \geq 1$.
3. $|E_{n+1}| > N$, neste caso se utiliza um algoritmo de corte sobre $|E_{n+1}|$.

Finalmente, se realiza o processo de seleção por torneio, cruzamento e mutação sobre $E_{n+1}$ para gerar a nova população $P_{n+1}$.

### Algoritmo de Corte em SPEA2

O algoritmo de corte do SPEA2, reduz o tamanho de $E_{n+1}$ para $N$. Em cada iteração escolhe-se uma solução tal que a sua distância para o seu vizinho mais próximo seja a menor possível. No caso de empate, se calcula a segunda menor distância, e assim por diante. Formalmente, em cada iteração uma solução $i$ é eliminada tal que $i \leq_d j$ para todas as $j \in E_{n+1}$. Formalmente:

$$\begin{aligned} i \leq_d j, \quad & \text{se} \quad \forall 0 < k < |E_{n+1}| \sigma_i^k = \sigma_i^j \\ & \text{ou} \quad \exists 0 < k < |E_{n+1}| : [(\forall 0 < l < k : \sigma_i^l = \sigma_j^l) \ \text{ e } \ \sigma_i^k < \sigma_i^j] \end{aligned}$$

A figura 4.12a ilustra o conjunto de soluções pertencentes à população externa $E_n$ e as setas que indicam o 2-vizinho mais próximo. Depois de aplicar o algoritmo de corte para SPEA2, as soluções $2, 4$ e $7$ são eliminadas da população externa $E_{n+1}$ (Figura 4.12b).

Este algoritmo garante que as soluções extremas para cada objetivo sejam mantidas.

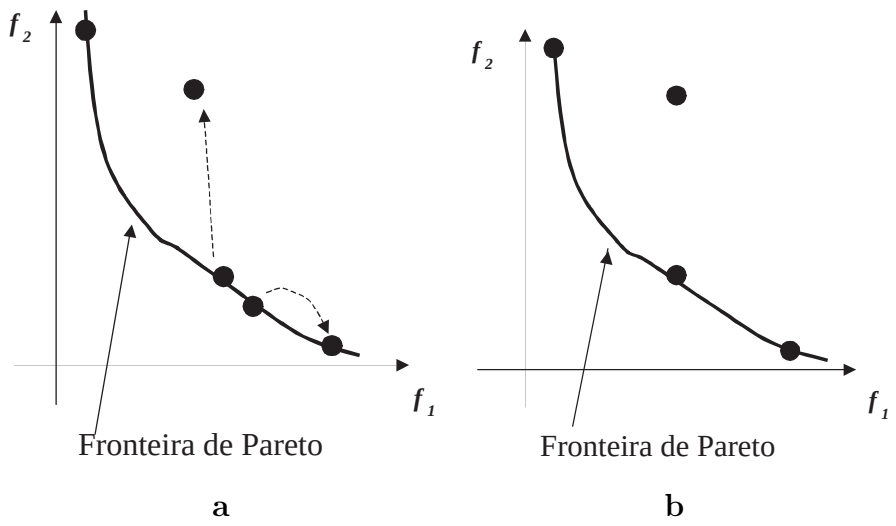



**Figura 4.12:** Algoritmo de Corte no modelo SPEA2

### Complexidade Computacional

A complexidade do algoritmo SPEA2 é dada por :

- Para realizar o cálculo de $s_i$ e $r_i$ precisamos comparar por dominância todas as soluções em $Q = P \cup E$ em cada objetivo. Se $|Q| = S$, tem-se que o número de comparações necessárias é da ordem $O(MS^2)$.
- A densidade $d_i$ requer ordenar cada par de distâncias de $Q$, para obter o k-ésimo vizinho. Cada solução $i$ em $Q$ possui uma lista com $S - 1$ elementos. Ordenhar cada lista tem uma complexidade de $O(MS^2 logS)$.
- Para o algoritmo de corte, escolher uma solução para ser eliminada pode levar ao pior caso $O(S^2)$.

A complexidade total do SPEA2 é governada pela ordenação das distâncias em $Q$. A complexidade total é então $O(MS^2 logS)$. Caso $|E| = N$ seja da mesma ordem que $|P|$, a complexidade total é de $O(N^2 logN)$. Porém, conforme os autores, aplicando estruturas de dados adequadas e deixando o fator $k = 1$ para o cálculo da densidade, a complexidade do SPEA2 pode ser reduzida para $O(MS^2)$, e portanto para $O(MN^2)$ [Laummanns, 2002].

**Algoritmo 4.5** O Modelo SPEA2

**Variáveis:**

$P$ : População interna
$E$ : População externa
$N$ : Tamanho máximo para $E$
$nMax$ : Máximo de gerações para o algoritmo
$n$ : Número de geração atual

1: Fazer n=0
2: Gerar a população inicial $P_0$ e $E_0 = \emptyset$.
3: Para cada solução $i$ em $Q_n = P_n \cup E_n$
4: Calcular os valores $s_i, r_i, d_i$ conforme as fórmulas 4.11, 4.12 e 4.13
5: Calcular o valor de aptidão $F_i$ dada por 4.14.
6: Copiar as soluções não dominadas de $Q_n$ em $E_{n+1}$.
7: Se $|E_{n+1}| > N$
8: Reduzir $P_{n+1}$ utilizando o algoritmo de corte.
9: Se $|E_{n+1}| < N$
10: Ordenar $Q_n$ conforme a $F_i$
11: Copiar as melhores $N - |E_{n+1}|$ soluções $i$ de $Q$ tal que $F_i \geq 1$
12: Se $n > nMax$ então parar
13: senão fazer $n = n + 1$
14: Aplicar seleção para os indivíduos de $E_{n+1}$.
15: Aplicar os operadores de cruzamento e mutação e gerar a nova população $P_{n+1}$, voltar para o passo 3.

### Vantagens e Desvantagens

SPEA2 introduz um método baseado na dominância de Pareto que não requer uma ordenação por não dominância, como no NSGA-II. Cada solução tem um aptidão proporcional ao número de indivíduos que domina, e também ao número de indivíduos que a dominam. Além disso a aptidão conta com uma informação sobre a densidade do indivíduo. O uso de uma população externa garante que as soluções Pareto-ótimas sejam conservadas para a geração seguinte.

O algoritmo de corte para SPEA2 padece do mesmo problema que o algoritmo de corte para NSGA-II. A Figura 4.13a apresenta um caso onde uma solução não Pareto-ótima, mas não dominada até o momento, seja preferida a uma solução Pareto-ótima após a aplicação do algoritmos de corte (Figura 4.13b). O cálculo de aptidão em SPEA2 não garante que as soluções externas que dominem o maior número de indivíduos tenham uma melhor aptidão. A Figura 4.13 apresenta o caso onde isto acontece. Finalmente, SPEA2 introduz um parâmetro adicional, o tamanho da população externa $E$.

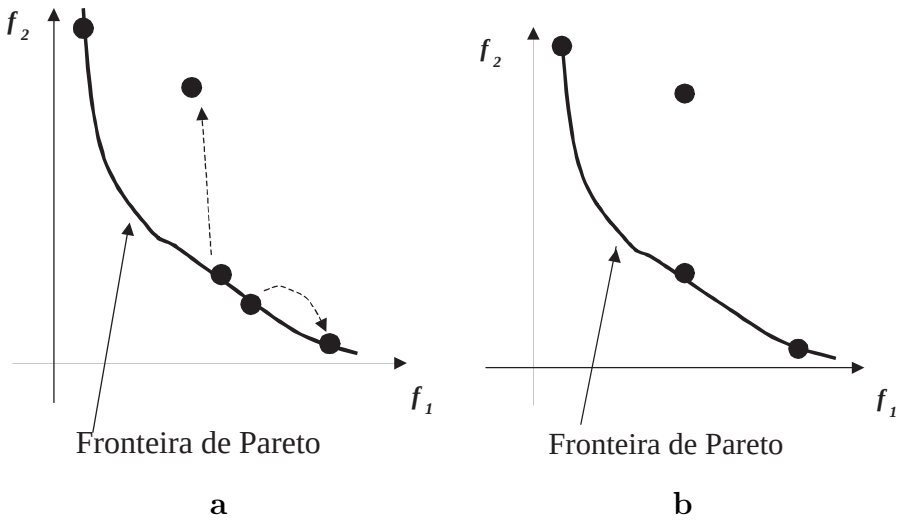



**Figura 4.13:** Situação onde o Algoritmo de corte em SPEA2 erra.

## 4.3 Métricas de Performance

Comparar experimentalmente diferentes métodos de otimização envolve a noção de performance. Para o caso da otimização multi-objetivo, medir a qualidade é mais complexo que no caso de otimização de objetivos simples [Zitzler et al., 2000].

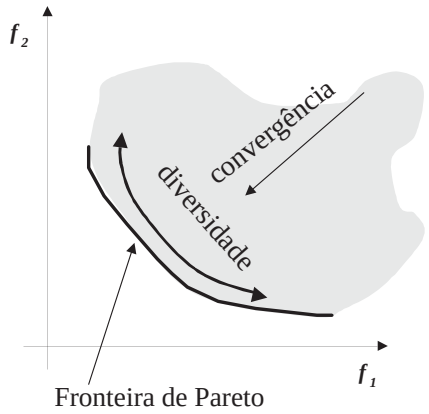



**Figura 4.14:** As dois metas da Otimização Multi-Objetivo [Deb, 2001]

.

Conforme mencionado no Capítulo 3, as metas da Otimização Multi-Objetivo são: encontrar soluções o mais perto possível da Fronteira de Pareto e obter a maior diversidade de soluções na fronteira. A Figura 4.14 ilustra ambas as metas. É necessário notar que convergência e diversidade podem ser metas conflitantes, portanto usar uma métrica não avalia completamente a performance de um algoritmo [Deb, 2001]. A Figura 4.15a ilustra um caso hipotético onde os resultados do algoritmo A tem boa convergência e pouca diversidade, ao contrário do algoritmo B (Figura 4.15b). Esta situação implica que um algoritmos não

é superior ao outro com relação aos critérios de convergência e diversidade, então são necessárias pelo menos de duas métricas, uma para medir a convergência e outra para calcular a diversidade.

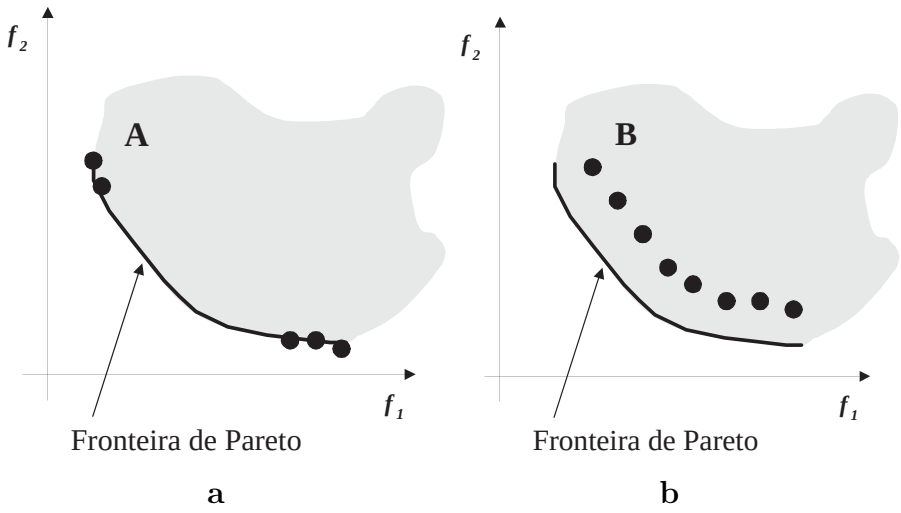



**Figura 4.15:** Distribuição vs. Convergência na Fronteira de Pareto. [Deb, 2001]

A Figura 4.16 ilustra outras situações. Na primeiro caso (Figura 4.16a), o algoritmo A é melhor que o algoritmo B. No segundo caso (Figura 4.16b), é difícil determinar qual algoritmo tem melhor performance. A comparação dos algoritmos dependerá fortemente da métrica usada.

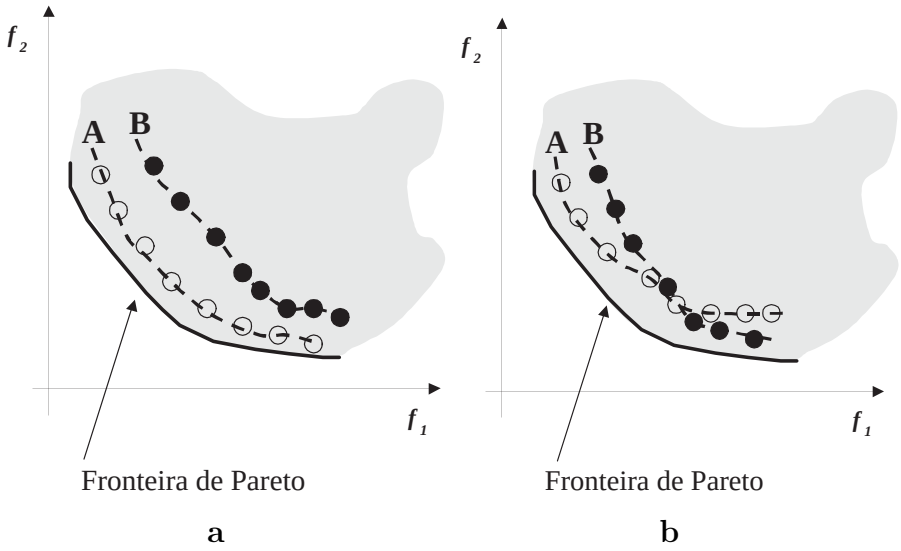



**Figura 4.16:** Distribuição vs. Convergência na Fronteira de Pareto. [Deb, 2001]

.

Nesta seção serão apresentadas algumas métricas de performance classificadas em dois grupos: métricas para medir a convergência e métricas de diversidade. Maiores detalhes podem ser encontrados na literatura [Zitzler et al., 2000, Deb, 2001, Veldhuizen, 1999].

### 4.3.1 Métricas de Convergência

Este tipo de métrica calcula a distância do conjunto de $N$ soluções $Q$ a um conjunto conhecido de soluções Pareto-ótimas $P^*$ que pode ser discreta ou contínua. Estas métricas são aplicáveis quando se conhece o conjunto Pareto-ótimo.

#### Taxa de Erro

Esta métrica conta as soluções de $Q$ que não estão no conjunto de soluções Pareto-ótimas $P^*$ [Veldhuizen, 1999], formalmente:

$$ER = \frac{\sum_{i=1}^{|Q|} e_i}{|Q|} \tag{4.15}$$

onde $e_i = 0$ se $i \in P^*$, e $e_i = 1$, caso contrário. Quanto menor for o valor de $ER$, melhor será a convergência. Se $ER = 0$, significa que $Q \subseteq P^*$.

#### Distância Geracional

É a Distância Média entre os conjuntos $Q$ e $P^*$ [Veldhuizen, 1999]:

$$GD = \frac{(\sum_{i=1} |Q| d_i^p)^{1/p}}{|Q|} \tag{4.16}$$

Para $p = 2$, $d_i$ é a distância euclidiana no espaço de objetivos entre a solução $i \in Q$ e o elemento $p \in P^*$ mais próximo. Formalmente:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*\,(k)})^2} \tag{4.17}$$

onde $f_m^{(k)}$ é o valor do *m-ésimo* objetivo da *k-ésima* solução de $P^*$. Quanto menor for $GD$ melhor a convergência de $Q$.

#### Métrica de Cobertura

Dados dois conjuntos de soluções $A$ e $B$, esta métrica calcula a proporção de soluções de $B$ que são fracamente dominadas pelas soluções de $A$ [Zitzler et al., 2000].

$$\mathcal{C}(A, B) = \frac{|b \in B,\ tal\ \ que\ \ \exists a \in A\ e\ a \preceq b|}{|B|} \tag{4.18}$$

Quando $\mathcal{C}(A, B) = 1$ todas as soluções de $B$ são fracamente dominadas por soluções de $A$. Se $\mathcal{C}(A, B) = 0$, então nenhuma solução de $B$ é fracamente dominada pelas soluções em

$A$. Dado que o operador de dominância não é simétrico, $\mathcal{C}(A, B)$ não é necessariamente igual a $\mathcal{C}(B, A)$. Então, deve-se calcular ambos valores para saber quantas soluções de $A$ dominam soluções $B$ e vice-versa.

### 4.3.2 Métricas de Diversidade

Estas métricas calculam a distribuição das soluções de um conjunto $Q$.

#### Espaçamento

Esta métrica calcula o desvio padrão entre as distâncias de soluções consecutivas [Schott, 1995]. Formalmente:

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \overline{d})^2} \tag{4.19}$$

onde

$$d_i = \min_{k \in Q \ e \ k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|, \quad e \quad \overline{d} = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|} \tag{4.20}$$

A variável $d_i$ representa o mínimo somatório do valor absoluto da diferença dos valores das funções objetivo entre a solução $i$ e qualquer outra solução em $Q$. Por outro lado, $\overline{d}$ representa a média dos valores $d_i$. Quanto menor for o desvio $S$, melhor distribuídas estarão as soluções.

#### Número de nichos

Esta métrica calcula o número de nichos dentro de um conjunto de soluções $Q$ [Zitzler et al., 2000].

$$NC = \frac{1}{|Q| - 1}\sum_{i=1}^{|Q|} |j \in Q, \text{ tal que } d_{ij} > \sigma| \tag{4.21}$$

sendo $d_{ij}$ a distância entre as soluções $i$ e $j$ de $Q$. Esta distância pode ser calculada no espaço de variáveis ou de objetivos. O valor $NC$ representa o número de soluções cuja distância entre elas é maior que o parâmetro $\sigma$. Quando $d_{ij} < \sigma$, as soluções $i$ e $j$ estão no mesmo nicho. Quanta maior a quantidade de nichos formados em $Q$, melhor distribuídas estão as soluções.

#### Espalhamento Máximo

Esta métrica retorna a extensão máxima das soluções em $Q$. [Zitzler et al., 2000]. Formalmente:

$$M3 = \sqrt{\sum_{m=1}^{M} (\max_{i=1}^{|Q|} f_m^i - \min_{i=1}^{|Q|} f_m^i)^2} \tag{4.22}$$

Um maior valor para $M3$ significa uma melhor cobertura do espaço (neste caso, espaço de objetivos).

## 4.4 Comentários Finais

Neste capítulo foram apresentados os Algoritmos Genéticos (AGs) como técnica de otimização. As características dos AGs permitem seu emprego em problemas de otimização Multi-Objetivo. Foram detalhados os principais modelos de AGs para estes problemas. Finalmente, algumas métricas para avaliar o desempenho dos AGs foram mostradas. Todos estes conceitos, vistos nos dois primeiros capítulos, serão usados para formular e aplicar MOEAs ao problema de Alinhamento de Seqüências Biológicas, o qual será apresentado no próximo capítulo.

Capítulo

5

# Um modelo de Algoritmo Genético Multi-Objetivo para o Alinhamento de Seqüências Biológicas

Neste capítulo é apresentado o modelo proposto para o problema de alinhamento de proteínas. Na primeira seção, o alinhamento de proteínas será definido em termos de um MOOP. O modelo de MOEA para alinhamento de proteínas é detalhado na segunda seção. A seguir, na seção 4, são formulados experimentos para testar o modelo e finalmente na seção 5, os resultados são apresentados e discutidos.

## 5.1 Formulação do Problema

No Capitulo 2 foi apresentado o problema de alinhamento de seqüências como um problema de otimização de objetivos simples, onde é procurada a solução com maior pontuação.

Para o caso de um MOOP, serão considerados dois critérios: a extensão e a qualidade. A extensão do alinhamento é definida como o comprimento da seqüência mais curta. O alinhamento pode ser visto também como a procura de uma subcadeia dentro de uma cadeia maior, permitindo a inserção de buracos. Por exemplo, seja o seguinte alinhamento de duas proteínas:

```
--P-AW-HEAE
HEAGAWGHE-E

Extensão :  9
```

Neste caso, a extensão é definida pela primeira seqüência (PAWHEAE) sem considerar os buracos do começo e do final. Portanto, considera-se apenas os buracos que estão dentro da seqüência. Neste caso, são dois buracos que somados aos sete caracteres da seqüência, da uma extensão total de nove.

A qualidade está definida pela pontuação obtida *dentro da extensão* do alinhamento. A pontuação será calculada considerando o critério de afinamento de buracos (*affine gaps*) onde se tem o custos de primeiro buraco que divide uma seqüência, e o custo de extensão para os buracos a continuação do primeiro buraco. No caso do alinhamento anterior e considerando a matriz de pontuações BLOSUM50, com o custo de primeiro buraco de -12 e de extensão do buraco de -2, tem-se uma pontuação de -1.

Em outros termos, o problema consiste em achar aquelas soluções com maior qualidade para cada extensão possível. O MOOP está formulado como:

$$\left.\begin{array}{rl} \text{maximizar} & f_1(\mathbf{x}), \\ & f_2(\mathbf{x}), \\ \text{restrita a} & \mathbf{x} \text{ seja um alinhamento válido} \end{array}\right\} \tag{5.1}$$

Onde $f_1(\mathbf{x})$ é a extensão do alinhamento $\mathbf{x}$, $f_2(\mathbf{x})$ é a qualidade do alinhamento $\mathbf{x}$.

## 5.2 Modelo Proposto

Nesta seção será descrito o modelo de MOEA proposto (SPEAModProp) para resolver o problema da equação 5.1.

### 5.2.1 Representação das Soluções

Internamente, cada alinhamento está representado segundo o modelo utilizado no trabalho de Romero Zaliz [Romero Zaliz, 2001]. A solução está representada por um vetor de números inteiros que indicam a posição onde estão presentes os buracos, para ambas as seqüências.

Por exemplo:

Solução:

| 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 0 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|

Representa:

```
PAWHEAE---------
---HEAEAWGHEAEGE
```

Solução:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 14 | 15 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|

Representa:

```
-------PAWHEAE--
HEAEAWG---HEAEGE
```

No exemplo pode ser observado que o vetor está dividido por uma linha dupla vertical. À esquerda da linha estão os valores das posições dos buracos para a primeira seqüencia. À direita estão os valores que representam as posições dos buracos na segunda seqüência.

### 5.2.2 Geração de soluções iniciais

O Modelo Proposto gera uma população inicial de soluções aleatórias, produzindo alinhamentos válidos.

Para gerar uma solução aleatória, calcula-se o número máximo e mínimo de buracos que podem ser inserido em cada seqüência. O número máximo de buracos em uma seqüência é igual ao comprimento da outra seqüência. Isto significa produzir um alinhamento onde as letras de cada seqüência estejam alinhadas com um buraco, por exemplo:

```
----------PAWHEAE
HEAGAWGHEE-------
```

Para a primeira seqüência, o número máximo de buracos é 10, e para a segunda seqüência, 7.

O número mínimo de buracos é simplesmente a diferença dos comprimentos no caso da seqüência mais curta, e zero no caso da seqüência mais comprida. No exemplo anterior. o número mínimo de buracos para a primeira seqüência é 3, um alinhamento provável seria:

```
PAWHEAE---
HEAGAWGHEE
```

Em seguida, gera-se o número de buracos para a primeira seqüência no intervalo $[3, 10]$, por exemplo 5. Conseqüentemente, o alinhamento vai ter 12 posições onde os buracos podem ser inseridos, das quais cinco serão preenchidas com buracos e o resto com as letras da seqüência. Neste exemplo, os buracos ocuparão as posições $[1, 5, 6, 7, 10]$. Para a segunda seqüência apenas dois buracos serão necessários para completar o alinhamento, neste caso são escolhidas as posições $[3, 12]$. Deve-se garantir que não existam dois buracos nas mesmas posições. O alinhamento resultante é:

```
-PAW---HE-AE
HE-AGAWGHEE-
```

Desta forma, são geradas todas as soluções correspondentes à população inicial do algoritmo.

### 5.2.3 Operadores Genéticos

#### Seleção

Para este modelo foi usado a seleção por torneio. Basicamente, duas soluções são escolhidas e aquela de melhor valor de aptidão passa para a lista de soluções para reprodução.

#### Operador de Cruzamento

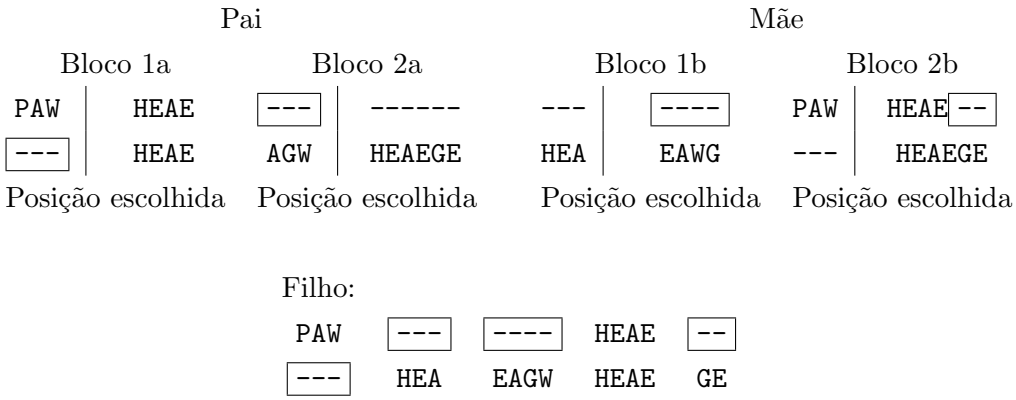
Nos AGs, o operador de cruzamento é usado para combinar as melhores características das soluções encontradas para que possam ser passadas para a geração seguinte. É necessário que o operador permita procurar a maior quantidade possível de soluções, sem reduzir o espaço de busca. Na implementação utilizada neste trabalho foram utilizados dois operadores baseados no trabalho de Romero Zaliz [Romero Zaliz, 2001].

O operador *OnePointCombine* (Combinação de um ponto) trabalha sobre *Blocos de Cruzamento*. Para obtê-los, divide-se os alinhamentos pais em duas partes. Por exemplo os alinhamentos:

```
PAWHEAE---------  -------PAWHEAE--
---HEAEAWGHEAEGE  HEAEAWG---HEAEGE
```

| Pai | | Mãe | |
|---|---|---|---|
| Bloco 1a | Bloco2a | Bloco 1b | Bloco2b |
| PAWHEAE | --------- | ------- | PAWHEAE-- |
| ---HEAE | AWGHEAEGE | HEAEAWG | ---HEAEGE |

Depois, é escolhida uma posição dentro de cada Bloco de Cruzamento. Primeiro copia-se os buracos à esquerda da posição escolhida no pai. Em seguida, os buracos à direita da posição escolhida na mãe. Finalmente, é obtido um filho juntando-se as partes selecionadas dos blocos 1a-1b-2a-2b.




Para gerar um segundo filho, usa-se o operador *GoodPosCombine* (Combinação de boas posições). Este operador extrai os buracos comuns à mãe e ao pai. O alinhamento é finalizado preenchendo com buracos tomados aleatoriamente do pai ou da mãe. Exemplo:

```
        Pai:                    Mãe:
PAWHEAE---------        -------PAWHEAE--
---HEAEAWGHEAEGE        HEAEAWG---HEAEGE

                Filho:
            PAW----HEA-E----
            --HEAEAOWGHEAEGE
```

Às vezes, os operadores de cruzamento podem gerar filhos inválidos. Neste caso, os filhos substituídos pelos pais.

### Operador de mutação

O operador genético de mutação realiza pequenas variações nos alinhamentos. São três os operadores de mutação usados:

1. *AddGap* (Agregar Buraco) como o próprio nome sugere, agrega um buraco em qualquer parte do alinhamento, tanto na primeira como na segunda seqüência.

2. *DeleteGap* (Apaga Buraco) elimina um buraco do alinhamento. Pode ser utilizado em ambas as seqüências.

3. *ShiftGap* (Deslocar Buraco) para esquerda ou direita. Para realizar o deslocamento de um buraco para uma posição adjacente que está ocupada por um outro buraco, o deslocamento é feito recursivamente.

Exemplos:

Agregar Buraco na sexta posição da segunda seqüência:

```
PAWHEAE---------  PAWHEAE----------
---HEAEAWGHEAEGE  ---HEA-EAWGHEAEGE
```

Apagar Buraco da quarta posição da primeira seqüência:

```
-------PAWHEAE--  ------PAWHEAE---
HEAEAWG---HEAEGE  HEAEAWG---HEAEGE
```

Deslocar Buraco da segunda posição da segunda seqüência para a esquerda:

```
PAWHEAE---------  PAWHEAE---------
---HEAEAWGHEAEGE  --H-EAEAWGHEAEGE
```

É possível que os operadores de mutação gerem alinhamentos inválidos. Caso isso ocorra, as mudanças são descartadas.

### 5.2.4 Nichos

O Modelo Proposto usará nichos para que as boas soluções que estejam em diferentes posições do alinhamento sejam mantidas. Dados os seguintes alinhamentos:

```
PAWHEAE---------  -------PAWHEAE--
---HEAEAWGHEAEGE  HEAEAWG---HEAEGE

 Extensão :  9     Extensão :  9
 Qualidade:  9    Qualidade :  9
```

Se o operador de dominância é aplicado diretamente, a segunda solução será eliminada embora seja muito diferente da primeira. Para evitar este caso, a dominância será aplicada apenas às soluções que estejam em um mesmo nicho. Este critério é conhecido também de *dominância local*.

Para determinar que duas soluções estão no mesmo nicho será usada uma medida de distância que é definicda a seguir.

**Definição 8** *A distância entre dois alinhamentos é o número de buracos que não são comuns entre as duas seqüências de busca para os alinhamentos.*

Por exemplo, coloca-se as primeiras seqüências usadas no exemplo uma acima da outra:

```
PAWHEAE---------
-------PAWHEAE--
```

As duas compartilham dois buracos (no final). Portanto a distância entre elas estará dada pelo número de buracos não comuns, no total 14, o que significa que os dois alinhamentos estão muito distantes e não podem ser comparados com o critério de dominância.

### 5.2.5 Cálculo do Valor de Aptidão

O valor de aptidão para cada solução é calculado conforme o algoritmo SPEA [Zitzler e Thiele, 1998], que foi modificado para incluir o conceito de dominância local.

A aptidão para a solução $i$ da população externa ($E$) é denotado como $S_i$ (strenght fitness), sendo proporcional ao número de soluções ($n_i$) da população interna que a solução $i$ domina *localmente*.

$$S_i = \frac{n_i}{N+1} \tag{5.2}$$

A divisão por $N+1$, onde $N$ é o número de soluções da população externa ($E$), garante que $S_i$ não seja maior ou igual a 1.

O valor de aptidão $F_j$ para uma solução $j$ da população interna é definido como a soma dos valores $S_i$ das soluções da população externa que dominam $j$.

$$F_j = 1 + \sum_{i \in E \ e \ i \preceq_\sigma j} S_i \tag{5.3}$$

O símbolo $\preceq_\sigma$ representa a dominância local dentro da vizinhança de tamanho $\sigma$. A soma de 1 no $F_j$ faz com que o valor de aptidão de qualquer solução da população interna seja maior que a aptidão das soluções da população externa. Quanto menor o valor de aptidão, melhor é a solução.

### 5.2.6 Elitismo

O operador de Elitismo preserva as melhores soluções encontradas até o momento. O algoritmo SPEA usa este conceito, guardando as soluções não dominadas em um conjunto externo. O procedimento de Elitismo está descrito no Algoritmo 5.1.

**Algoritmo 5.1** Elistismo para o modelo proposto

**Variáveis:** $P$ : População interna
$E$ : População externa
$N$ : Tamanho máximo $E$

1: Copiar todas as soluções não dominadas localmente de $P$ para $E$
2: Remover as soluções de $E$ que são dominadas localmente por quaisquer outros membros de $E$.

Para esta implementação, o algoritmo de corte foi omitido devido a que a população externa não vai ter restrição de tamanho. Isto foi necessário porque a Fronteira de Pareto para o problema de alinhamento de seqüencias é desconhecida. No Algoritmo 5.2 está detalhado o Modelo Proposto.

**Algoritmo 5.2** O Modelo Proposto para o Problema de Alinhamento de Seqüências Biológicas (SPEAModProp)

**Variáveis:** $P$ : População interna
$E$ : População externa
$nMax$ : Máximo de gerações para o algoritmo
$n$ : Número de geração atual

1: Gerar a população inicial $P$
2: Fazer $E = \emptyset$ n = 0.
3: Aplicar elitismo conforme ao Algoritmo 5.1.
4: Calcular o valor de aptidão para cada membro de $P$ e $E$ confome as equações 5.2 e 5.3.
5: Aplicar seleção para os indivíduos de $P + E$.
6: Aplicar os operadores de cruzamento e mutação.
7: Se $n > nMax$ então parar
8: Senão
9: $n = n + 1$
10: voltar para o passo 3.

## 5.3 Experimentos

O SPEAModProp será comparado com dois modelos de MOEA conhecidos como SPEA2 e NSGA-II. Estes algoritmos foram ligeiramente modificados para incorporar o conceito de dominância local.

### 5.3.1 Algoritmo SPEA2 Modificado (SPEA2Mod)

O algoritmo SPEA2m descrito na seção 4.2.4 do Capitulo 4, foi modificado em seu método de cálculo de aptidão para levar em conta a dominância local.

O valor de aptidão $s_i$ *strength fitness* agora é calculado como:

$$s_i = |\{|j|, j \in Q, \quad \text{tal que} \quad i \preceq_\sigma j\}| \tag{5.4}$$

onde o símbolo $i \preceq_\sigma$ representa a dominância local dentro da vizinhança de tamanho $\sigma$. Formalmente, $i \preceq_\sigma j$ *se* $i \preceq j$ *e* $d_{ij} \leq \sigma$.

Da mesma forma, o valor $r_i$ foi modificado como segue:

$$r_i = \sum_{j \in Q, j \preceq_\sigma i} s_j \tag{5.5}$$

Finalmente, o valor de aptidão permaneceu sem modificações

$$F_i = r_i + d_i \tag{5.6}$$

onde o valor $d_i$ é a informação de densidade, como explicado na seção 4.2.4 do Capítulo 4.

A outra modificação feita no algoritmo SPEA2 foi na estratégia de elitismo. Ao invés de manter uma população externa de tamanho constante, foi adotado o modelo descrito no Algoritmo 5.1. O algoritmo de corte não é usado. O Algoritmo 5.3 descreve o modelo SPEA2 modificado (SPEA2Mod).

**Algoritmo 5.3** O Modelo SPEA2Mod

**Variáveis:**

$P$ : População interna
$E$ : População externa
$N$ : Tamanho máximo para $E$
$nMax$ : Máximo de gerações para o algoritmo
$n$ : Número de geração atual

1: Fazer n=0
2: Gerar a população inicial $P_0$ e $E = \emptyset$.
3: Para cada solução $i$ em $Q_n = P_n \cup E_n$
4: Calcular os valores $s_i, r_i$ conforme as fórmulas 5.4, 5.5
5: Calcular o valor de aptidão $F_i$ dada por 5.6.
6: Aplicar elitismo conforme ao algoritmo 5.1.
7: Se $n > nMax$ então parar
8: senão fazer $n = n + 1$
9: Aplicar seleção para os indivíduos de $E_{n+1}$.
10: Aplicar os operadores de cruzamento e mutação.
11: Gerar a nova população $P_{n+1}$
12: Voltar para o passo 3.

### 5.3.2 Algoritmo NSGAII Modificado (NSGAIIMod)

O algoritmo NSGA-II foi modificado apenas no método de ordenação por não dominância. A ordenação no NSGA-II modificado é feita por não dominância *local*, ou seja, as fronteiras $\mathcal{F}_1, \mathcal{F}_2, \ldots$ são obtidas usando este conceito. O resto do algoritmo permaneceu sem alterações. O algoritmo NSGA-II Modificado está descrito no Algoritmo 5.4.

**Algoritmo 5.4** O Modelo NSGAIIMod

**Variáveis:**

$P$ : População pai
$Q$ : População filha
$N$ : Tamanho fixo para $P$ e $Q$
$F_j$ : Conjunto de soluções na fronteira $j$
$nMax$ : Máximo de gerações para o algoritmo
$n$ : Número de geração atual

1: Gerar a população inicial $P_0$ e $Q_0 = \emptyset$.
2: Atribuir $n = 0$
3: Realizar a seleção, o cruzamento e a mutação para gerar a população filha $Q_0$.
4: Fazer $R_t = P_t \cup Q_t$
5: Realizar a ordenação por *não dominância local* em $R_n$
6: Criar $P_{n+1} = \emptyset$
7: Enquanto $|P_{n+1} + F_j| \leq N$
8: copiar as soluções de $F_j$ em $P_{n+1}$
9: Calcular as distâncias de multidão em $F_j$.
10: Ordenar $F_j$ conforme as distâncias $d_j$.
11: Copiar as primeiras $N - |P_{n+1}|$ soluções de $F_j$ para $P_{n+1}$.
12: Aplicar seleção de torneio por multidão para os indivíduos de $P_{n+1}$.
13: Aplicar os operadores de cruzamento e mutação e gerar a nova população $Q_{n+1}$
14: se $n > nMax$ então pare
15: senão atribuir $n = n + 1$ e voltar ao passo 4.

### 5.3.3 Experimentos

Para os experimentos são usadas seqüências artificiais de proteínas que permitem avaliar o desempenho dos três algoritmos. As seqüências são:

PPHEAGA
HEAGAPWGHEEPAWHEAE

Os parâmetros usados para cada algoritmo foram determinadas experimentalmente e são mostrados na Tabela 5.1.

| Parâmetro | SPEAModProp | SPEAModProp | NSGAIIMod |
|---|---|---|---|
| Número de Gerações | 100 | 100 | 100 |
| Tamanho da população | 1000 | 1000 | 500 |
| Tamanho da população externa | Var | Var | 500 |
| Probabilidade de cruzamento | 0.6 | 0.6 | 0.6 |
| Probabilidade de mutação | 0.3 | 0.3 | 0.3 |
| Diâmetro do nicho (espécies) | 3 | 3 | 3 |

**Tabela 5.1:** Parâmetros utilizados para os Modelos SPEAModProp, SPEA2Mod e o NSGAIIMod

Os resultados obtidos nas execuções para cada algoritmo foram juntadas. A este resultado foi aplicada a dominância local para obter un Pareto-Ótimo Local de cada algoritmo. Finalmente os Pareto-ótimos Locais de cada algoritmos foram combinados para obter o Pareto-Ótimo Local total. A Tabela 5.2 mostra as soluções do Pareto Ótimo Local (segunda coluna) e as soluções de cada algoritmo (colunas SPEAModProp, SPEA2Mod e NSGAII-Mod). As soluções são agrupadas pela extensão. Esta forma de agrupamento é simplesmente ilustrativa e não corresponde aos nichos formados. O algoritmo SPEAModProp encontrou a maior quantidade de soluções seguido do SPEA2Mod e NSGAIIMod, este último achou poucas soluções diferentes e muitas iguais as que foram eliminadas previamente. Estes dados são apenas quantitativos e não fornecem informação sobre a qualidade das soluções.

| Extensão | Pareto-Ótimo Local | SPEAModProp | SPEA2Mod | NSGAIIMod |
|---|---|---|---|---|
| 7 | 2 | **3** | 2 | 2 |
| 8 | 5 | **4** | 3 | 4 |
| 9 | 8 | 8 | 11 | 4 |
| 10 | 6 | **10** | 6 | 10 |
| 11 | 16 | **22** | 15 | 11 |
| 12 | 13 | **21** | 17 | 10 |
| 13 | 18 | **26** | 26 | 12 |
| 14 | 16 | **26** | 25 | 9 |
| 15 | 20 | 32 | 36 | 11 |
| 16 | 25 | **38** | 35 | 4 |
| 17 | 33 | **52** | 37 | 6 |
| 18 | 37 | **61** | 46 | 5 |
| 19 | 60 | 76 | 88 | 5 |
| 20 | 49 | **93** | 66 | 7 |
| 21 | 99 | **129** | 125 | 7 |
| 22 | 135 | **178** | 151 | 5 |
| 23 | 189 | **205** | 198 | 3 |
| 24 | 209 | 191 | 197 | 1 |
| 25 | 276 | 161 | 171 | 0 |
| TOTAL | 1216 | **1336** | 1255 | 116 |

**Tabela 5.2:** Resumo dos resultados obtidos por SPEAModProp, SPEA2Mod e NSGAIIMod junto ao Pareto-Ótimo Local

Algumas soluções fornecidas pelos três algoritmos são mostradas na Tabela 5.3. Pode ser visto que todos os algoritmos encontraram a seqüência de busca em diferentes posições da seqüência objetivo. Os três algoritmos foram capazes de obter a solução ótima que corresponde a executar o algoritmo de programação dinâmica:

```
PPHEAGA-------------
--HEAGAPWGHEEPAWHEAE
Extensão :7
Qualidade :30
```

SPEAModProp e SPEA2Mod acharam um grande número de soluções para várias extensões e qualidade, embora muitas delas sejam de qualidade baixa.

| | | |
|---|---|---|
| -----P-PHEAGA-----<br>HEAGAPWGHEEPAWHEAE<br>Extensão :8<br>Qualidade:14 | PPHEAGA-------------<br>--HEAGAPWGHEEPAWHEAE<br>Extensão :7<br>Qualidade:30 | PPHEAG------------A-<br>--HEAGAPWGHEEPAWHEAE<br>Extensão :19<br>Qualidade:-4 |
| -----P-PHEAG----A-<br>HEAGAPWGHEEPAWHEAE<br>Extensão :12<br>Qualidade:-4 | ----------PPHEAGA-<br>HEAGAPWGHEEPAWHEAE<br>Extensão :7<br>Qualidade:4 | PPHEA-G-A-----------<br>--HEAGAPWGHEEPAWHEAE<br>Extensão :9<br>Qualidade:-10 |
| -----P-----P--HEAGA<br>HEAGAPWGHEEPAWHEAE-<br>Extensão :14<br>Qualidade:2 | -----P-PHEAGA-----<br>HEAGAPWGHEEPAWHEAE<br>Extensão :8<br>Qualidade:14 | ---P-P--HEAGA-----<br>HEAGAPWGHEEPAWHEAE<br>Extensão :10<br>Qualidade:0 |
| --PPHEAG----A-----<br>HEAGAPWGHEEPAWHEAE<br>Extensão :11<br>Qualidade:-14 | PPHEA-GA------------<br>--HEAGAPWGHEEPAWHEAE<br>Extensão :8<br>Qualidade:4 | --PPHEAGA---------<br>HEAGAPWGHEEPAWHEAE<br>Extensão :7<br>Qualidade:-3 |
| -----P-PHE--AGA---<br>HEAGAPWGHEEPAWHEAE<br>Extensão :10<br>Qualidade:-2 | ----PP--------HEAGA<br>HEAGAPWGHEEPAWHEAE-<br>Extensão :15<br>Qualidade:-1 | |

**Tabela 5.3:** Amostra dos alinhamentos obtidos por SPEAModProp, SPEA2Mod e NSGAIIMod

Para medir o desempenho dos algoritmos, são usadas as métricas descritas na seção 4.3 do Capítulo 4. Em cada execução dos algoritmos, foram calculadas as métricas de Distância Geracional (GD), Cobertura $\mathcal{C}$(ambas em relação ao Pareto-Ótimo Local $P^*$), Espaçamento (SP) e Espalhamento Máximo (M3). Depois foi calculada a média e o Desvio Padrão para cada métrica. Os resultados são mostrados na Tabela 5.4

| Alg/Metrica | | $\mathcal{C}$(P*,P) | M3 | SP | GD |
|---|---|---|---|---|---|
| SPEAModprop | Valor | **0.3821** | **193.8385** | 0.9058 | 0.0855 |
| | Desvio | 0.0291 | 6.5193 | 0.0348 | 0.0020 |
| SPEA2Mod | Valor | 0.4430 | 180.8990 | **0.8847** | **0.0829** |
| | Desvio | 0.0244 | 6.6347 | 0.0386 | 0.0013 |
| NSGAIIMod | Valor | 0.4611 | 141.7797 | 1.8452 | 0.3390 |
| | Desvio | 0.1563 | 24.5089 | 0.1789 | 0.0526 |

**Tabela 5.4:** Resultados das Métricas para os algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod

### Interpretação dos Resultados

Os resultados obtidos por cada algoritmo são analisados para cada métrica, o que revela diferentes aspectos da performance.

- Distância Geracional (GD). Esta métrica representa o valor médio das distâncias entre as soluções achadas pelos algoritmo ao Pareto-Ótimo Local ($P^*$). Observa-se na Tabela 5.4 que SPEAModProp e SPEA2Mod obtiveram valores muito próximos a zero (0.08) entanto NSGAIIMod obteve um valor maior (0.3). Pode-se concluir que SPEAMod-Prop e SPEA2 tiveram resultados muito próximos ao Pareto-Ótimo Local.

- Espaçamento (SP). Esta métrica calcula o Desvio Padrão entre soluções consecutivas obtidas pelos algoritmos. Um valor pequeno para SP significa que o espaço entre soluções consecutivas é uniforme. Conforme à Tabela 5.4 SPEAModProp e SPEA2 obtiveram os melhores valores de SP.

- Espalhamento Máximo (M3). Representa a distância entre os valores extremos no espaço de objetivos das soluções achadas pelos algoritmos. Quanto maior este valor, maior área atingida do Pareto-Ótimo Local. O algoritmo SPEAModProp obteve neste caso o melhor resultado, seguido por SPEA2Mod e NSGAIIMod.

- Cobertura ($\mathcal{C}$). Esta métrica retorna o grau de dominância com que o Pareto-Ótimo Local domina localmente as soluções obtidas em cada execução dos três algoritmos. Na média, os resultados do algoritmo SPEAModProp foram dominados em menor proporção pelos Pareto-Ótimo Local, seguido dos algoritmos SPEA2Mod e NSGAIIMod.

A Figura 5.1 mostra o Pareto-Ótimo Local (representado com +). Esta representação é ilustrativa e não reflete o número de soluções nem os nichos formados. É possível que um ponto no gráfico represente várias soluções em nichos diferentes. Por exemplo:

```
---P-P--HEAGA-----  ----P-P--HEAGA-----
HEAGAPWGHEEPAWHEAE  HEAG-APWGHEEPAWHEAE
   Extensão :10        Extensão :10
   Qualidade :0        Qualidade :0
```

Ambas as soluções possuem a mesma extensão e qualidade, portanto são representadas pelo mesmo ponto, embora estejam em nichos diferentes pois a sua distância de Hamming é 4.

As Figuras 5.2, 5.3 e 5.4 mostram as soluções achadas pelos algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod, respectivamente, sobre o Pareto-Ótimo Local. Pode-se pensar que um maior número de pontos sobrepostos significa uma maior aproximação ao Pareto-Ótimo Local. Não obstante, devido ao fato que os pontos não refletem os nichos, a afirmação anterior não é correta. Para ter um referencial adicional, foi calculada a Taxa de Erro entre o Pareto-Ótimo Local e o Pareto-Ótimo Local de cada algoritmo. Os resultados são mostrados na Tabela 5.5

| Extensão | Taxa de Erro |
|---|---|
| SPEAModProp | 0.4548 |
| SPEA2Mod | **0.4301** |
| NSGAIIMod | 0.9778 |

**Tabela 5.5:** Taxa de Erro para SPEAModProp, SPEA2Mod e NSGAIIMod

Neste caso, o SPEA2Mod obteve uma Taxa de Erro levemente menor que o SPEAModProp. O NSGAIIMod obteve uma Taxa de Erro elevada devido ao reduzido número de soluções encontradas pelo algoritmo.

Como conclusão, cada métrica utilizada está relacionada com algum aspecto da performance dos algoritmos embora o uso de uma única métrica que reflita todos os aspectos do desempenho de um algoritmo atualmente é um assunto de pesquisa [Coello Coello, 2001]. O algoritmo SPEAModProp teve melhor desempenho nas métricas de Espalhamento Máximo ($M3$) e Cobertura ($\mathcal{C}$) em relação ao Pareto-Ótimo Local. SPEA2Mod teve melhores resultados com a Taxa de Erro. Em relação às métricas de Distância Geracional ($GD$) e no Espaçamento ($SP$), ambos os algoritmos tiveram resultados muito próximos; o que não permitem afirmar a superioridade de um sobre o outro. No entanto, o NSGAIIMod tive os piores resultados em todos os aspectos.

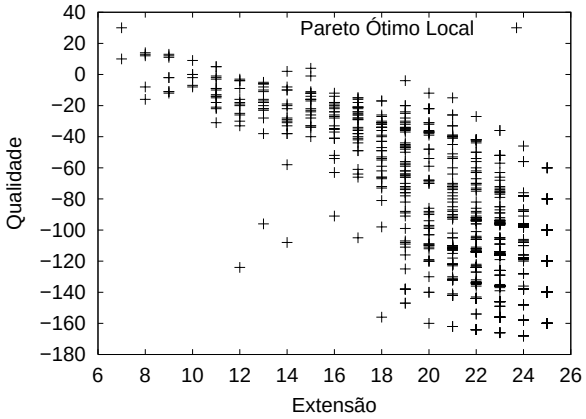



**Figura 5.1:** Pareto-ótimo Local

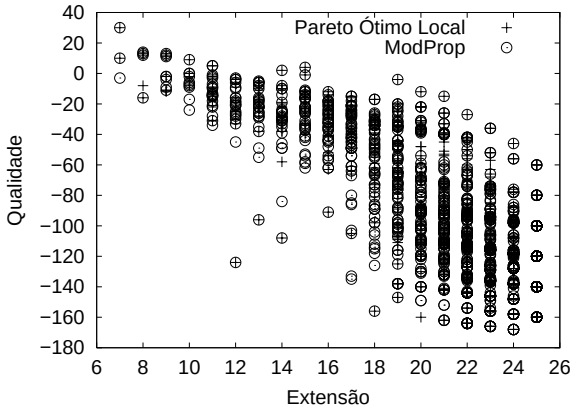



**Figura 5.2:** SPEAModProp vs. Pareto-Ótimo Local

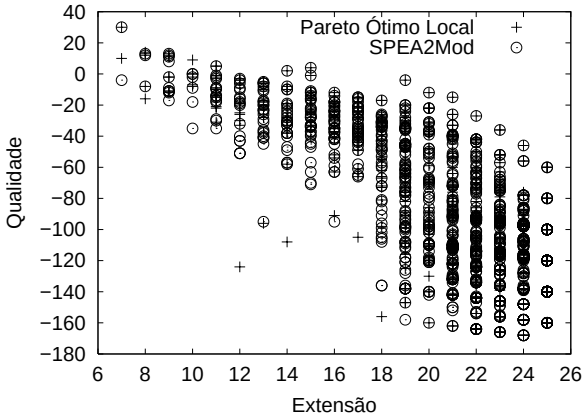



**Figura 5.3:** SPEA2Mod vs. Pareto-Ótimo Local

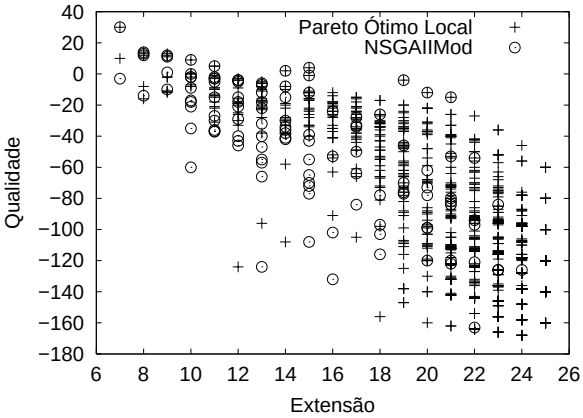



**Figura 5.4:** NSGAIIMod vs. Pareto-Ótimo Local

### 5.3.4 Critério de Resumo de Soluções

Existe um grande número de soluções fornecidas pelos três algoritmos. Tais soluções estão distribuídas em diversos nichos. O Critério de Resumo escolhe o representante de melhor qualidade por nicho. Por exemplo:

```
-----P-PHEAGA-----  ---P-P--HEAGA-----  ----P--PHEAGA-----
HEAGAPWGHEEPAWHEAE  HEAGAPWGHEEPAWHEAE  HEAGAPWGHEEPAWHEAE
Extensão :8         Extensão :10        Extensão :9
Qualidade:14        Qualidade:0         Qualidade:1
```

As três soluções estão no mesmo nicho pois a distância entre elas é de 3. Quando o Critério de Resumo é aplicado, a primeira solução é escolhida, eliminando as outras duas.

O resumo das soluções será aplicado iterativamente nos algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod. Os três algoritmos serão executados por 50 iterações adicionais nas quais será realizado apenas o deslocamento de buracos (*Shiftgap*) com uma probabilidade de 0.1. Estas iterações são necessárias cada vez que o operador de Deslocamento de Buracos permite melhorar as soluções. Por exemplo, em algumas execuções dos algoritmos SPEAModProp, SPEA2Mod, NSGAIIMod forneceram soluções como a seguinte:

```
PPHEA-G-A-----------
--HEAGAPWGHEEPAWHEAE
```

Deslocando os buracos das posições 8 e 6 da seqüência de busca, tem como resultado:

```
PPHEAGA-------------
--HEAGAPWGHEEPAWHEAE
```

Esta solução é igual da encontrada pelo algoritmo de programação dinâmica. Portanto este procedimento é uma busca local dentro da vizinhança para encontrar a solução de melhor qualidade.

Em outros termos, o Critério de Resumo está formulado como um algoritmo evolutivo de busca local que permite corrigir os resultados. Os resultados obtidos agrupados pela extensão para SPEAModProp, SPEA2Mod e NSGAIIMod são mostrados na Tabela 5.6.

| Extensão | Pareto-Ótimo Local | SPEAModProp | SPEA2Mod | NSGAIIMod |
|---|---|---|---|---|
| 7 | 2 | **4** | 3 | 2 |
| 8 | 2 | **3** | 2 | 3 |
| 9 | 4 | **7** | 4 | 1 |
| 10 | 1 | **3** | 3 | 2 |
| 11 | 4 | 4 | 8 | 5 |
| 12 | 4 | **8** | 8 | 3 |
| 13 | 8 | 17 | 19 | 6 |
| 14 | 7 | **14** | 12 | 6 |
| 15 | 14 | **26** | 23 | 5 |
| 16 | 19 | **30** | 6 | 3 |
| 17 | 19 | **29** | 14 | 5 |
| 18 | 13 | **18** | 14 | 4 |
| 19 | 12 | **19** | 8 | 2 |
| 20 | 4 | **9** | 3 | 5 |
| 21 | 8 | **11** | 8 | 3 |
| 22 | 10 | **20** | 5 | 2 |
| 23 | 9 | **10** | 10 | 3 |
| 24 | 9 | **15** | 3 | 0 |
| 25 | 3 | 4 | 6 | 0 |
| TOTAL | 152 | **251** | 159 | 60 |

**Tabela 5.6:** Resultados obtidos por SPEAModProp, SPEA2Mod e NSGAIIMod junto ao Pareto-Ótimo Local

Pode-se observar que o número de soluções decresceu para SPEAModProp e SPEA2Mod devido à aplicação do Critério e Resumo. Da mesma forma, o algoritmo NSGAIIMod reduziu o número de soluções embora em menor proporção em relação aos outros algoritmos. O conjunto de Pareto-Ótimo Local foi obtido aplicando o Critério de Resumo à união dos resultados dos três algoritmos. A Tabela 5.7 apresenta algumas das soluções obtidas.

```
--PPHEAGA---------    PPHEAGA-------------  ----------PP--HEAGA
HEAGAPWGHEEPAWHEAE    --HEAGAPWGHEEPAWHEAE  HEAGAPWGHEEPAWHEAE-
Extensão :7           Extensão :7           Extensão :9
Qualidade :-3         Qualidade :30         Qualidade :11

----PP--HEAG----A-    ---P--P--HEAGA-----
HEAGAPWGHEEPAWHEAE    HEA-GAPWGHEEPAWHEAE
Extensão :13          Extensão :11
Qualidade :-5         Qualidade :-2

PPHEA----GA---------  -----P-PHEAGA-----
--HEAGAPWGHEEPAWHEAE  HEAGAPWGHEEPAWHEAE
Extensão :11          Extensão :8
Qualidade :5          Qualidade :14
```

**Tabela 5.7:** Amostra dos alinhamentos obtidos por SPEAModProp, SPEA2Mod e NSGAIIMod após o Resumo

Para medir o desempenho dos algoritmos foram usadas as métricas da seção 5.3.3. Adicionalmente, foi calculado o número de nichos (NC) em cada execução dos algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod. Os resultados são mostrados na Tabela 5.8

| Alg/Métrica | | C(P*,P) | NC | M3 | SP | GD |
|---|---|---|---|---|---|---|
| SPEAModprop | Valor | 0.2034 | **110.0000** | **192.1456** | 0.3188 | **0.3349** |
| | Desvio | 0.0569 | 6.7165 | 5.6319 | 0.0848 | 0.0139 |
| SPEA2Mod | Valor | **0.1991** | 96.4000 | 173.5822 | **0.2959** | 0.3574 |
| | Desvio | 0.0282 | 12.1308 | 16.7513 | 0.0653 | 0.0210 |
| NSGAIIMod | Valor | 0.4422 | 21.1356 | 136.3791 | 1.8293 | 0.6221 |
| | Desvio | 0.0949 | 3.9650 | 24.4659 | 0.1283 | 0.0463 |

**Tabela 5.8:** Resultados das Métricas para os algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod após a aplicação do Critério de Resumo

### Interpretação dos Resultados

Os resultados obtidos em cada algoritmo são analisados em cada métrica revelando diferentes aspectos da performance dos resultados.

- Distância Geracional (GD). SPEA2Mod e SPEAModProp obtiveram os melhores valores. Embora SPEAModProp tenha obtido um resultado superior, pode-se dizer que o desempenho é equivalente ao SPEA2Mod nesta métrica. NSGAIIMod obteve resultado notavelmente inferior.

- Espaçamento (SP). Conforme à Tabela 5.8, SPEA2Mod obteve o melhor valor de $SP$, sendo levemente superior ao valor obtido por SPEAModProp. NSGAIIMod obteve novamente o pior resultado.

- Espalhamento Máximo (M3). O algoritmo SPEAModProp obteve neste caso o melhor resultado, seguido por SPEA2Mod e NSGAIIMod. Neste caso, a diferença entre SPEAModProp e SPEA2Mod é significativa, sendo que SPEA2Mod apresenta um Desvio Padrão maior, o que indica um comportamento mais irregular nas diferentes execuções.

- Número de Nichos (NC). Esta métrica indica a quantidade de nichos encontrados em cada execução, quanto maior o número de nichos mais diversas serão as encontradas. O algoritmo SPEAModProp obteve o melhor resultado com uma diferença significativa sobre SPEA2Mod. O menor Desvio Padrão em SPEAModProp indica um comportamento mais estável em relação a SPEA2Mod. NSGAIIMod encontrou um menor número de nichos comparado aos outros algoritmos.

- Cobertura ($\mathcal{C}$). Esta métrica foi calculada usando o Critério de Resumo, ou seja, para que uma solução domine a outra deve estar no mesmo nicho e ter uma qualidade superior. A dominância é do Pareto-Ótimo Local com relação às soluções encontradas em cada execução de SPEAModProp, SPEA2Mod e NSGAIIMod. Os resultados da Tabela 5.8 indicam que as soluções de SPEA2Mod foram proporcionalmente menos dominadas que as de SPEAModProp, não obstante esta diferença não ser significativa. As soluções de NSGAIIMod foram dominadas em uma proporção maior.

A Figura 5.5 mostra o Pareto-Ótimo Local. Neste caso, cada ponto corresponde a um nicho diferente. É possível que um ponto no gráfico represente várias soluções em nichos diferentes, entreando isto acontece em uma proporção pequena. As Figuras 5.6, 5.7 e 5.8 mostram as soluções achadas pelos algoritmos SPEAModProp, SPEA2 e NSGAII respectivamente sobre o Pareto-Ótimo Local. A Taxa de Erro entre o Pareto-Ótimo Local Total e Pareto-Ótimo Local de cada algoritmo é mostrado na Tabela 5.9

O SPEAModProp apresentou uma Taxa de Erro menor em relação a SPEA2Mod e NSGAIIMod. Este resultado pode ser deduzido também das Figuras 5.5, 5.6, 5.7 e 5.8 onde pode-se observar que SPEAModProp cobre uma maior "área"da superfície formada pelos pontos do Pareto-Ótimo Local.

| Extensão | Taxa de Erro |
|---|---|
| SPEAModProp | **0.342105** |
| SPEA2Mod | 0.631579 |
| NSGAIIMod | 0.927632 |

**Tabela 5.9:** Taxa de Erro para SPEAModProp, SPEA2Mod e NSGAIIMod

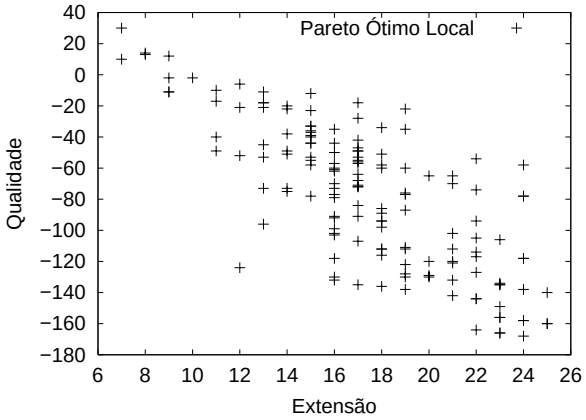



**Figura 5.5:** Pareto-ótimo Local

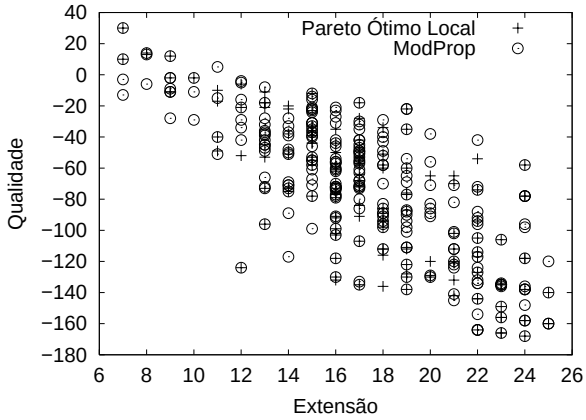



**Figura 5.6:** SPEAModProp vs. Pareto-Ótimo Local

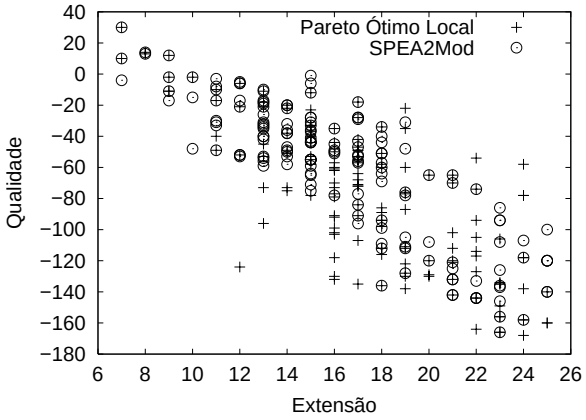



**Figura 5.7:** SPEA2Mod vs. Pareto-Ótimo Local

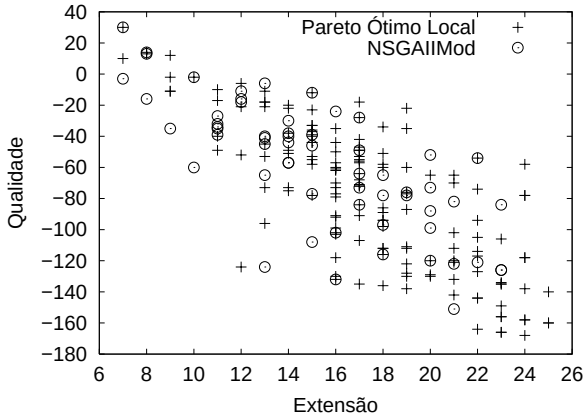



**Figura 5.8:** NSGAIIMod vs. Pareto-Ótimo Local

Em resumo, o algoritmo SPEAModProp teve melhor desempenho com as métricas de Distância Geracional, Número de Nichos, Espalhamento Máximo e Taxa de Erro ao Pareto-Ótimo Local. Enquanto que com as métricas de Cobertura e Espaçamento, o SPEA2 teve melhores resultados que o SPEAModProp, embora esta diferença não seja grande. Finalmente, NSGAIIMod obteve os piores resultados em todas as métricas.

### 5.3.5 Filtragem Final de Soluções

Por existir ainda um número considerável de soluções, uma filtragem final foi feita considerando os seguintes critérios:

**Definição 9** *Um intervalo está definido nas posições de começo e fim onde a seqüência de busca está posicionada.*

A seguir, vários exemplos de intervalos são mostrados:

```
PPHEAGA-------------  ---P--P--HEAGA-----
--HEAGAPWGHEEPAWHEAE  HEA-GAPWGHEEPAWHEAE
  Intervalo [1,7]       Intervalo [4,14]

 -----P-PHEAGA-----   ----------PP--HEAGA
 HEAGAPWGHEEPAWHEAE   HEAGAPWGHEEPAWHEAE-
  Intervalo [6,13]      Intervalo [11,19]

 -----P---PHEAGA---   -----P-PHE--AGA---
 HEAGAPWGHEEPAWHEAE   HEAGAPWGHEEPAWHEAE
  Intervalo [6,15]      Intervalo [6,15]
```

Se duas soluções existem no mesmo intervalo será eliminada aquela que apresenta uma qualidade menor, por exemplo as duas ultimas soluções do exemplo anterior existem no mesmo intervalo, mas a primeira seria eliminada.

**Definição 10** *O Porcentagem de Matchs é a razão entre número de caracteres coincidentes e comprimento da seqüência de busca.*

```
   PPHEAGA-------------          ---P--P--HEAGA-----
   --HEAGAPWGHEEPAWHEAE          HEA-GAPWGHEEPAWHEAE
Porcentagem de Matchs:  5/7  Porcentagem de Matchs:  4/7

    -----P-PHEAGA-----           ----------PP--HEAGA
    HEAGAPWGHEEPAWHEAE           HEAGAPWGHEEPAWHEAE-
Porcentagem de Matchs:  4/7  Porcentagem de Matchs:  4/7

    -----P---PHEAGA---           -----P-PHE--AGA---
    HEAGAPWGHEEPAWHEAE           HEAGAPWGHEEPAWHEAE
Porcentagem de Matchs:  2/7  Porcentagem de Matchs:  4/7
```

As soluções que apresentam um porcentagem de *matchs* maior que um valor predeterminado serão conservadas.

Os critérios de filtragem final foram aplicados diretamente nos resultados obtidos pelos três algoritmos. Esta filtragem de soluções não formam um conjunto Pareto-Ótimo Local devido a dominância local não foi considerada. Portanto, tem-se simplesmente um Conjunto Final de Soluções. A Tabela 5.10 mostra os resultados finais agrupados por extensão. Pode-se observar que algumas extensões não possuem soluções dado à baixa qualidade das soluções. A segunda coluna mostra os valores de qualidade obtidos por cada extensão no Conjunto Final de Soluções.

| Extensão | Conjunto Final | SPEAModProp | SPEA2Mod | NSGAIIMod |
|---|---|---|---|---|
| 7 | 2 | 2 | 2 | 1 |
| 8 | 2 | 2 | 2 | 2 |
| 9 | 2 | 3 | 3 | 1 |
| 10 | 2 | 1 | 3 | 1 |
| 11 | 2 | 3 | 4 | 0 |
| 12 | 1 | 3 | 3 | 0 |
| 13 | 2 | 2 | 3 | 1 |
| 14 | 1 | 2 | 3 | 0 |
| 15 | 1 | 3 | 3 | 0 |
| 16 | 0 | 3 | 4 | 0 |
| 17 | 1 | 1 | 3 | 0 |
| 18 | 1 | 1 | 1 | 0 |
| 19 | 1 | 2 | 0 | 0 |
| 20 | 0 | 1 | 0 | 1 |
| 21 | 1 | 1 | 2 | 0 |
| 22 | 1 | 1 | 1 | 0 |
| TOTAL | 20 | 31 | 37 | 7 |

**Tabela 5.10:** Resultados de SPEAModProp, SPEA2Mod e NSGAIIMod após o Filtragem Final de soluções.

Uma vez que os critérios aplicados no filtragem final não refletirem os conceitos de dominância de Pareto ou nichos, apenas foram aplicadas as métricas de Distância Geracional, Espaçamento e Espalhamento Máximo. Os resultados são mostrados na Tabela 5.11.

SEAModProp possui um melhor Espalhamento Máximo (M3) comparado com SPEA2Mod e NSGAIIMod. O algoritmo NSGAIIMod obteve um valor Distância Geracional menor ao obtido por SPEA2Mod e SPEAModProp. Isto ocorre por que seus resultados estão contidos no Conjunto Final. Porém, a maioria das soluções do Conjunto Final não foram achadas por NSGAIIMod. SPEA2Mod e SPEAModProp obtiveram valores próximos de Distância Geracional (GD). O valor de Espaçamento (SP) de SPEA2Mod é melhor que os de SPEAModProp e NSGAIIMod. É importante dizer que os critérios de Filtragem Final aplicados

| Alg/Métrica | | C(P2,P1) | M3 | SP | GD |
|---|---|---|---|---|---|
| SPEAModprop | Valor | 0,0943 | 79,7778 | 0,8453 | 0,7596 |
| | Desvio | 0,0398 | 8,5506 | 0,3130 | 0,0834 |
| SPEA2Mod | Valor | 0,1082 | 74,0236 | 0,6139 | 0,6915 |
| | Desvio | 0,0572 | 17,6918 | 0,2677 | 0,0514 |
| NSGAIIMod | Valor | 0,0308 | 64,3884 | 2,0048 | 0,6611 |
| | Desvio | 0,0506 | 26,5358 | 0,5080 | 0,1457 |

**Tabela 5.11:** Resultados das Métricas para os algoritmos SPEAModProp, SPEA2Mod e NSGAIIMod após o Filtragem Final

podem ser outros e, conseqüentemente, os resultados para as métricas são muito variáveis. As soluções finais obtidas podem ser vistas na Tabela 5.12.

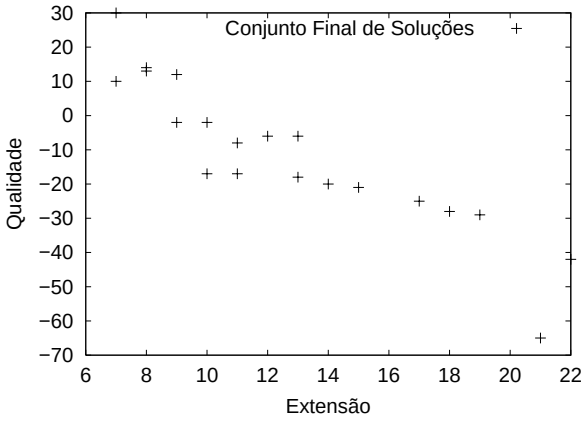



**Figura 5.9:** Pareto-ótimo Local

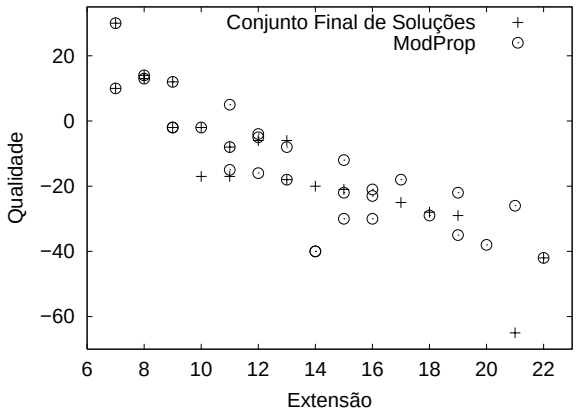



**Figura 5.10:** SPEAModProp

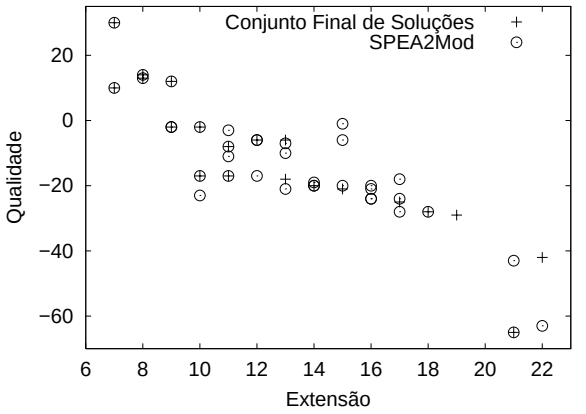



**Figura 5.11:** SPEA2Mod

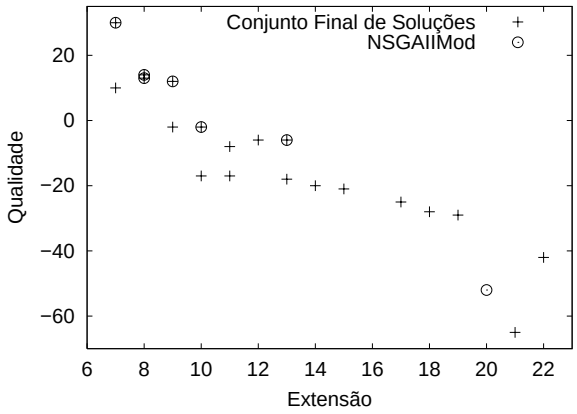



**Figura 5.12:** NSGAIIMod

Durante os experimentos realizados pode-se observar a paridade que existe entre os algoritmos SPEAModProp e SPEA2Mod conforme as métricas de desempenho utilizadas. Estas métricas revelam vários aspectos de qualidade de soluções como distância ao Pareto-Ótimo

| | | |
|---|---|---|
| -PP-----HE------AGA<br>HEAGAPWGHEEPAWHEAE-<br>Extensão :18<br>Qualidade :-28 | ---P--P--HEA----GA-<br>HEA-GAPWGHEEPAWHEAE<br>Extensão :15<br>Qualidade :-21 | ---PP-----HEAGA-----<br>HEA--GAPWGHEEPAWHEAE<br>Extensão :12<br>Qualidade :-6 |
| ------PPHE----AGA-<br>HEAGAPWGHEEPAWHEAE<br>Extensão :11<br>Qualidade :-8 | -------PPHE--AGA---<br>HEAGAPW-GHEEPAWHEAE<br>Extensão :9<br>Qualidade :-2 | -PP-----HE--AGA---<br>HEAGAPWGHEEPAWHEAE<br>Extensão :14<br>Qualidade :-20 |
| PPHEAGA-------------<br>--HEAGAPWGHEEPAWHEAE<br>Extensão :7<br>Qualidade :30 | ----PP--HEA----GA-<br>HEAGAPWGHEEPAWHEAE<br>Extensão :13<br>Qualidade :-6 | -----PP--HEAGA-----<br>HEAGA-PWGHEEPAWHEAE<br>Extensão :9<br>Qualidade :12 |
| P-----------P--H--EAGA<br>-HEAGAPWGHEEPAWHEAE---<br>Extensão :22<br>Qualidade :-42 | --P---P--------HEA-GA<br>HE-AGAPWGHEEPAWHEAE--<br>Extensão :19<br>Qualidade :-29 | -P--P-----H------EA-GA<br>H-EA-GAPWGHEEPAWHEAE--<br>Extensão :21<br>Qualidade :-65 |
| ---P--P--------HEAGA-<br>HEA-GAPWGHEEPAWHEA--E<br>Extensão :17<br>Qualidade :-25 | --PPHEAGA-------------<br>HE----AGAPWGHEEPAWHEAE<br>Extensão :7<br>Qualidade :10 | -------PPHE-AG--A-<br>HEAGAPWGHEEPAWHEAE<br>Extensão :10<br>Qualidade :-17 |
| -PPHE--A--GA---------<br>H---EAGAPWGHEEPAWHEAE<br>Extensão :11<br>Qualidade :-17 | ---P--P--HE--AGA---<br>HEA-GAPWGHEEPAWHEAE<br>Extensão :13<br>Qualidade :-18 | -----P-PHEAGA-----<br>HEAGAPWGHEEPAWHEAE<br>Extensão :8<br>Qualidade :14 |
| -----------PP-HEAGA<br>HEAGAPWGHEEPAWHEAE-<br>Extensão :8<br>Qualidade :13 | -----P-PHE--AGA---<br>HEAGAPWGHEEPAWHEAE<br>Extensão :10<br>Qualidade :-2 | |

**Tabela 5.12:** Conjunto de Soluções Finais

Local, Dominância, Espaçamento, Espalhamento e Taxa de Erro. Algumas medidas como o Espalhamento favorecem a SPEAModProp por uma boa margem embora as outras métricas forneçam valores similares para SPEAModProp e SPEA2Mod. Um fator que não foi mencionado até agora é o tempo de execução de cada algoritmo. SPEA2Mod é consideravelmente mais lento do que SPEAModProp devido ao cálculo das distâncias dos *k-vizinhos* para o valor de aptidão. NSGAIIMod teve um tempo de execução intermediário entre SPEAModProp e SPEA2Mod. É pertinente concluir que o SPEAModProp teve um bom comportamento com relação a duas versões dos algoritmos mais conhecidos da literatura.

Na seção seguinte, serão comparados os resultados obtidos mediante este algoritmo com as técnicas tradicionais de alinhamento de seqüências tratadas na seção 2.3 do Capítulo 2.

### 5.3.6 Comparação com as Técnicas Tradicionais

Os experimento da seção 5.3 foram executados usando os algoritmos heurísticos BLAST, FASTA e o algoritmo de Programação Dinâmica. Os resultados obtidos podem ser observados nas Tabelas 5.13, 5.14, e 5.15. FASTA e BLAST obtiveram uma solução (a solução de maior pontuação), enquanto os diversos modelos de Programação Dinâmica calcularam os alinhamentos global, semi-global e locais. A comparação feita foi simplesmente referencial, dado que estes algoritmos não foram desenvolvidos para avaliar alinhamentos com vários critérios, importando apenas o pontuação dos mesmos.

## 5.4 Comentários Finais

Neste capítulo, foi apresentado um modelo de Algoritmo Genético Multi-Objetivo para o Alinhamento de Seqüências Biológicas. Para isto, formulou-se o problema de Alinhamento de Seqüências Biológicas como um problema de Otimização Multi-Objetivo, usando os critérios de extensão e a qualidade do alinhamento. A seguir, o Modelo Proposto (SPEAModProp) foi detalhado em aspectos como operadores genéticos, representação de soluções, estratégia de seleção, nicho e elitismo.

Para efeitos comparativos, foram formulados dois modelos alternativos: o SPEA2Mod e o NSGAIIMod, baseados em dois dos modelos mais conhecidos da literatura atual. Em ambos os algoritmos foi adotado o critério de dominância local para o cálculo da aptidão das soluções.

Nos experimentos realizados usando duas seqüências de proteínas artificiais, os algoritmos SPEAModProp e SPEA2Mod produziram um grande número de soluções apresentando um desempenho muito similar, fato sustentado pela aplicação de métricas de rendimento aos resultados obtidos.

Posteriormente, os resultados foram reprocessados para obter soluções representantes por cada nicho (protótipos). Novamente, os algoritmos SPEAModProp e SPEA2Mod obtiveram desempenho similar com uma leve vantagem de SPEAModProp. Após isto, foram aplicados dois critérios de filtragem de soluções, um critério intervalar o outro baseado em porcentagem de coincidências, para todas as soluções obtidas em cada algoritmo.

Os resultados finais mostram uma grande variedade de formas em que uma seqüência pode ser alinhada sobre uma outra, o que pode ser visto também como a procura de uma

```
Query= SEQ1
 (7 letters)
>SEQ2
 Length = 18
 Score = 14.3 bits (34), Expect = 0.025
 Identities = 5/5 (100%), Positives = 5/5 (100%)
Query:  3 HEAGA 7
          HEAGA
Sbjct:  1 HEAGA 5
Lambda K H
 0.210 0.104 0.337
Gapped
Lambda K H
 0.193 0.0350 0.120
Matrix:  BLOSUM50
Gap Penalties:  Existence:  13, Extension:  2
Number of Hits to DB: 15
Number of Sequences:  0
Number of extensions:  1
Number of successful extensions:  1
Number of sequences better than 10.0:  1
Number of HSP's better than 10.0 without gapping:  1
Number of HSP's successfully gapped in prelim test:  0
Number of HSP's that attempted gapping in prelim test:  0
Number of HSP's gapped (non-prelim):  1
length of query:  7
length of database:  18
effective HSP length:  0
effective length of query:  28
effective length of database:  18
effective search space:  504
effective search space used:  504
T: 11
A: 40
X1:  3 ( 0.9 bits)
X2:  53 (14.8 bits)
X3:  89 (24.8 bits)
S1:  3 ( 4.2 bits)
S2:  3 ( 5.7 bits)
```

**Tabela 5.13:** Saída BLAST do alinhamento de seqüências

seqüência curta em uma seqüência mais longa, mostrando as diferentes posições onde podem ser encontradas similaridades em maior ou menor grau.

Comparado como as técnicas tradicionais do alinhamento de seqüências como BLAST, FAST e Programação Dinâmica, o algoritmo SPEAModProp mostrou uma maior variedade

```
 FASTA searches a protein or DNA sequence data bank
 version 3.4t10 Dec 12, 2001
Please cite:
 W.R. Pearson & D.J. Lipman PNAS (1988) 85:2444-2448
 seq1.txt, 7 aa
 vs seq2.txt library

     18 residues in     1 sequences
FASTA (3.44an0 Dec 2001) function [BL50 matrix (15:-5)] ktup: 2
 join: 44, opt: 32, open/ext: -10/-2, width:  16
The best scores are:                                          opt
SEQ2                                                (  18)   34
>>SEQ2                                                      (18 aa)
 initn:  34 init1:  34 opt:  34
Smith-Waterman score: 34;  100.000% identity (100.000% ungapped)
                           in 5 aa overlap (3-7:1-5)
SEQ1   PPHEAGA
         :::::.............
SEQ2     HEAGAPWGHEEPAWHEAE
                10
7 residues in 1 query   sequences
18 residues in 1 library sequences
 Scomplib [34t10]
 start: Wed Jan 15 22:03:33 2003 done: Wed Jan 15 22:03:47 2003
 Scan time:  0.000 Display time: 14.000
Function used was FASTA [version 3.4t10 Dec 12, 2001]
```

**Tabela 5.14:** Saída FAST do alinhamento de seqüências

| Alinhamento Global | Alinhamento Semi-Global |
|---|---|
| PPHEAGA----------<br>--HEAGAPWGHEEPAWHEAE<br>Qualidade : -16 | PPHEAGA<br>HEAGAPWGHEEPAWHEAE<br>Qualidade : 34 |
| Alinhamento Local | |
| HEAGA<br>HEAGAPWGHEEPAWHEAE<br>Qualidade : 34<br><br>HEAGA<br>HEAGAPWGHEEPAWHEAE<br>Qualidade : 18 | HEA<br>HEAGAPWGHEEPAWHEAE<br>Qualidade : 21 |

**Tabela 5.15:** Soluções de Programação Dinâmica

de soluções finais devido a aplicação de critérios de Otimização Multi-Objetivo, cuja característica é justamente brindar um conjunto de alternativas factíveis para um problema.

Capítulo

# 6

# Conclusões e Trabalhos Futuros

Neste trabalho apresentou-se o problema de alinhamento de Seqüências Biológicas e as principais técnicas para resolução desse problema usadas na atualidade. Essas técnicas enfatizaram a determinação das melhores soluções levando em conta um critério de pontuação das mesmas. Para esse trabalho foi adotado um enfoque diferente que considerou, além da pontuação das soluções, a extensão das mesmas. Em outras palavras, o problema de alinhamento de seqüências foi formulado como um problema de Otimização Multi-Objetivo, que permitiu encontrar um conjunto de soluções que representaram um compromisso entre os critérios de avaliação de soluções. Para encontrar o conjunto de soluções que forneceram alinhamentos com diferentes extensões e qualidades foi proposto um Algoritmo Evolutivo Multi-Objetivo que permitiu encontrar tanto as melhores soluções como a maior diversidade possível delas. Os resultados mostraram diversas soluções onde a seqüência de busca foi posicionada em diversos lugares da seqüência objetivo, o que forneceu valiosa informação sobre os possíveis alinhamentos entre duas seqüências. Dado o grande número de resultados obtidos, foi necessária a aplicação de critérios de resumo de soluções e uma filtragem final adicional que mostraram os alinhamentos mais relevantes.

Além disso, o Modelo Proposto foi comparado com outros modelos de Algoritmos Evolutivos Multi-Objetivo existentes na literatura, onde cada um deles foi executado várias vezes e tiveram calculadas suas métricas de performance. Os valores das métricas mostraram um bom desempenho do Modelo Proposto em relação aos outros modelos.

Em seguida, os experimentos foram rodados usando as técnicas tradicionais para alinhamento de seqüências e os resultados mostraram um número reduzido de soluções para cada caso. Isso foi devido ao fato de tais técnicas não terem sido desenvolvidas para essa finalidade. Foi perceptível também que o número de soluções achadas pelo Modelo Proposto foi significativamente superior, pois permitiu o conhecimento de alinhamentos interessantes que não foram possíveis obter mediante o uso das técnicas tradicionais.

Embora os resultados obtidos neste trabalho sejam encorajadores, existem vários vários aspectos importantes que merecem um estudo mais aprofundado tais como:

- Processamento Paralelo. Um problema de Otimização Multi-Objetivo pode ser resolvido com maior eficiência e possivelmente com maior eficácia paralelizando os algoritmos usados para resolvê-lo [Coello Coello et al., 2002]. Trabalhar com múltiplos processadores ou computadores implica um maior potencial de exploração do espaço de busca e uma redução do tempo de execução de um algoritmo. Durante o desenvolvimento deste trabalho os experimentos foram feitos com seqüências pequenas para facilitar a análise dos resultados, mas as seqüências reais normalmente são bem maiores. Esta grande quantidade de dados implica em um tempo de execução proibitivo para um algoritmo seqüencial, por isso o processamento paralelo é necessário. Existem vários procedimentos de um Algoritmo Evolutivo Multi-Objetivo que podem ser processados em paralelo, com operadores genéticos e cálculo de aptidão. Existem também vários modelos de algoritmos Evolutivos Paralelos como Modelo Mestre-Escravo, Modelo de Ilhas e Modelo de Difusão. [Coello Coello et al., 2002]. Todos estes aspectos poderão ser pesquisados em trabalhos futuros.

- Cálculo Automático de Parâmetros. É desejável que um algoritmo possa rodar eficientemente ajustando o menor número de parâmetros possível. Durante este trabalho os parâmetros de execução do algoritmo foram determinados experimentalmente tomando como base o trabalho de Romero Zaliz [Romero Zaliz, 2001]. O parâmetro de raio de nicho deveria ser calculado automaticamente pois o comportamento do Modelo Proposto é muito sensível ao mesmo. Um valor muito pequeno levaria a formação de uma grande quantidade de nichos com a conseqüente lentidão do processo. Contrariamente, um valor muito grande implica a possibilidade de sobrepor várias soluções interessantes em um mesmo nicho para depois alguma ser eliminada. Existem ainda estudos para ajuste automático deste parâmetro de forma dinâmica durante as iterações do algoritmo [Deb, 2001].

- Novas Métricas de Desempenho. Para comparar os resultados fornecidos pelos diferentes modelos avaliados foram adaptadas métricas que usam os conceitos de dominância global para que fosse incluído o conceito de dominância local. Porém, a natureza do

Pareto Ótimo Global e o Pareto Ótimo Local é diferente e, por tanto, pesquisas são necessárias nessa área a fim de achar métricas que sejam mais adequadas para esse trabalho.

- Estruturas de Dados Eficientes. Um dos aspectos que torna mais demorada a execução do algoritmo é o cálculo de dominância que deve ser realizado entre cada par de soluções. Este processo é usado para obter a aptidão das soluções e para atualizar os conjuntos externos onde são guardadas as soluções não dominadas. Nesta implementação foram usadas as estruturas de dados mais simples para estas tarefa. Existem na literatura exemplos de estruturas que melhoram a eficiência do cálculo da dominância como as Árvores de Dominância [Everson et al., 2002] e os KD-Tree [Mostaghim et al., 2002]. Incluir estas estruturas no algoritmo reduziriam significativamente o tempo de execução do mesmo.

- Novos Critérios de Resumo de Soluções. Embora tenham sido aplicados vários critérios para o resumo de soluções e filtragem de soluções irrelevantes, ainda ficaram algumas soluções de qualidade baixa. Para a apresentação de soluções finais seriam necessários ainda mais testes que incluíssem novas estratégias de filtragem de soluções.

Espera-se que este trabalho sirva como base para futuras pesquisas sobre a aplicação da otimização Multi-Objetivo e Algoritmos Evolutivos a outros problemas relacionados a Bioinformática.

# Referências Bibliográficas


Altschul, S. F., Gish, W., Miller, W., Myers, E. e Lipman, D. [1990], 'Basic local alignment search tool', *Journal of Molecular Biology* **215**, 403–410.

Coello Coello, C. A. [2001], A Short Tutorial on Evolutionary Multiobjective Optimization, *in* E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello e D. Corne, eds, 'First International Conference on Evolutionary Multi-Criterion Optimization', Springer-Verlag. Lecture Notes in Computer Science No. 1993, pp. 21–40.

Coello Coello, C. A. e Toscano Pulido, G. [2001], Multiobjective Optimization using a Micro-Genetic Algorithm, *in* L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon e E. Burke, eds, 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)', Morgan Kaufmann Publishers, San Francisco, California, pp. 274–282.

Coello Coello, C. A., Van Veldhuizen, D. A. e Lamont, G. B. [2002], *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York.

Corne, D. W., Jerram, N. R., Knowles, J. D. e Oates, M. J. [2001], PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization, *in* L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon e E. Burke, eds, 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)', Morgan Kaufmann Publishers, San Francisco, California, pp. 283–290.

Corne, D. W., Knowles, J. D. e Oates, M. J. [2000], The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization, *in* M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo e H.-P. Schwefel, eds, 'Proceedings of the Parallel Problem Solving from Nature VI Conference', Springer. Lecture Notes in Computer Science No. 1917, Paris, France, pp. 839–848.

Dayhoff, M., Schwartz, R. e Orcutt, B. [1978], A model of evolutionary change in proteins, *in* M. Dayhoff, ed., 'Atlas of Protein Sequence and Structure', Vol. 5, Springer-Verlag. Lecture Notes in Computer Science No. 1993, Washington D.C., pp. 345–352.

Deb, K. [2001], *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, New York.

Deb, K., Agrawal, S., Pratab, A. e Meyarivan, T. [2000], A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

Everson, R. M., Fieldsend, J. E. e Singh, S. [2002], Full Elite Sets for Multi-Objective Optimisation, *in* I. Parmee, ed., 'Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)', Vol. 5, Springer-Verlag, University of Exeter, Devon, UK, pp. 343–354.

Fonseca, C. M. e Fleming, P. J. [1993], Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *in* S. Forrest, ed., 'Proceedings of the Fifth International Conference on Genetic Algorithms', University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California, pp. 416–423.

Gen, M. e Cheng, R. [1997], *Genetic Algorithms and Engineering Design*, Wiley, New York.

Goldberg, D. E. [1989], *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, MA.

Goldberg, D. e Richardson, J. [1987], Genetic algorithms with sharing for multimodal function optimization, *in* 'Proceedings of hte First International Conference on Genetic Algorithms and Their Applications', pp. 471–483.

Haimes, Y., Lasdon, L. e Wismer, D. [1971], 'On a bicriterion formulation of the problems of integrated system identification and system optimization', *IEEE Transactions on Systems, Man, and Cybernetics* **1**(3), 296–297.

Hajela, P. e Lin, C. Y. [1992], 'Genetic search strategies in multicriterion optimal design', *Structural Optimization* **4**, 99–107.

Haupt, R. I. e Haupt, S. E. [1998], *Practical Genetic Algorithms*, Wiley, New York.

Henikoff, S. e Henikoff, J. [1991], Amino acid substitution matrices from protein blocks, *in* 'Proceedings of the National Academy of Sciences of the USA', Vol. 19, pp. 6565–6572.

Horn, J., Nafpliotis, N. e Goldberg, D. E. [1994], A Niched Pareto Genetic Algorithm for Multiobjective Optimization, *in* 'Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence', Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 82–87.

Ishikawa, M., Toya, T., Hoshida, M., Nitta, K., Ogiwara, A. e Kanehisa, M. [1993], 'Multiple sequence alignment by parallel simulated annealing', *Comput. Applic. Biosci.* **9**, 267–273.

Kim, J. e Cole, J. [1993], 'Alignment of possible secondary structures in multiple rna sequences using simulated annealing', *Comput. Applic. Biosci.* **12**, 259–267.

Kita, H., Yabumoto, Y., Mori, N. e Nishikawa, Y. [1996], Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm, *in* H.-M. Voigt, W. Ebeling, I. Rechenberg e H.-P. Schwefel, eds, 'Parallel Problem Solving from Nature—PPSN IV', Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, pp. 504–512.

Knowles, J. D. e Corne, D. W. [1999], The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation, *in* '1999 Congress on Evolutionary Computation', IEEE Service Center, Washington, D.C., pp. 98–105.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K. e Haussler, D. [1993], Hidden Markov Models in Computational Biology: Applications to Protein Modeling, Technical Report UCSC-CRL-93-32, University of California. citeseer.nj.nec.com/krogh93hidden.html.

Laumanns, M., Rudolph, G. e Schwefel, H.-P. [1998], A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study, *in* A. E. Eiben, M. Schoenauer e H.-P. Schwefel, eds, 'Parallel Problem Solving From Nature — PPSN V', Springer-Verlag, Amsterdam, Holland, pp. 241–249.

Laummanns, M. [2002], 'Pisa : A platform and programming language independent interface for search algorithms'. http://www.tik.ee.ethz.ch/pisa/spea2/spea2_documentation.txt. Data de Acceso: 24/11/2002.

Lecompte, O., Thompson, J. D., Plewniak, F., Thierry, J.-C. e Poch, O. [2001], 'Multiple alignment of complete sequences (macs) in the post-genomic era', *Gene* **270**, 17–30.

Mostaghim, S., Teich, J. e Tyagi, A. [2002], Comparison of Data Structures for Storing Pareto-sets in MOEAs, *in* 'Congress on Evolutionary Computation (CEC'2002)', Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 843–848.

Needleman, S. e Wunsch, C. [1970], 'A general method applicable to the search for similarities in the amino acid sequence of two proteins', *Journal of Molecular Biology* **48**, 443–453.

Notredame, C. e Higgins, D. G. [1996], 'Saga : sequence alignment by genetic algorithm', *Nucleic Acids Reseach* **24**(8), 1515–1524.

Pearson, W. R. e Lipman, D. J. [1978], Improved tools for biological sequence comparison, *in* 'Proceedings of the National Academy of Sciences', Vol. 85, pp. 2444–4448.

Pietrokovski, S. e Rubin, E. [2002], 'Advanced topics in bioinformatics', Web Course. Feinberg Graduate School of the Weizmann Institute of Science. http://bioportal.weizmann.ac.il/course/ATIB/. Data de acceso: 28/01/2003.

Rodrigues, M. R. [2001], 'Hidden markov models aplicados à biologia computacional', Tutorial Slides. http://www.inf.ufrgs.br/l̃uciana/limiteap.pdf.

Romero Zaliz, R. [2001], Reconocimiento y descripción de objetos complejos en biología molecular, Master's thesis, Departamento de Computación, Universidad de Buenos Aires.

Rudolph, G. [2001], Evolutionary Search under Partially Ordered Fitness Sets, *in* 'Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)', ICSC Academic Press: Millet/Sliedrecht, pp. 818–822.

Sankoff, D. [1975], 'Minimal mutation trees of sequences', *SIAM Journal on Applied Mathematics* **28**, 35–42.

Schaffer, J. D. [1985], Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *in* 'Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms', Lawrence Erlbaum, pp. 93–100.

Schott, J. R. [1995], Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Setubal, J. e Meidanis, J. [1997], *Introduction to Computational Molecular Biology*, Brooks, United States.

Smith, T. e Waterman, M. [1981], 'Identification of common molecular subsequences', *Journal of Molecular Biology* **147**, 195–197.

Srinivas, N. e Deb, K. [1994], 'Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms', *Evolutionary Computation* **2**(3), 221–248.

Veldhuizen, D. A. V. [1999], Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Waterman, M. S. [1996], *Introduction to Computational Biology*, Chapman Hall, United States.

Zaha, A. [2000], *Biologia Molecular Básica*, 2 edn, Mercado Aberto, Brasil.

Zhang, C. e Wong, A. K. C. [1997], 'A genetic algorithm for multiple molecular sequence alignment', *Comput. Applic. Biosci.* **13**(6), 565–581.

Zitzler, E., Deb, K. e Thiele, L. [2000], 'Comparison of Multiobjective Evolutionary Algorithms: Empirical Results', *Evolutionary Computation* **8**(2), 173–195.

Zitzler, E., Laumanns, M. e Thiele, L. [2001], SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E. 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.

Apêndice

# A

# Revisão de Otimização

## A.1 Definições Básicas

Um problema de otimização (minimização) não restrito é definido da forma:

$$\begin{array}{ll} \text{minimizar} & f(\mathbf{x}) \\ \text{restrito a:} & \mathbf{x} \in X, \end{array} \tag{A.1}$$

onde $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$, $f : \mathbb{R}^n \mapsto \mathbb{R}$ e $X$ é um conjunto aberto (usualmente $X = \mathbb{R}^n$).

Um problema de otimização restrito está definido da seguinte forma:

$$\begin{array}{lll} \text{maximizar/minimizar} & f(\mathbf{x}) & \\ \text{restrito a:} & g_i(\mathbf{x}) \leq 0 & i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0 & i = 1, \ldots, l \\ & \mathbf{x} \in X, & \end{array} \tag{A.2}$$

onde $g_1, \ldots, g_m, h_1, \ldots, h_l : \mathbb{R}^n \mapsto \mathbb{R}$.

A função $f(\mathbf{x})$ é denominada função objetivo, as $g_i(\mathbf{x})$ são restrições de desigualdades e as $h_i(\mathbf{x})$ são restrições de igualdades.

Seja $g(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})]^T : \mathbb{R}^n \mapsto \mathbb{R}^m$ e $h(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_l(\mathbf{x})]^T : \mathbb{R}^n \mapsto \mathbb{R}^l$. A equação A.2 pode ser reescrita como:

$$\begin{array}{ll} \text{minimizar} & f(\mathbf{x}) \\ \text{restrito a:} & g(\mathbf{x}) \leq 0 \\ & h(\mathbf{x}) = 0 \\ & \mathbf{x} \in X \end{array} \tag{A.3}$$

**Definição 11** *Um ponto extremo de uma função $f(\mathbf{x})$ define um máximo ou mínimo de $f$.*

**Definição 12** *Um ponto $\mathbf{x}$ é mínimo se $f(\mathbf{x}+\mathbf{h}) \geq f(\mathbf{x})$ para toda $\mathbf{h} = [h_1, \ldots, h_n]^T$ tal que $|h_j|$ seja arbitrariamente pequena para toda $j$.*

**Definição 13** *Um ponto $\mathbf{x}$ é máximo se $f(\mathbf{x}+\mathbf{h}) \leq f(\mathbf{x})$ para toda $\mathbf{h} = [h_1, \ldots, h_n]^T$ tal que $|h_j|$ seja arbitrariamente pequena para toda $j$.*

## A.2 Condições de Otimalidade para problemas de otimização não restritos

**Teorema 4** *Seja $f$ diferenciável em $\mathbf{x}^*$. Se $\mathbf{x}^*$ é um ponto extremo, então $\nabla f(\mathbf{x}^*) = 0$.*

O Teorema 4 define a *condição necessária de otimalidade de primeiro ordem* para um problema de otimização não restrito.

**Definição 14** *Uma matriz $M_{n\times n}$ é denominada **positiva semidefinida**, se $\mathbf{x}^T M \mathbf{x} \geq 0$, para toda $\mathbf{x} \in \mathbb{R}^n, x \neq \mathbf{0}$.*

**Definição 15** *A matriz Hessiana $H$ para uma função $f : \mathbb{R}^n \mapsto \mathbb{R}$ está definida como:*

$$\begin{bmatrix} \frac{\partial^2 f}{dx_1^2} & \frac{\partial^2 f}{dx_1 dx_2} & \cdots & \frac{\partial^2 f}{dx_1 dx_n} \\ \frac{\partial^2 f}{dx_2 dx_1} & \frac{\partial^2 f}{dx_2^2} & \cdots & \frac{\partial^2 f}{dx_2 dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{dx_n dx_1} & \frac{\partial^2 f}{dx_n dx_2} & \cdots & \frac{\partial^2 f}{dx_n^2} \end{bmatrix}$$

**Teorema 5** *Seja $f$ duas vezes diferenciável em $\mathbf{x}^*$, se $\mathbf{x}^*$ é um mínimo local, então $\nabla f(\mathbf{x}^*)$ e $H(\mathbf{x}^*)$ é positiva semidefinida .*

O Teorema 5 define a *condição necessária de otimalidade de segundo ordem* para um problema de otimização não restrito.

## A.3 Condições de Otimalidade para problemas de otimização restritos

**Teorema 6** *Seja* $\mathbf{x}^*$ *uma solução para o problema A.3. Se* $\mathbf{x}^*$ *é um mínimo local, então existem vetores* $\mathbf{u} \in \mathbb{R}^m$ *e* $\mathbf{v} \in \mathbb{R}^l$ *tal que:*

$$\begin{aligned} \nabla f(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^T \mathbf{u} + \nabla h(\mathbf{x}^*)^T \mathbf{v} &= 0 \\ \mathbf{u} &\geq 0 \\ \mathbf{u}_i g_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, m. \end{aligned}$$

As condições do Teorema 6 são conhecidas como as condições de Kuhn-Tucker para problemas de otimização restritos.