Capítulo 1

Algoritmos Evolutivos

Yvan Jesus Tupac Valdivia Universidade Católica de San Pablo, Peru, ytupac@ucsp.pe.

Este capítulo apresenta os conceitos fundamentais da computação evolutiva base para a otimização, os princípios básicos da Computação Evolutiva são: princípio com base na natureza e evolução darwiniana, paradigmas da computação evolutiva, algoritmos genéticos canônicos, modelos evolutivos para a otimização numérica e modelos evolutivos otimização combinatória.

1.1 Otimização e Heurística

1.1.1 Otimização

Um dos princípios fundamentais da natureza está buscando um estado ideal. Este princípio é observado a partir do microcosmo, quando os átomos tentam formar ligações de elétrons de baixa energia [Pauling1960]; ao nível molecular é observada durante o congelamento quando moléculas são convertidos em corpos sólidos para encontrar estruturas cristalinas de energia ideais. Em ambos os casos, estes processos são levados a esses estados de mínima energia apenas por leis físicas. Por outro lado, há o princípio biológico da *sobrevivência do mais apto* [Spencer1960] que, juntamente com a evolução biológica, levam uma melhor adaptação de espécies ao seu meio ambiente. Neste caso, uma condição local ótima é uma espécie que domina todos os outros ao seu redor. O *Homo Sapiens* tem alcançado este nível, dominando formigas, bactérias, moscas, baratas e todos as classes de criaturas existentes

em nosso planeta. Ao longo da nossa história os seres humanos, viemos à procura da perfeição em muitos aspectos. Por um lado, queremos alcançar a máxima bem estar fazendo o menor esforço em um aspecto econômico, busca-se maximizar as vendas e rentabilidade com o menor custo possível. Então, podemos dizer que a otimização é um aspecto da ciência, estendendo-se até a vida diária [Neumaier2006]. A partir dessas ideias, podemos inferir que é importante generalizar e compreender que, por trás desses fenômenos existe um formalismo matemático que estuda toda esta área: *otimização global*, que é um ramo da matemática aplicada e análise numérica cujo foco é claro ... Otimização. O objetivo de uma otimização global é encontrar os melhores elementos possíveis $\mathbf{x}^*$ de um conjunto $\mathbb{X}$ que segue um critério $F = \{f_1, f_2, \cdots, f_n\}$. Estes critérios são expressas como funções matemáticas $f(x)$ na área de otimização são muitas vezes chamados de *funções objetivo* ou funções de custo, os quais são definidos a seguir.

Definição 7.1.1 Função objetivo: Uma função objetivo $f : \mathbb{X} \to Y$, onde $Y \subseteq \mathbb{R}$ é uma função matemática para ser otimizado. A imagem Y da função objetivo é normalmente um subconjunto de números reais, ou seja, $(Y \subseteq \mathbb{R})$. $\mathbb{X}$, domínio da função f, se denomina de *espaço de problema* podendo ser representado (computacionalmente) por qualquer tipo de elemento, embora principalmente como listas ou vetores n-dimensionais, etc. Cabe destacar que as funções objetivo não se limitam a ser expressões matemáticas mas que podem ser compostas de algoritmos complexos ou equações que para ser calculado precisa de várias simulações. Assim, a otimização global deve compreender toda a técnica, analítica e não-analítica que pode ser usado para encontrar os melhores elementos $\mathbf{x} \in \mathbb{X}$ de acordo com a função $f \in F$. Em suma, a otimização é pesquisar e encontrar a solução ou soluções ótimas para um determinado problema, tendo em conta certas restrições. Como exemplos podemos citar:

- Aumentar a rentabilidade de um negócio.
- Reduzir custos, multas ou perda.
- Aumentar a eficácia de um determinado processo.

Para montar um problema qualquer da vida real como um problema de otimização, você deve identificar as seguintes entidades:

1. O problema: suas características, limitações e variáveis,
2. As variáveis de decisão $\mathbf{x} = x_1, \cdots, x_n$ do problema, cujos valores influenciem a solução,

3. A função $f(\mathbf{x})$ que calcula a qualidade da solução - *função objetivo*,
4. Um método, algoritmo ou heurística de busca de soluções,
5. O espaço discreto ou contínuo $\mathbb{X}$ de soluções válidas para o problema,
6. Os recursos computacionais necessários para: implementar o processo, a função de avaliação (ou simulação), tratamento de características do problema e selecionar a melhor solução.

1.1.2 Definições em Otimização

Como mencionado antes, o objetivo do *problema de otimização* é encontrar as soluções **ótimas** para um dado problema, ou seja, a melhor solução possível. Então vale a pena definir as características que produzem uma solução *ótima*.

Definição 7.1.2 Espaço de problema: O espaço de problema para um problema de otimização, denotado por $\mathbb{X}$, é o conjunto contendo todos os elementos $\mathbf{x}$ que poderiam ser a solução a ser encontrada.

Na maioria das vezes, o espaço $\mathbb{X}$ está sujeito a:

1. *Restrições lógicas* que indicam elementos x, que pode não ser soluções, como a divisão por zero ou uma raiz quadrada de um número negativo.
2. *Restrições práticas* que nos limitam o espaço de problema em questão de tecnologia. Um exemplo são os tipos de variável de ponto flutuante `float` ou `double` que possuem precisão limitada.

Definição 7.1.3 Candidato a solução: Um candidato a solução $\mathbf{x}$ é um elemento válido do conjunto $\mathbb{X}$ para um determinado problema de otimização.

Definição 7.1.4 Espaço de Soluções: O espaço de soluções $\mathbb{S}$ é definido como a união de todas as soluções de um problema de otimização.

$$\mathbf{x}^* \subseteq \mathbb{S} \subseteq \mathbb{X} \tag{1.1}$$

Como se mostra na Equação *1.1*, o espaço de soluções contém (que pode ser igual) o conjunto ótimo global de $\mathbf{x}^*$. Podem haver soluções válidas $\mathbf{x} \in S$ que não fazem parte de $\mathbf{x}^*$, o que geralmente acontece em otimização com restrições.

Definição 7.1.5 Espaço de pesquisa: O espaço de pesquisa de um problema de otimização, indicado por $\mathbb{G}$, é o conjunto de todos os elementos elementos $\mathbf{g} \in \mathbb{G}$ que pode ser processada pelos operadores de pesquisa. Assim $\mathbb{G}$ é uma codificação de um espaço de problema $\mathbb{X}$.

1.1.3 Funções de único objetivo

Quando um único critério é otimizado $f(x)$, um ideal é um máximo ou mínima, dependendo se você está maximizando ou minimizando. Por exemplo, em problemas do mundo real (transformação), procura geralmente *minimizar* vezes, custos ou perdas para um determinado processo. Por outro lado, tratando-se de negócios, procura *maximizar* a rentabilidade ou o valor econômico. A Figura *1.1* mostra uma função f definida num espaço X bidimensional com elementos de $\mathbf{x} = (x_1, x_2) \in \mathbb{X}$. Se destaca no gráfico o ótimo local e global. Um ótimo global é ótimo em toda conjunto, enquanto um ótimo local é ideal apenas em um subespaço de X.

Definição 7.1.6 Máximo local: Seja uma função de um alvo $f : \mathbb{X} \to \mathbb{R}$, um máximo local é definido $\hat{\mathbf{x}} :\in \mathbf{X}$ como um valor de entrada que atenda a $f(x_l) \geq f(x)$ para todos os x em um bairro de x_l.

Se $f : \mathbb{X} \subseteq \mathbb{R}^n$ podemos dizer que:

$$\forall \bar{x}_l,\ \exists \epsilon > 0 : f(\bar{x}_l) \leq f(x), \qquad \forall x \in \mathbb{X},\ |x - \bar{x}_l| < \epsilon \tag{1.2}$$

Definição 7.1.7 Mínimo local: Seja uma função de um objetivo $f : \mathbb{X} \to \mathbb{R}$, um mínimo local é definido $\bar{x}_l \in \mathbb{X}$ como um valor de entrada que atenda $f(\bar{x}_l) \leq f(x)$ para todos x em uma vizinhança de $\bar{x}_l$.

Se $f : \mathbb{X} \subseteq \mathbb{R}^n$ podemos dizer que

$$\forall \bar{x}_l,\ \exists \epsilon > 0 : f(\bar{x}_l) \leq f(x), \qquad \forall x \in \mathbb{X},\ |x - \bar{x}_l| < \epsilon \tag{1.3}$$

Definição 7.1.8 Ótimo local: Seja uma função de um alvo $f : \mathbb{X} \to \mathbb{R}$, um ótimo local é definido $x_l^* \in \mathbb{X}$ como um valor que está em conformidade com sendo um máximo local ou um mínimo local.

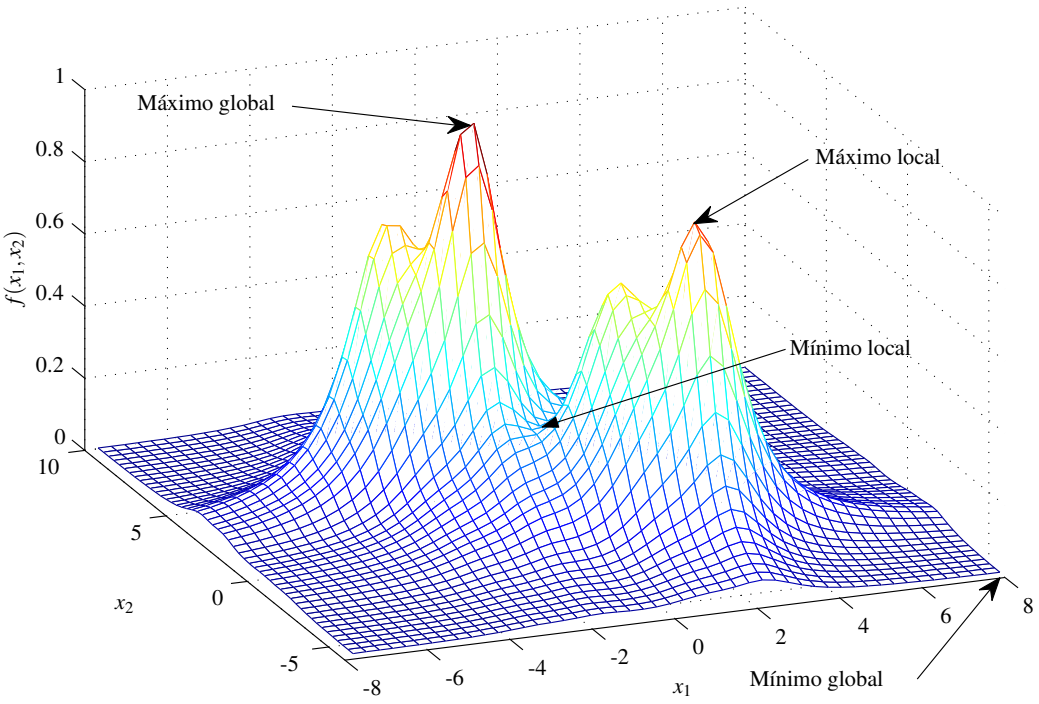

Figura 1.1: Exemplo de função $f(x_1, x_2)$ com ótimo local e global.

Definição 7.1.9 Máximo global: Seja uma função de um objetivo $f : \mathbb{X} \to \mathbb{R}$, em um máximo global é definido $\hat{\mathbf{x}} \in \mathbb{X}$ como um valor de entrada que atenda a $f(x) \geq f(x),\ \forall x \in X$

Definição 7.1.10 Mínimo global: Seja uma função de um objetivo $f : \mathbb{X} \to \mathbb{R}$, um mínimo global é definido $\bar{\mathbf{x}} \in \mathbb{X}$ como um valor de entrada que atenda a $f(\bar{\mathbf{x}}) \leq f(x),\ \forall x \in \mathbb{X}$.

Definição 7.1.11 Ótimo global: Seja uma função de um objetivo $f : \mathbb{X} \to \mathbb{R}$, um ótimo global é definido $\mathbf{x}^* \in \mathbb{X}$ como um valor de entrada que atenda para ser um valor máximo ou mínimo global.

É comum encontrar em um máximo global ou mínimo, inclusive dentro o domínio de uma função $\mathbb{X}$ unidimensional $f : \mathbb{X} \subseteq \mathbb{R}^n \to \mathbb{R}$. Um exemplo desta situação é a função coseno $\cos(x)$, onde $\mathbf{x} = \{x\}$, que tem valores globais máximos em $\hat{\mathbf{x}}$ quando $\hat{\mathbf{x}} = 2i\pi$ e valores mínimos globais $\bar{\mathbf{x}}$ quando $\bar{\mathbf{x}} = (2\pi + 1)\pi$. A solução correta seria um conjunto $\mathbf{x}^*$ de entradas ideais e sem o mínimo ou o máximo isolado. Além disso, o significado de *ótima*

depende do problema:

- Em otimização com um objetivo, um máximo ou mínimo global.
- Em otimização multi-objetivo existem várias estratégias para definir a ótima.

Definição 7.1.12 Melhor conjunto: O melhor conjunto é definido $\mathbf{x}^*$ como o conjunto que contém todos os melhores elementos.

Eles são geralmente múltiplos e muitas vezes, infinitas melhores soluções; mas dadas as limitações da computação, é possível encontrar um subconjunto soluções ótimas finito. Assim, é feita uma distinção entre o conjunto ótimo $\mathbf{x}^*$ ideal e o conjunto X sub-ótimo, o qual pode conter uma ótimo global. Este conjunto X é o que um algoritmo de otimização real pode encontrar. Então, podemos dizer que um algoritmo de otimização tem como tarefa:

1. Encontrar soluções que sejam o melhor possível para o problema e,
2. o qual, por sua vez, são tão diferentes uns dos outros.

1.2 Otimização Clássica

Em otimização clássica ou otimização numérica visamos encontrar o valor das variáveis univariadas de uma função de x que maximiza ou minimiza uma determinada função $f(x)$. Este problema de otimização genérico é definido como se segue:

Deixe a função

$$\begin{array}{rcl} f : \mathbb{X} & \mapsto & \mathbb{R} \\ \mathbf{x} & \mapsto & f(\mathbf{x}) \end{array} \tag{1.4}$$

O problema de otimização é encontrar

$$Y^* = \max(\min)\{f(x)\} \tag{1.5}$$

sujeito às seguintes condições:

$$\begin{array}{l} g_1(\mathbf{x}) \leq 0, \quad i = 1, \ldots, p \\ h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, n \end{array} \tag{1.6}$$

onde:

$\mathbb{X} \subseteq \mathbb{R}^n$ é o espaço do problema que é um subespaço de $\mathbb{R}^n$ que contêm os pontos possíveis, ou seja, pontos que satisfazem as restrições questão.

$\mathbf{x} \in \mathbb{X}$ é um ponto no espaço de problema definido como $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ cujos componentes $x_1, x_2, \ldots, x_n$ são chamados de variáveis de decisão do problema.

$f(\mathbf{x})$ é a função de objetivo, que é o que está a ser otimizado.

Definição 7.1.13 Restrições de Domínio Definido como os limites

$$a_i \leq x_i \leq b_i, \quad \forall i = 1, \ldots, n$$

relacionada com os valores mínimos e máximos das variáveis x_i permitidos no espaço X.

Definição 7.1.14 Restrições de igualdade definidas como o conjunto de tamanho p

$$g_i(\mathbf{x}) = 0, \quad \forall i = 1, \ldots, p$$

Definição 7.1.15 Restrições de desigualdade definida como o conjunto com tamanho q

$$h_j(\mathbf{x}) = 0, \quad \forall j = 1, \ldots, q$$

Convexidade

Definição 7.1.16 Espaço Convexo: O espaço de problema $\mathbb{X}$ é dito convexo se, para cada elemento $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}$ e todos os $\alpha \in [0, 1]$ vale que

$$f(\alpha \mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1 - \alpha) f(\mathbf{x}_2) \tag{1.7}$$

A interpretação da Equação *1.7* é: $\mathbb{X}$ é convexa se dois quaisquer pontos $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}$, o segmento de linha que une esses pontos também pertence ao conjunto. Em outras palavras, se $\mathbb{X}$ é convexo pode-se ir a partir de qualquer ponto de partida para qualquer ponto final em linha reta, sem sair do espaço. Os conjuntos da Figura **??** são convexos enquanto que os na Figura **??** não são.

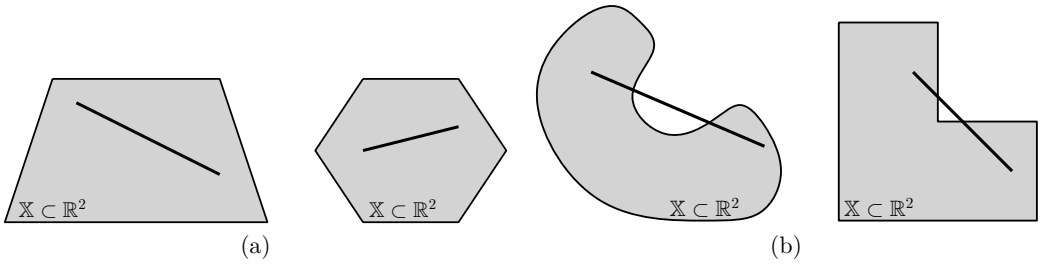

Figura 1.2: Espaços convexos e não convexos.

O objetivo de qualquer técnica de otimização é encontrar o ótimo global de qualquer problema. Infelizmente, só em casos muito limitados que podemos garantir a convergência para o ótimo global. Por exemplo, para problemas com espaço de pesquisa $\mathbb{X}$ convexos, as condições de Kuhn-Tucker [KuhnTucker1951] são necessárias e suficientes para assegurar a otimização, em geral, de um ponto.

Em problemas de otimização não lineares, estas condições não são suficientes. Portanto, neste caso, qualquer técnica pode ser usada para encontrar ótimos locais sem garantia de convergência para o ótimo global. Podemos apenas garantir esta convergência por meio de métodos exaustivos ou que existe tempo infinito. Uma classe especial de problemas de otimização que são igualmente muito interesses é o caso em que as variáveis de decisão são discretas, isto é, o vetor $\mathbf{x}$ consiste de valores $x_i \in \mathbb{N}$ e $\mathbf{x}$ é um produto cartesiano ou uma permutação de valores x_i. Tais problemas conhecidos como otimização combinatória e programação inteira, são de grande interesse na área de Ciência da Computação cujo mais conhecido exemplo é o problema do Caixeiro Viajante (Travelling Salesman Problem - TSP).

1.2.1 Técnicas de Otimização clássicas

Existem muitas técnicas muito conhecidas e aplicadas para resolver problemas de otimização, desde que cumpram certas características específicas, tal que $f(\mathbf{x})$ é uma função linear, ou seja unimodal[1] ou que as restrições são lineares.

A importância de se conhecer, a existência pelo menos, de uma dessas técnicas é se o problema a ser resolvido é ajustado para as exigências que impõem, não necessário o uso de heurística. Por exemplo, se a função é linear, o método

[1] Ter um único máximo ou mínimo é, portanto, ter o ótimo global.

Simplex sempre continuará a ser a opção mais viável.

Otimização Linear - Método Simplex A idéia básica da otimização Linear é buscar uma solução válida para melhorar a função objetivo tomando como solução inicial válida. O problema de programação linear é para minimizar o produto interno:

$$\min z = \mathbf{c}^T\mathbf{X} \tag{1.8}$$

sujeito às seguintes condições:

$$\begin{array}{rcl} \mathbf{Ax} & = & \mathbf{b} \\ \mathbf{x} & \geq & 0 \end{array} \tag{1.9}$$

onde: $\mathbf{A}_{m\times n}$; $r(\mathbf{A}) = m$; $\mathbf{b} \in \mathbb{R}^m$ e $\mathbf{c} \in \mathbb{R}^n$

Dado o conjunto $\mathcal{S} = \{\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$, qualquer ponto $\mathbf{x}_i \in \mathcal{S}$ é uma combinação linear convexa de seus pontos finais (ou vértices) mais um combinação linear positiva de suas direções extremas (arestas).

Otimização não linear Para otimização não linear, existem métodos diretos, como busca aleatória e métodos indiretos, tais como método do gradiente conjugado [singiresu1996]. Uma das principais limitações das técnicas clássicas é precisamente o requerer de informações que nem sempre estão disponíveis. Por exemplo, os métodos de gradiente necessitam que se calcule a primeira derivada da função objetivo. Outros métodos, como o de Newton precisam da segunda derivada. Portanto, se a função objetivo não é diferenciável, esses métodos não podem ser aplicados. Em muitos casos, no mundo real, não há nem mesmo uma função objetivo explícita.

Método *Steepest Descend* (Descida mais íngreme) Um exemplo clássico é o método de técnicas de otimização de descida íngreme ou descida mais íngreme, originalmente proposto por Cauchy em 1847. A idéia deste método é escolher um ponto $\mathbf{x}_i$ qualquer do espaço de pesquisa e se mover ao longo da direção da descida mais íngreme (Direção de ∇f_i). Para encontrar o valor ideal. Algoritmo 1 descreve este método.

Algoritmo 1 Algoritmo descida mais íngreme

escolher um ponto $\mathbf{x}_1$
$i = 1$
calcule ∇f_i
encontre o sentido $\mathbf{s}_i = -\nabla f_i = -\nabla f(\mathbf{x}_i)$
determinar o aumento ótimo λ_i^* na direção $\mathbf{s}_i$
calcular $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i^* \mathbf{s}_i = \mathbf{x}_i - \lambda_i^* \nabla f_i$
$i = i + 1$
$\mathbf{x}_{t+1}$ será ótimo.

Método de Fletcher-Reeves (Gradiente Conjugado) Que foi inicialmente proposto por Hestenes & Stiefel em 1952 como uma forma de sistemas de equações lineares derivadas das condições de resolver de um estacionário quadrático. Pode ser visto como uma variante do método da mais íngreme descida, que utiliza o gradiente de uma função para encontrar a mais promissora direção. O Algoritmo 2 descreve o método Fletcher-Reeves.

Algoritmo 2 Algoritmo do Gradiente Conjugado

escolher um ponto arbitrário $\mathbf{x}_1$
calcular a direção de busca $\mathbf{s}_1 = -\nabla f(\mathbf{x}_1) = -\nabla f_1$
encontrar $\mathbf{x}_2 = \mathbf{x}_1 + \lambda_1^* \mathbf{s}_1$, onde $\lambda *_1$ é o passo ótimo na direção $\mathbf{s}_i$
$i = 2$
compute $\nabla f_i = \nabla f(\mathbf{x}_i)$
calcule $\mathbf{s}_i = -\nabla f_i + \frac{|\nabla f_i|^2}{|\nabla f_{i-1}|^2} \mathbf{s}_{i-1}$
calcule $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i^* \mathbf{s}_i = \mathbf{x}_i - \lambda_i^* \nabla f_i$
calcular λ_i^*
calcular o novo ponto $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i^* \mathbf{s}_i$
$i = i + 1$
$\mathbf{x}_{t+1}$ será ótimo.

Existem outras técnicas que constroem soluções parciais para um problema. Como exemplos temos a Programação Dinâmica [Programação Bellman57] e método *Branch & Bound*.

Em conclusão, podemos dizer que, se a função a ser otimizada se encontra definida algebricamente, é importante para tentar resolvê-lo usando técnicas clássicas antes de atacar com todas as heurísticas.

1.2.2 Técnicas de Otimização heurísticas

Heurísticas Em otimização Clássica é assumido que os problemas devem cumpri certos requisitos que garantem a convergência para o ótimo global. No entanto, a maioria dos problemas do mundo real não satisfazem tais requisitos. Mesmo muitos destes problemas não podem ser resolvidos utilizando um algoritmo de tempo polinomial determinado utilizando computadores nísticos (conhecido como problemas NP, NP- Difícil, NP-completos) nos quais o melhor algoritmo conhecido exige tempo exponencial. De fato, em muitas aplicações práticas, é mesmo possível dizer que não há uma solução eficiente.

Quando confrontados com tais grandes espaços de busca como o problema do caixeiro viajante (*Traveling Salesman Problem* - TSP), e que também os algoritmos mais eficiente para a solução do problema requerem tempo exponencial, é óbvio que as técnicas de pesquisa clássica e otimização são insuficientes. Ou seja, é quando nos voltamos para a heurística.

Definição 7.1.17 Heurística: A palavra heurística é derivada do grego *heuriskein*, que significa para encontrar ou descobrir.

O significado do termo tem variado historicamente. Alguns utilizaram o termo como um antônimo de algorítmico.

É chamado heurístico um processo que pode resolver um problema determinado, mas não oferece nenhuma garantia de sucesso.

As heurísticas foram uma área de destaque nas origens da Inteligência Artificial. Hoje, o termo é frequentemente usado como um adjetivo, referindo qualquer técnica que melhora o desempenho na solução média de um problema, mas não necessariamente melhora o desempenho no pior dos casos [Stuart2002]. Uma definição mais precisa e adequada é fornecida por [Reeves1993].

> "A heurística é uma técnica que procura boas soluções (ou seja, quase ideal) a um custo computacional razoável, mas sem garantia de ótimabilidade ou viabilidade do mesmo. Em alguns casos, pode determinar o quão perto a solução ideal é particularmente viável."

Definição 7.1.18 Metaheurísticas: Uma Metaheurística é um método que é aplicado para resolver problemas genéricos. Ele combina as funções

objetivas ou heurísticas, de uma forma eficiente e abstrata que normalmente não dependem da estrutura do problema.

Exemplos de técnicas heurísticas e metaheurísticas aplicadas para problema de otimização são os seguintes:

Busca Tabu A busca Tabu (Pesquisa Tabu) [GloverLaguna1998] é realmente uma metaheurísticas, porque é um procedimento que deve ser anexado a outras técnicas, já que ele não funciona por si só. A busca tabu usa uma memória para orientar a busca, de modo que algumas soluções recentes examinadas são *armazenadas* e levadas como tabu (proibido), quando for tomar decisões sobre o próximo ponto de pesquisa. A busca tabu é deterministica, mas você pode adicionar elementos probabilísticos.

Simulated Annealing Baseia-se no arrefecimento de cristais [Kirkpatrick1983]. O algoritmo requer uma temperatura inicial e uma função de variação da temperatura. Tal função variação é crucial para o bom desempenho do algoritmo e sua definição é, portanto, extremamente importante. Este é um algoritmo de busca local probabilística.

Seu princípio é baseado no uso do algoritmo de Metropolis que, usando a Simulação de Monte Carlo calcula a mudança de energia que ocorre durante o arrefecimento de um sistema físico.

Definição 7.1.19 Algoritmo de Metropolis: Ele gera uma perturbação e a mudança de energia é calculada, se ela diminui, o novo estado é aceito, caso contrário, a aceitação deve ser objeto de um sorteio com a probabilidade da Equação *1.10*:

$$P(\partial E) = e^{-\frac{\partial E}{kt}} \tag{1.10}$$

Algoritmo 3 representa o processo de *Simulated Annealing*.

Algoritmo 3 Algoritmo Simulated Annealing (f)

Entrada: f: a função objetivo a minimizar-se.
Dado: k_{max}: Número máximo iterações.
Dado: e_max: Energia máxima.
Dado: $\mathbf{x}_{new}$ o novo elemento criado.
Dado: x_0 o melhor elemento conhecido.
Saída: $\mathbf{x}$ o melhor elemento encontrado.
$\mathbf{x} \leftarrow \mathbf{x}_0$, $e \leftarrow E(x)$, $\mathbf{x}_{best} \leftarrow \mathbf{x}$, $e_{best} \leftarrow e$
$k = 0$, $k < k_{max}$, $e > e_{max}$.
encontrar $x_{new} = \text{vizinho}(x)$
$e_{new} \leftarrow E(\mathbf{x}_{new})$
$P(e, e_{new}, T(k/k_{max}) > U(t)$
$\mathbf{x} \leftarrow \mathbf{x}_{new}$
e $\leftarrow e_{new}$
$e_{new} < e_{best}$
$\mathbf{x}_{best} \leftarrow \mathbf{x}_{new}$, $e_{best} \leftarrow e_{new}$
$k = k + 1$

***Hill Climbing* (Subida de encosta)** O algoritmo de subir a colina ou encosta [Stuart2002] é uma técnica simples de busca e otimização local aplicado a uma função f de um único objetivo em um espaço de problema X. Por iterações a partir de um ponto inicial $\mathbf{x} = \mathbf{x}_0$, que geralmente é a melhor solução conhecida para o problema, se obtém um novo descendente $\mathbf{x}_{new}$ na vizinhança de $\mathbf{x}$. Se o novo indivíduo $\mathbf{x}_{new}$ é melhor do que o antecessor $\mathbf{x}$, ele o substitui e assim por diante. Este algoritmo não volta atrás ou transporta qualquer registro histórico (embora estes e outros anexos são suscetíveis de serem incorporado). É importante ressaltar que, o *hill Climbing* visa maximizar ou minimizar a função $f(x)$, onde $\mathbf{x}$ é discretizado, ou $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{N}^p$. Pode-se dizer que a subida de montanha é uma maneira simples de pesquisa para na do gradiente, assim você pode facilmente ser pego em um ótimo local. O algoritmo 4 representa o *Hill Climbing*

Algoritmo 4 Algoritmo *hill Climbing* (f)

f: a função objetivo de minimizar,
n: número de iterações.
Dado: $\mathbf{x}_{new}$ o novo elemento criado.
Dado: $\mathbf{x}_0$ o melhor elemento conhecido.
Saída: $\mathbf{x}_{best}$ elemento encontrado.
$i = 0$, $\mathbf{x} = \mathbf{x}_0$
Avalie $f(\mathbf{x})$.
Encontre um $\mathbf{x}_{new} = \text{vizinho}(\mathbf{x})$.
Avalie $\mathbf{x}_{new}$
$f(\mathbf{x}_{new}) > f(\mathbf{x})$
$\mathbf{x} = \mathbf{x}_{new}$
$i = i + 1$, $i \geq n$

1.3 Noções básicas de Algoritmo Evolutivo

1.3.1 Algoritmos evolutivos

Definição 7.2.1 Algoritmos evolutivos (EA) São algoritmos metaheurísticos com base em uma população de indivíduos que empregam mecanismos biologicamente inspirados como mutação, recombinação, a seleção natural e sobrevivência do mais apto para ir ajustando de forma iterativa ou refinando um conjunto de soluções.

Uma vantagem de algoritmos evolutivos sobre outros métodos de otimização é a sua característica de caixa preta, ou seja, têm pouco conhecimento e poucos pressupostos sobre a função objetivo a ser otimizada. Mesmo a definição da função objetivo exige menos conhecimento da estrutura espaço que o problema de modelar uma heurística viável para o problema. Além disso, algoritmos evolutivos têm uma qualidade de resposta consistente a vários tipos de problemas.

Assim, quando se trata de Computação Evolutiva e Algoritmos Evolutivos, devemos entender que há um conjunto de técnicas e metodologias que os compõem, todos com inspiração biológica na evolução Neo-Darwinista.

1.3.2 Conceitos utilizados em Computação Evolutiva

Após a definição *7.2.1*, vários conceitos devem ser bem entendidos já que eles são comumente usados em Computação Evolutiva, descrito abaixo.

Definição 7.2.2 Individuo É uma proposta de solução na população, sem qualquer alteração. Aqui denominada como $\mathbf{x}$.

Definição 7.2.3 Cromossomo Refere-se a uma estrutura de dados contendo uma série de parâmetros de projeto (ou genes). Esta estrutura se pode armazenar computacionalmente de várias maneiras: cadeias de bits, *array* de números inteiros ou números reais. Neste trabalho um cromossomo é denotado como $\mathbf{g}$.

Definição 7.2.4 Gene Essa é uma subseção de um cromossomo que geralmente codifica o valor de um dos parâmetros do problema, o gene que codifica o i-ésimo parâmetro é indicado por g_i.

Definição 7.2.5 População Esse é o conjunto de cromossomos que serão tratados no processo evolutivo. Uma população é denotado como $\mathbf{X}$.

Definição 7.2.6 Geração É uma iteração consiste em calcular a medida de adequação de todos os indivíduos de uma população $\mathbf{X}$ existente em seguida, obter a próxima população a partir de um processo de operações de seleção genética e reprodução. Para uma geração t sua população é de $\mathbf{X}_t$.

Definição 7.2.7 Genótipo É a codificação usada no cromossomo, para os parâmetros do problema a ser resolvido. Por exemplo, temos: binários, ternários, inteiros, reais, o conjunto de valores possíveis genótipo torna-se o espaço de busca $\mathbb{G}$.

Definição 7.2.8 Fenótipo É o resultado da descodificação de um cromossomo, ou seja, os valores obtidos passam da representação genética $\mathbf{g} \in \mathbb{G}$ (Fenótipo) ao problema de espaço $\mathbf{x} \in \mathbb{X}$.

Definição 7.2.9 Função de avaliação Que é a medida de qualidade do individuo no ambiente. Por exemplo, com $f(\mathbf{x}) = x^2$ sendo a função de avaliação e $\mathbf{g}_k \rightarrow \mathbf{x}_k = \{x\} = 9$, então, a avaliação do indivíduo é $f(9) = 81$.

Definição 7.2.10 Aptidão É uma transformação g aplicada à função avaliação $f(\mathbf{x})$ para medir a possibilidade de um indivíduo $\mathbf{x}_k \in \mathbf{X}$ tem de se reproduzir quando a seleção é aplicada. Em muitos casos, a função de aptidão coincide com a função de avaliação, ou seja, $g(f(\mathbf{x})) = f(\mathbf{x})$. Note que uma avaliação individual $f(\mathbf{x}_k)$ não depende da avaliação de outros indivíduos da população $f(\mathbf{x}_j) \in \mathbf{X},\ j \neq k$, mas aptidão g de um indivíduo k é definida com respeito aos outros membros da população.

Definição 7.2.11 Alelo Que é cada valor possível que um gene poderá adquirir. Estas dependem do espaço de busca definido para o genótipo G e dependendo da codificação utilizada. Se a codificação binária é utilizado, em seguida, o espaço de busca $\mathbb{G} \equiv \mathbb{B}^l$ alelos permitindo $a_i \in \{0, 1\}$.

Definição 7.2.12 Operador de Reprodução É qualquer mecanismo que influencia a forma como a informação genética é transferida de pais para crianças. Esses operadores possuem duas categorias conhecidas:

- Operadores de cruzamento (sexual)
- Operadores de Mutação (assexuada)

Além disso, em alguns outros modelos mais genéricos são usados outros operadores, tais como:

- Operadores Panmíticos (um pai que se reproduz com múltiplos parceiros)
- Operadores usando 3 ou mais pais.

Definição 7.2.13 Elitismo O elitismo é um mecanismo adicional nos algoritmos evolutivos, que garante que o mais apto cromossomo $\mathbf{x}_{best} \in \mathbf{X}_t$ é transferida para a próxima geração $\mathbf{X}_{t+1}$ sem depender da seleção ou reprodução.

$$\mathbf{x}_{best} \in \mathbf{X}_t \rightarrow \mathbf{X}_{t+1} \tag{1.11}$$

Elitismo garante que a melhor adequação de uma população (individuo $\mathbf{x}_{best} \in \mathbf{X}_t$) nunca será pior do que uma geração t para a próxima $t+1$. Não é suficiente para encontrar o ótimo global, mas foi demonstrada ser uma condição necessária para assegurar a convergência para o ótimo de um algoritmo genético [Rudolph1994].

Princípio baseado na Natureza Um algoritmo evolutivo é uma abstração de processos e princípios estabelecidos instituídos pelo Darwinismo e Neo-Darwinismo junto ao princípio *goal-driven* (Impulsionada por objetivos) [Hauw1996]. Você também pode dizer que o espaço de busca $\mathbb{G}$ é uma abstração do conjunto de possíveis cadeias de DNA da natureza, jogando cada elemento $\mathbf{g} \in \mathbb{G}$ o papel de genótipos naturais.

Assim, pode-se ver o espaço $\mathbb{G}$ como um *genoma* e os elementos $\mathbf{g} \in \mathbb{G}$ como genótipos.

Da mesma forma qualquer coisa viva que sendo uma *instância* de seu genótipo formada durante a *embriogênese*, um candidato à solução (ou fenótipo $\mathbf{x}$ no espaço de problema $\mathbb{X}$ é também uma instância de seu genótipo foi obtido por uma função de mapeamento $\gamma : \mathbf{g} \mapsto \mathbf{x}$ (também conhecida como codificação), e sua aptidão é calculada de acordo com as funções objetivo que estão sujeitas à otimização e dirigem a evolução para um objetivo específico (*goal-driven*).

Ciclo básico de um algoritmo evolutivo Um processo evolutivo, em geral, pode ser simulado computacionalmente usando os seguintes mecanismos:

1. Uma forma de codificar as soluções $\mathbf{x}$ em estruturas $\mathbf{g}$ que é reproduzido formando uma população $\mathbf{G}_0$ inicial gerada aleatoriamente.

2. O mapeamento de aptidão $h(\cdot)$ dependente de $\mathbf{x}$ e sua avaliação $f(\mathbf{x})$

3. Um mecanismo de seleção com base na aptidão.

4. Operações que atuam sobre os indivíduos codificados $\mathbf{g}_i \in \mathbf{G}$ para os reproduzir.

Estes mecanismos são uma ordem de marcha tal como mostrado na Figura *1.3*.

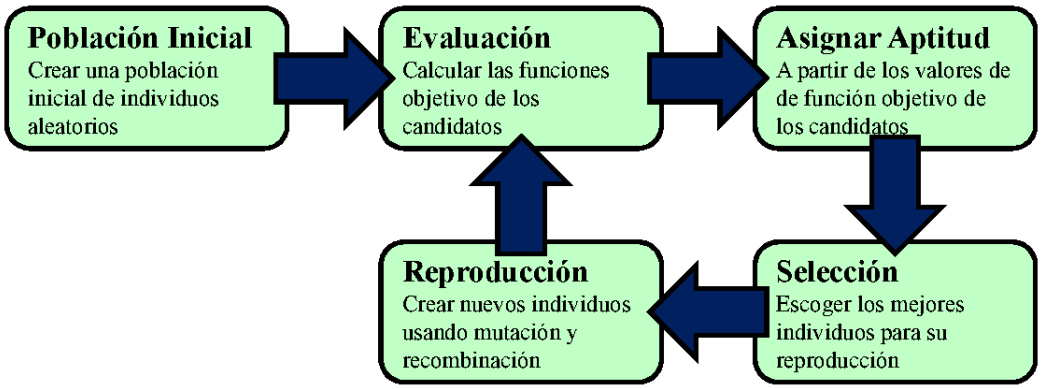

Figura 1.3: O ciclo básico de um algoritmo evolutivo.

1.3.3 Paradigmas da Computação Evolutiva

Apesar de não ser fácil distinguir as diferenças entre os diferentes tipos de algoritmos evolutivos atualmente, pode-se distinguir três principais paradigmas da computação evolutiva [Back1997]

- Programação Evolutiva
- estratégias evolutivas
- Algoritmos Genéticos

Além disso, há dois paradigmas: *Learning Classifier Systems* e Programação Genética, que são bastante próximos de Algoritmos Genéticos e Programação Evolutiva respectivamente. Cada um destes paradigmas originados de forma independente e com motivações muito diferentes. A Figura *1.4* ilustra uma classificação dos principais paradigmas que conformam com a família de algoritmos evolutivos.

Em seguida, se analisam rapidamente os principais paradigmas e mais a diante todos os detalhes de Algoritmos Genéticos.

Programação Evolutiva É uma concepção inicial da evolução simulada destinada a resolver problemas, especialmente a predição [Fogel65]. A técnica, chamada *Programação Evolutiva* [Fogel1966] foi basicamente fazer a evolução de autômatos de estados finitos, que foram expostos a uma série de símbolos de entrada (a atmosfera), na esperança de que em algum momento sejam capazes de prever futuras sequências de símbolos que receberiam. Fogel usou uma *função recompensa* indicando o quão bom foi um

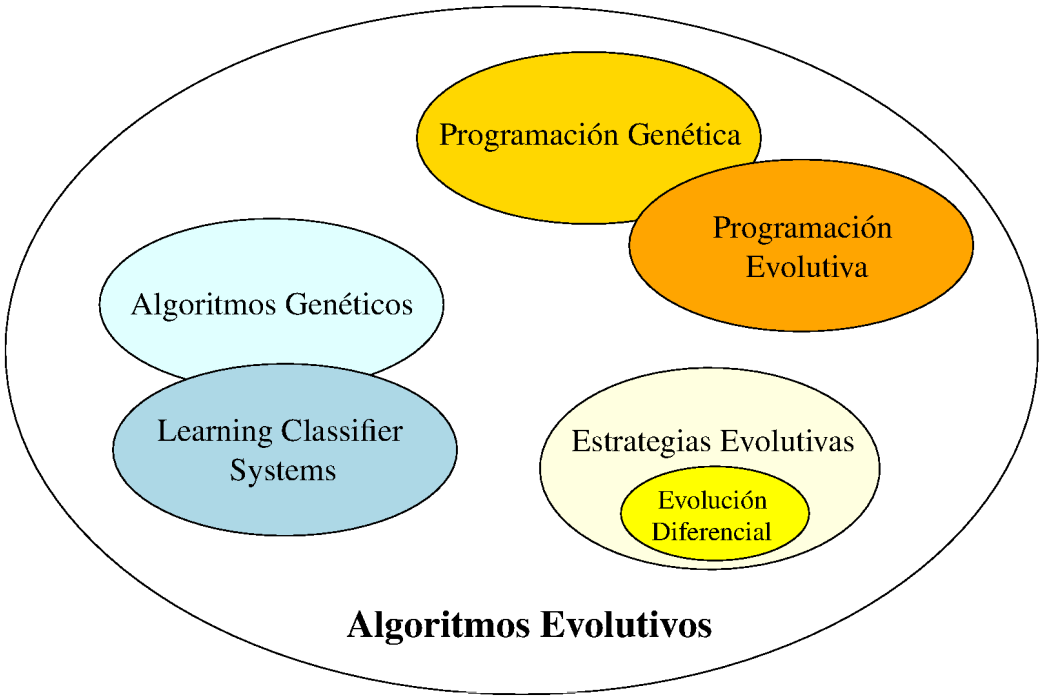

Figura 1.4: Familia de Algoritmos Evolutivos.

certo autômato para prever um símbolo, e usou um operador de mutação para fazer mudanças em transições e estados do autômato que tenderiam a torná-los mais adequados para sequências de símbolos preditos.

Essa técnica não considera a utilização de um operador de recombinação sexual porque procura modelar o processo evolutivo ao nível da espécie e não no nível dos indivíduos.

Programação evolutiva foi originalmente aplicada a problemas de previsão, controle automático, identificação de sistemas e teoria dos jogos, entre outros. Provavelmente a programação evolutiva foi a primeira técnica com base em evolução aplicada aos problemas de predição, além de ser a primeira a usar codificação de comprimento variável (o número de estados nos autômatos podem variar após a conclusão de uma mutação), além de ser uma das primeiras tentativas de simular a co-evolução.

Na programação evolutiva inteligência é vista como um comportamento adaptativo [Fogel1966], onde é dada mais importância às ligações de comportamento entre pais e filhos, em vez de operadores específicos. O algoritmo básico de programação evolutiva é como se segue:

Algoritmo 5 Algoritmo básico da programação evolutiva

$t = 0$ gerar uma população inicial $\mathbf{X}_t$
CFin = falso **mutar** $\mathbf{X}_t$
avaliar $\mathbf{X}_t$
selecionar elementos de $\mathbf{X}_t$
$t = t + 1$

Assim, a programação evolutiva é uma abstração da evolução à nível de espécies, onde todo mundo é uma espécie e por isso não são necessários operadores de recombinação, como espécies diferentes não podem se cruzar. A seleção está usando probabilística (por exemplo, torneio estocástico).

Estratégias evolutivas As estratégias evolutivas foram desenvolvidos em [Rechenberg73] como uma otimização heurística baseada na idéia de adaptação e evolução, com a intenção inicial de resolução de problemas hidrodinâmicos de alta complexidade. Estes problemas consistiam de experimentação em um túnel de vento para optimizar a forma de um tubo curvo, minimizar a resistência encontrada entre uma placa de ligação e otimizar a estrutura de um bocal intermitente de duas fases.

A descrição analítica desses problemas era impossível e, claro, suas soluções usando métodos tradicionais, tais como gradiente também eram. Este Ingo Rechenberg impelido a desenvolver um método de configurações aleatórias discretas inspirados no mecanismo de mutação que existe na natureza. Os resultados destas experiências são apresentados pela primeira vez no Instituto Hidrodinâmica da Universidade [Fogel98].

Nos dois primeiros casos (o tubo e placa), Rechenberg prosseguiu com mudanças aleatórias em determinadas posições das articulações e no terceiro problema passou a trocar, adicionar ou remover segmentos de bico. Sabendo que na natureza pequenas mutações ocorrem com maior frequência que grandes, Rechenberg decidiu fazer essas mudanças com base em uma distribuição binomial com uma variância fixada. O mecanismo básico destes primeiros experimentos era criar uma mutação, ajustar as juntas e segmentos de bicos de acordo com elas, levar a análise correspondente e determinar o quão bom era a solução. Se isso era melhor do que o seu predecessor, então passou a ser utilizado como uma base para o próximo experimento. Assim, nenhuma informação foi exigida, no montante de melhorias ou deteriorações que foram feitas. Esta técnica tão simples deu lugar a inesperadamente bons resultados para os três problemas em questão.

Características As estratégias evolutivas têm as seguintes características:

1. Os candidatos à solução são vetores $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n, \mathbf{x} = (x_1, \ldots, x_n)$ para eles não se aplicam qualquer codificação, ou seja, $\mathbb{G} = \mathbb{X}$. Assim o espaço de solução é definido como $\mathbb{X} \subseteq \mathbb{R}^n$. Isto significa que tanto o espaço de busca, como o problema do espaço são expressos computacionalmente com variáveis de números reais (ponto flutuante).

2. A mutação e seleção são os principais operadores sendo menos usual o cruzamento.

3. A mutação é explorar os elementos x_i do vetor $\mathbf{x}$ e ir substituindo-os por um número obtido a partir de uma distribuição normal $N(x_i, \sigma_i^2)$. Isto significa que o item é mutado usando uma distribuição normal multivariada $N(\mathbf{x}, \Sigma)$.

4. Existe um critério para atualizar o valor de σ_i^2 usando autoadaptação.

Algoritmo $(1+1)$**-EE** A versão original do $(1+1)$-EE, usava apenas um pai e filho. O filho competiu com o pai, e o melhor permaneceu na geração, esse tipo de estratégia de seleção é determinista e com característica extinta, já que os piores indivíduos têm probabilidade zero de seleção e simplesmente desaparecem.

A mutação no modelo $(1 + 1)$-EE para a geração t é obtida aplicando a seguinte equação:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{N}(0, \Sigma) \tag{1.12}$$

onde $\mathbf{N}(0, \Sigma)$ é uma variável aleatória que segue uma distribuição de Gauss multivariada com média $\mu = 0$ e matriz de covariância σ.

Σ é a matriz de covariância com elementos $\sigma_i \sigma_j = 0, \forall i \neq j$, ou seja, a matriz de covariância é diagonal, isso significa que são usadas distribuições normais independentes para a mutação de cada componente x_i de $\mathbf{x}$. O algoritmo 6 ilustra a estratégia $(1 + 1)$-EE.

Algoritmo 6 Algoritmo $(1+1)$-EE

$k, m,\ t = 0,\ p_s = 0$
inicializa $\mathbf{x}_t = (x_1, \ldots, x_n)$
Aleatoriamente **avalia** $f(\mathbf{x}_t)\ t \leq m$
mutar $\mathbf{x}_{t-mut} = \mathbf{x}_t + N(0, \Sigma_t)$
avaliar $\mathbf{x}_{t-mut}\ \mathbf{x}_t = \text{melhor}(\mathbf{x}_{t-mut}, \mathbf{x}_t)$
$\mathbf{x}_t = \mathbf{x}_{t-mut}$
$p_s = p_s + 1,\ t = t + 1,\ t \mod k = 0$

$$\Sigma_t = \begin{cases} \Sigma_t / C & \text{se } p_s/k < 1/5 \\ c\Sigma_t & \text{se } p_s/k > 1/5 \\ \Sigma_t & \text{se } p_s/k = 1/5 \end{cases}$$

$p_s = 0,\ \Sigma_t = \Sigma_{t-1}$

O próprio Rechemberg estendeu a estratégia inicial, introduzindo o conceito de população na estratégia denominada $(\mu + 1)$-EE, onde existem μ antecessores e um único filho, que pode substituir o pior dos ancestrais da população (seleção extintiva).

Estratégias $(\mu + \lambda)$-EE / (μ, λ)-EE Mais tarde, Schwefel [Schwefel77] propôs ter prole múltipla estabelecendo estratégias $(\mu + \lambda)$-EE e (μ, λ)-EE, cuja diferença é a metodologia de seleção:

- Em $(\mu+\lambda)$-EE se juntam os conjuntos de μ antepassados e de λ descendentes em um conjunto de tamanho $\mu + \lambda$ e os μ melhores indivíduos sobrevivem
- Em (μ, λ)-EE, apenas os μ melhores descendentes sobrevivem, isto força que $\lambda \geq \mu$.

Convergência de estratégias evolutivas Rechemberg fez uma regra para ajustar o valor do desvio padrão σ durante o processo evolutivo, de modo que o processo mantenha a convergência para o ótimo. Esta regra é conhecida como a *regra do sucesso 1/5* que é descrita como:

A proporção de mutações bem sucedidas (que melhoram a solução) e o total de mutações deve ser 1/5. Se for maior, então o desvio padrão aumenta, se for menor, então ele deve ser diminuído.

De uma maneira mais elegante:

$$\sigma_t = \begin{cases} \sigma_{t-n}/C & \text{se } p_s < 1/5 \\ c\sigma_{t-n} & \text{se } p_s > 1/5 \\ \sigma_{t-n} & \text{se } p_s = 1/5 \end{cases} \tag{1.13}$$

onde n é a dimensão do problema, t é a geração atual, p_s é a frequência relativa de mutações de sucesso contadas por um certo intervalo de gerações e varridas de mutações ($10n$, por exemplo) e, c é uma constante de atualização, o valor mais típico é $c = 0,817$.

Mecanismo de auto-adaptação Nas estratégias evolutivas além de evoluir variáveis do problema, também evoluem parâmetros técnicos, neste caso, os desvios-padrão σ_i. Aos antecessores selecionados para operar aplica-se o seguinte:

$$\sigma_i = \sigma_i e^{(\tau' N(0,1) + \tau N_i(0,1))} \tag{1.14}$$

$$x_i' = x_i + N(O, \sigma_i) \tag{1.15}$$

onde τ e τ' são constantes que são uma função de n.

Note que as estratégias evolutivas simulam o processo evolutivo a nível de um indivíduo permitindo operadores de recombinação, cuja quantidade de antecessores a recombinar é parametrizado por ρ, além disso, a recombinação pode ser:

Sexual, em que o operador atua sobre dois indivíduos selecionados aleatoriamente.

Panmítica, ou seja, um indivíduo é selecionado x_k e vários outros indivíduos $\mathbf{x}_l$, $l = 1, \ldots m$. O indivíduo x_k recombina com cada um dos vetores $\mathbf{x}_l$ para obter cada componente $\mathbf{x}_l'$ do vetor descendente $\mathbf{x}'$.

Além disso, o processo de seleção utilizado é determinístico.

1.4 Algoritmo Genético clássico

1.4.1 Introdução

Os algoritmos genéticos - originalmente chamados *planos reprodutivos Genéticos* - foram desenvolvidos por John H. Holland no início da década 1.960 [Holland62 ; Holland62a] os interessados em estudar os processos lógicos que ocorreram na adaptação, inspirados por estudos feitos naquela época com autômatos celulares [Schiff2007] e redes neurais [Haykin1998] notaram que o uso de regras simples podem levar a um comportamento de visualização flexível, a possibilidade de estudar a evolução de sistemas complexos. Holland percebeu que para estudar a adaptação foi necessário considerar os seguintes princípios:

1. a adaptação ocorre num ambiente,
2. a adaptação é um processo de uma população,
3. o comportamento individual pode ser representado como programas,
4. o comportamento futuro pode ser gerado por variações aleatórias nos programas,
5. a saída dos programas tem relação entre elas se suas respectivas estruturas também têm relação.

Assim, Holland conseguiu ver o processo de adaptação como um mecanismo em que os programas de uma população estão melhorando e interagem de acordo com um ambiente que determina a sua qualidade. A combinação de variações aleatórias com um processo de seleção com base na qualidade dos programas de meio ambiente, deve levar a um sistema adaptativo geral. É este sistema que Holland chamou de *Plano de Genética Reprodutiva* [Holland1975].

Embora algoritmos genéticos foram concebidos no âmbito da adaptação, parte de *Machine Learning*, atualmente são massivamente utilizados como ferramenta de otimização [Eiben2003].

1.4.2 Definição de Algoritmos Genéticos

Algoritmos genéticos, tais como técnicas de Inteligência Artificial são algoritmos de execução de métodos estocásticos de busca de um modelo de alguns

fenômenos naturais que são: a hereditariedade e o princípio darwiniano da sobrevivência do mais apto. A metáfora que está por trás de algoritmos genéticos é a evolução natural. Na evolução em nível de espécie, o problema que cada espécie enfrenta consiste em buscar adaptações que são benéficas para o ambiente que é complicado e mutável. Este conhecimento adquirido por cada uma das espécies, é construído sobre a estrutura dos cromossomos dos seus membros. O algoritmo genético trabalha em uma população de soluções e enfatiza a importância de cruzamentos sexuais (Operador) em mutação (Operador Secundário) e usa seleção probabilística com base na aptidão de soluções, ao contrário de outros paradigmas de programação evolutiva, como formação evolutiva e estratégias evolutivas.

1.4.3 Componentes de um Algoritmo Genético

[Michalewicz1996] afirma que, a fim de aplicar o algoritmo genético se requer os seguintes cinco componentes básicos:

1. Uma representação das possíveis soluções para o problema, de natureza genotípica, é chamada **cromossomo**.
2. Uma forma de criar uma população inicial de possíveis soluções, a que se refere de **inicialização**, que, tipicamente, é baseado num processo aleatório.
3. Uma função objetivo que reproduz o ambiente, qualificando as soluções de acordo com sua aptidão, é chamada função de avaliação.
4. Operadores genéticos que alteram a composição dos genomas selecionados produzindo descendentes por gerações.
5. Configurações, isto é, os valores para os diferentes parâmetros que usam o algoritmo genético (tamanho da população, a probabilidade de aplicação de cruzamento, mutação, o número máximo de gerações, etc.)

1.4.4 Algoritmo Genético Canônico

Também chamado de algoritmo genético tradicional, foi introduzido por [Goldberg89] e são usados indivíduos compostos por sequências de números binarios $\mathbf{b}_i \in 0, 1$ de fixo comprimento l que codificam todas as variáveis em questão. Neste modelo de algoritmo genético se nota claramente o conceito

de genótipo que são cada uma das cadeias binárias $\mathbf{b}_i$ que codificam uma respectiva solução ou fenótipo x_i. Embora uma codificação de fenótipo a genótipo $\gamma^{-1} : x \to g$ não está limitada a representação binária $\{0, 1\}$, esta codificação é mais semelhante aos cromossomas biológicos, que é a que [Holland1975] concebeu e implementou.

Na Tabela *1.4.4* um modelo individual típico usado no algoritmo canônico de Holland consiste em uma sequência binária $\mathbf{b}_i$.

$\mathbf{b}_i$	0	1	0	0	1	1	$\cdots$	0	1	0	1	1	0
	b_{l-1}	b_{l-2}				b_k		b_j				b_1	b_0

Tabela 1.1: Cadeia binária com comprimento l.

Uma sequência binária é chamada de um cromossomo. Cada posição da cadeia é chamada *gene* e o valor dentro desta posição (que pode ser um valor $\{0, 1\}$) é chamado *alelo*.

Procedimento Elemental O pseudocódigo para um algoritmo de básico é o seguinte:

Algoritmo 7 Método do Algoritmo Genético

$t = 0$
inicializa população de genótipos $\mathbf{G}_t$
decodificar e avaliar as estruturas de $\mathbf{G}_t$
c_{fim} = false
$t = t + 1$
escolha $\mathbf{G}_t$ de $\mathbf{G}_{t-1}$
operar G_t Formando $\mathbf{G}_t$
avaliar $\mathbf{G}_t$

onde $\mathbf{G}_t$ é a população de cromossomos na geração t, t é o iterador de gerações, G_t é o conjunto de indivíduos selecionados para reproduzir (que sofrerão cruzamento ou mutação), c_{fim} é uma condição de finalização do algoritmo.

Para a avaliação de um genótipo $\mathbf{g}_i \in \mathbf{G}_t$ é necessária a decodificação de seus fenótipos $\gamma_i : \mathbf{g}_i \mapsto \mathbf{x}_i$ para que possamos aplicar a função de avaliação $f(\mathbf{x}_i)$. Após a condição de término o algoritmo satisfeito deve parar de iterar, e o indivíduo com a melhor aptidão $\mathbf{x}_{best}$ encontrado até agora será considerado a solução obtida pelo algoritmo genético. Este critério final é tipicamente um número máximo de gerações It_{max} ou o cumprimento de um teste de convergência aplicado à população $\mathbf{X}_t$.

A Figura *1.5* ilustra este ciclo população inicial, que inclui a seleção, reprodução (cruzamento e mutação) e avaliação.

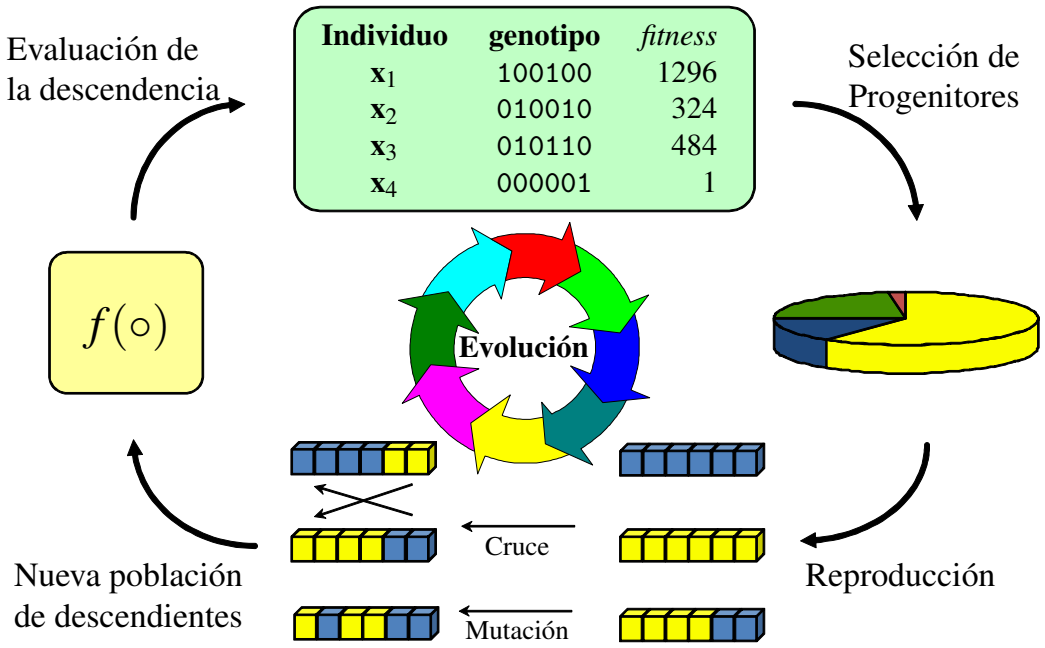

Figura 1.5: Ciclo principal de un Algoritmo Genético.

Representação Binária Como já mencionado, um algoritmo genético canônico usa cadeias de números binários com comprimento l como cromossomos que codificam as variáveis de decisão, ilustradas na Figura *1.5*.

Definição 7.3.1 (Cadeias Binárias) A cadeia binária é definida utilizando um alfabeto binário $\mathbb{B} = \{0, 1\}$ e uma cadeia binária de comprimento l ocupa um espaço $\mathbb{B}^l = \mathbb{B} \times \ldots \times \mathbb{B} = \{0, 1\}^l$. Dada uma função $f(x)$ para ser otimizada, onde $x = \{x_1, \cdots, x_n\}$, Cada uma das variáveis x_i pode ser codificada em uma cadeia binária de comprimento l_i definido pelo usuário. A codificação de todas as variáveis é obtida pela concatenação das n cadeias comprimento l_i de modo a formar uma única grande cadeia de comprimento $l = l_1 + \cdots + l_n$. Sabe-se que uma cadeia binária de comprimento l pode codificar para um total de 2^l pontos de busca, isso significa que o comprimento l_i para uma variável x_i depende da precisão desejada para esta variável. A Equação *1.18* ilustra uma típica codificação de n variáveis $\mathbf{x} = (x_1, \cdots, x_n)$ em uma cadeia binária.

$$\overbrace{\underbrace{100\ldots1}_{l_1}}^{x_1} \quad \ldots \quad \overbrace{\underbrace{010\ldots0}_{l_i}}^{x_i} \quad \ldots \quad \overbrace{\underbrace{110\ldots0}_{l_n}}^{x_n} \tag{1.16}$$

onde a variável $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ é codificado por uma concatenação de cadeias binárias de comprimento $l = l_1 + \cdots + l_i + \ldots + l_n$. Este tipo de codificação permite um algoritmo genético possa ser aplicada em uma variedade de problemas, uma vez que o mecanismo genético age na evolução das cadeias binárias de acordo com o valor de *fitness* derivado da função de avaliação $f(\mathbf{x})$ sem se preocupar com a natureza das variáveis $\mathbf{x}$. Note que o número real de possíveis valores das variáveis e de seus domínios são tratados pela codificação. Com esta opção é possível que o mesmo esquema seja aplicado em vários problemas sem muitas alterações [Deb2000]. A decodificação usada para extrair os valores das variáveis de uma cadeia binária vai começar fazendo o seguinte:

1. Dada a sequência binária $\mathbf{b} = \{b_1, \ldots, b_l\} \in \mathbb{B}^l$ se extrai a subcadeia $\mathbf{b}_i \in \mathbb{B}^{l_i}$ correspondente à variável x_i.
2. aplica a decodificação $\gamma : \mathbb{B}^l \mapsto \mathbb{X}$, onde $\mathbb{X}$ é definido de acordo a natureza das variáveis do problema.

Codificação de variáveis reais em binárias Dada uma função $f(\mathbf{x})$, onde $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n$, a ser otimizada, é necessário codificar cadeias binárias obedecendo às seguintes propriedades:

- Cada vetor $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{X}$ deve ser codificado por uma única palavra binária de comprimento l. Isso significa que você deve definir uma função de codificação γ^{-1} tal que

$$\gamma^{-1} : \mathbb{X} \to \mathbb{B}^l \tag{1.17}$$

- Cada variável x_i tem o seu próprio domínio $\langle \text{left}_i, \text{right}_i \rangle$ definido tal que satisfaz $x_i \in \langle \text{left}_i, \text{right}_i \rangle, \forall i = (1, \ldots, n)$. Se deve definir uma função de codificação γ_i^{-1} para a variável x_i como:

$$\gamma_i^{-1} : \langle \text{left}_i, \text{right}_i \rangle \subset \mathbb{R} \to \mathbb{B}^{l_i} \tag{1.18}$$

- Se limitamos precisão de cada variável de x_i para p_i dígitos decimais o espaço $\mathbb{B}^{l_i}$ deve ser capaz de codificar pelo menos $(\text{right}_i - \text{left}_i) \times 10^{p_i}$ valores para a variável x_i, como se mostra na Figura *1.6*.

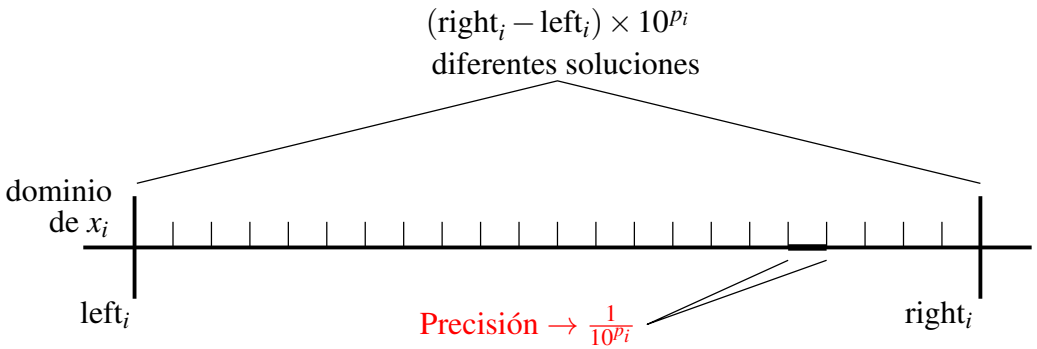

Figura 1.6: Número de possíveis valores para x_i.

- Dadas estas exigências de precisão, é possível encontrar o valor de comprimento l_i de *bits* que as satisfaçam para a variável x_i. Este valor l_i é definido pela Equação *1.21*:

$$2^{l_i} \geq (\text{right}_i - \text{left}_i) \times 10^{p_i} \tag{1.19}$$

$$l_i \geq \log_2(\text{right}_i - \text{left}_i) + p_i \log_2(10)$$

onde $l_i \in \mathbb{N}$ e que representa um número de *bits*, de que irá ser definido pelo valor natural imediatamente maior do que o valor calculado como:

$$l_i = \text{ceil}[\log_2(\text{right}_i - \text{left}_i) + p_i \log_2(10)] \tag{1.20}$$

Essa regra é aplicada porque a condição de precisão indica que, ao menos se satisfaça a precisão de dígitos decimais especificada.

- Para encontrar o tamanho l nos caracteres binários cromossômicas que codifique todas as variáveis $\mathbf{x} = \{x_1, \ldots, x_n\}$, basta adicionar todos os l_i encontrados: $\sum_i l_i$.
- É verdade que, para um dado valor $\text{left}_i \leq x_i \leq \text{right}_i$ codificado em binário, a sequência binária $\mathbf{b}_{inf} = (00\ldots0)$ representa o valor left_i e a cadeia binária $\mathbf{b}_{sup} = (11\ldots1)$ representa o valor right_i.
- A decodificação de real a binário γ_i para uma variável x_i é definida como:

$$\gamma_i : \mathbb{B}^{l_i} \to \langle \text{left}_i, \text{right}_i \rangle \subset \mathbb{R} \tag{1.21}$$

Decimal	Binario	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabela 1.2: Cadeias binárias e código de cadeia de Gray.

e é para converter a cadeia binária $\mathbf{b}_i$ para decimal e encaixá-la no intervalo de domínio $\langle \text{left}_i, \text{right}_i \rangle$ de acordo com a Equação *1.24*:

$$x_{i(\text{real})} = \text{left}_i + \left(\sum_{j=0}^{l_i - 1} 2^j a_j \right) \frac{\text{right}_i - \text{left}_i}{2^{l_i - 1}} \tag{1.22}$$

Note que para esta codificação, presume-se que a granularidade do espaço de busca para a variável x_i é uniforme e igual a $\frac{1}{2^{l_i}}$. Se esta hipótese não for cumprida, ou seja, a granularidade não é uniforme, como ocorre nos números de ponto flutuante $\pm 1.ddd \ldots \times b^E$, a função de codificação-descodificação pode ser ajustada de forma apropriada utilizando uma transformação. No entanto, para problemas reais de busca e otimização podem não existir padrões definidos de granularidade tornando mais complicado tanto como função de descodificação quanto de codificação.

Há casos em que algumas variáveis tomam valores negativos e positivos. Se o domínio destas variáveis é simétrico, ou seja, $x_i \in \langle -u_i, l_i \rangle$ pode ser usado uma codificação de representação com base de inteiros "complemento de 2" onde o *bit* mais significativo refere-se ao sinal.

Vantagens e desvantagens A representação binária é simples e fácil de usar, nos dá bons resultados e facilita a ação dos operadores genéticos, sendo capaz de descodificar reais ou inteiros; mas nem sempre é o mais adequado e tem dois problemas.

Risco por distância de *Hamming* Se define distância de *Hamming* como o número de *bits* diferentes entre duas palavras binárias. Quando é

empregado codificação binária com base em LSB e MSB, em muitos casos acontece que a distância em *bits* entre dois valores binários não tem correspondência com a distância entre os valores numéricos codificados em binários, isso pode ser observado na Tabela *1.4.4* nos valores 3 e 4. Uma opção que pode mitigar este problema é o uso de codificação Gray descrita abaixo.

O **código de Gray** é usada para obter palavras binárias $\mathbb{B} = \{0, 1\}$ utilizando uma metodologia diferente para a codificação e descodificação, tal como descrito no seguintes fases:

1. Dada uma cadeia convencional $\mathbf{b} = \{b_1, \ldots, b_l\} \in \mathbb{B}^l$, é convertido em código gray $\mathbf{a} = \{a_1, \ldots, a_l\} \in \mathbb{B}^l$ utilizando uma codificação $\gamma^{-1} : \mathbb{B}^l \mapsto \mathbb{B}^l{}_{gray}$ expressa na Equação *1.27*:

$$a_i = \begin{cases} b_i & \text{se } i = 1 \\ b_{i-1} \oplus b_i & \text{caso contrário} \end{cases} \tag{1.23}$$

   onde $\oplus$ é o operador de adição módulo 2. A principal vantagem da presente codificação é precisamente que para inteiros adjacentes, a distância de Hamming das cadeias binárias resultantes é 1.

2. A decodificação de um binário para Código Gray $\mathbf{a} = \{a_1, \ldots, a_l\} \in \mathbb{B}^l$ para convertê-lo em cadeia binária é aplicar a expressão

$$b_i = \bigoplus_{j=1}^{i} a_j, \qquad \forall i = \{1, \ldots, l\} \tag{1.24}$$

   em seguida, é possível aplicar a decodificação $\gamma_i : \mathbb{B}^{l_i} \mapsto \mathbb{X}$ que é necessária.

   A codificação completa tem a seguinte sequência:

$$\gamma^{-1} : \mathbb{X} \to \mathbb{B}^l \to \mathbb{B}^l_{gray} \tag{1.25}$$

$$\gamma : \mathbb{B}^l_{gray} \to \mathbb{B}^l \to \mathbb{X} \tag{1.26}$$

O aumento da dimensionalidade do problema Este efeito ocorre porque o comprimento l da cadeia binária resultante é muito maior do que a dimensão n dos valores reais originais para precisão de k *bits* especificado. O exemplo a seguir ilustra este efeito para uma codificação $\gamma^{-1} : \mathbb{R}^n \mapsto \mathbb{B}^l$:

Exemplo 7.1 Deixe a função $f(x_1, x_2)$ com espaço de problema $\mathbb{X} \subset \mathbb{R}^2$ definido por domínios $\langle \text{left}_1, \text{right}_1 \rangle = \langle \text{left}_2, \text{right}_2 \rangle = \langle 0, 5 \rangle$. É claro que a dimensão original do problema é $n = 2$

- Se você quer encontrar a codificação $\gamma : R^2 \mapsto B^l$ para os domínios do problema e considerando uma precisão de $k = 5$ dígitos decimais.
- A aplicação da metodologia da Seção *7.3.4* se obtém cadeias de longitude $l_1 = l_2 = 19$, e o comprimento total da cadeia de *bits* é $l_1 + l_2 = 38$ *bits* binários.
- Portanto, a codificação resultante é $\gamma^{-1} : R^2 \mapsto B^{38}$.

Inicialização da População Seja $\mathbf{X}\{x_i\}_{i=1}^m$ uma população de m indivíduos e dimensão n, isto é, com n variáveis $\mathbf{x} = (x_1, \ldots, x_n)$. A inicialização da população inicial $\mathbf{X}_0$ é realizada para cada indivíduo $\mathbf{x}_i \in \mathbf{X}_0,\ i = 1, \ldots, m$ conforme a Equação *1.31*, colocando valores aleatórios em cada uma das variáveis $x_1, x_2, \ldots, x_n$. Claramente, essa inicialização deve respeitar os domínios de cada variável $x_j,\ \forall j = (1, \ldots, n)$.

$$x_j \in \langle \text{left}_j, \text{right}_j \rangle \rightarrow x_j = \text{left}_j + (\text{right}_j - \text{left}_j)\mathbf{U}(t) \tag{1.27}$$

onde $\mathbf{U}(t)$ é um valor aleatório com distribuição uniforme $(0, 1)$. Nota-se que se a realização de $\mathbf{U}(t) = 0$, o valor inicializado coincide com o left_j, e se realizando $\mathbf{U}(t) = 1$, o valor inicializado corresponde a right_j.

Codificação de inicialização $\gamma : \mathbb{R}^n \mapsto \mathbb{B}^l$ Se você estiver usando um mapeamento $\gamma^{-1} : \mathbb{R}^n \mapsto \mathbb{B}^l$, cada variável x_i será codificada por cadeias $\mathbf{b} \in \mathbb{B}^{l_i}$ com valores $b_i \in \{0, 1\}$. A inicialização pode ser feita diretamente gerando *bits* aleatórios 0 ou 1 e colocando-os em cada alelo b_i da cadeia $\mathbf{b}$ sendo inicializado, sem preocupações com os domínios pois, a precisão foi determinada pela aplicação Equação *1.22* e domínios são definidos e garantidos tanto codificação $\gamma^{-1} : \mathbb{R}^n \rightarrow \mathbb{B}^l$ quanto na decodificação $\gamma : \mathbb{B}^l \mapsto \mathbb{R}^n$ como já foi visto na Equação *1.24*.

Exemplo 7.2 Deixe a função F_6, aplicada a duas variáveis, $F_6(x_1, x_2)$ que é uma função típica cheia de máximos locais e mínimos, o que torna difícil encontrar o ótimo global x^* que é bem conhecido e localizado em $f(x^*) = f(0, 0) = 1,00$. A Equação *1.32* representa a função F_6.

$$F_6(x_1, x_2) = 0,5 - \frac{(\text{sen}\sqrt{x_1^2 + x_2^2})^2 - 0,5}{(1,0 + 0,001(x_1^2 + x_2^2))^2} \tag{1.28}$$

Esta equação tem duas variáveis $x_1, x_2 \in X \subset \mathbb{R}^2$ e não se restringe a valores a priori para x_1 ou x_2. Para este exemplo, se encaixa x_1, x_2 nos seguintes domínios:

$$x_1 \in \langle -100, 100 \rangle$$
$$x_2 \in \langle -100, 100 \rangle$$

Na Figura *1.7* se exibe parte dessa função, para os intervalos $x_1 \in \langle -10, 10 \rangle, x_2 \in \langle -10, 10 \rangle$, onde os múltiplos máximos e mínimos são observados que são característicos da função f_6. Observe que o ponto $x^* = (0, 0)$ e o valor $f(0, 0) = 1,0$ é o máximo global, tal como indicado na figura.

Definindo o cromossomo Suponha que você quer codificar o indivíduo usando cadeias binárias, isso nos obriga a pensar em uma codificação do tipo $\gamma^{-1} : \mathbb{X} \to \mathbb{B}^l$, onde $\mathbb{B}^l$ é o espaço de cadeias binárias de comprimento l inteiro representando indivíduos. Se uma precisão de 4 a 5 casas decimais é definido para cada variável x_i, aplicando a Equação *1.22*, se calcula o número $l_i \in \mathbb{N}$ de *bits* binários para cada x_i do seguinte modo:

$$\begin{aligned} \log_2((100 - (-100) \times 10^4) \leq \ & l_i \ \leq \log_2((100 - (-100) \times 10^5) \\ \log_2(2 \times 10^6) \leq \ & l_i \ \leq \log_2(2 \times 10^7) \\ 20,93 \leq \ & l_i \ \leq 24,25 \end{aligned}$$

Temos que l_i pode assumir os valores 21, 22, 23, 24, que é selecionado $l_i = 22$. Portanto, o cromossomo binário que codifica $\mathbf{x} = (x_1, x_2)$ tem um comprimento total $l_1 + l_2 = 44$ *bits*. A inicialização da população inicial consiste simplesmente no preenchimento de valores $\{0, 1\}$ aleatórios para cada cromossomo codificado $\mathbf{b}_i \in \mathbf{B}^0$.

1.4.5 Função de avaliação

A função de avaliação é o elo entre o algoritmo genético e o problema em questão. Pode ser definida como o resultado da aplicação no problema a ser

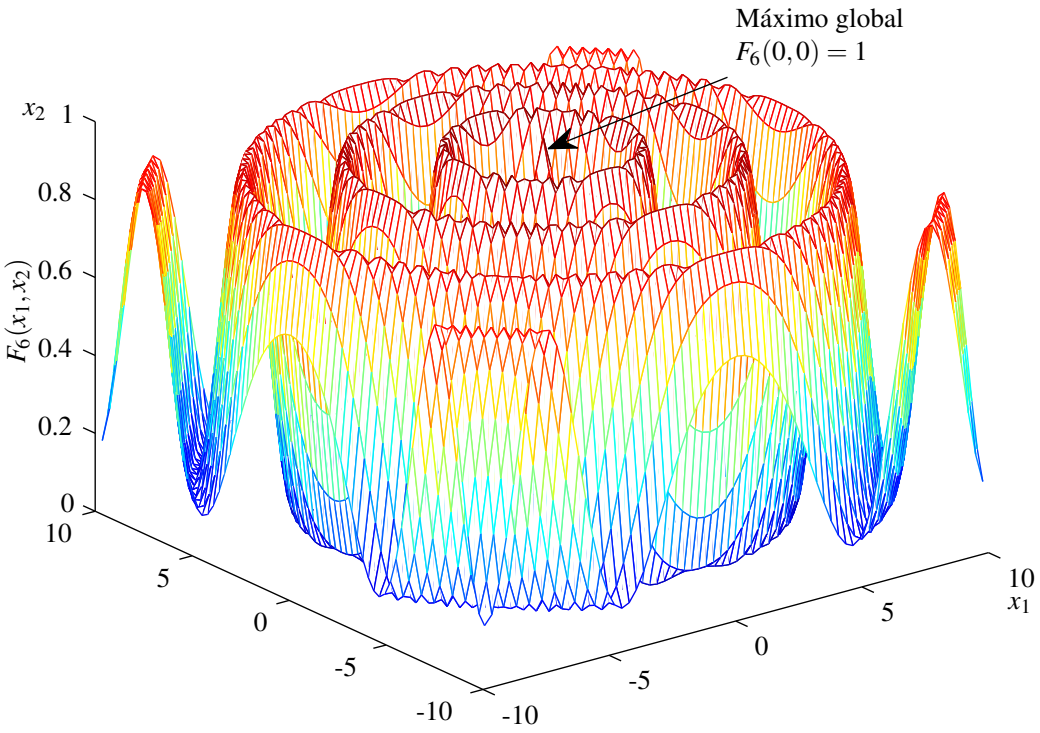

Figura 1.7: Visualização da função F_6.

otimizado, os valores $\mathbf{x} = \{x_1, \ldots, x_n\}$ propostos em um indivíduo. Dado o indivíduo $\mathbf{x}_i$, sua função de avaliação é denotada como $f(\mathbf{x}_i)$.

Aptidão A partir da função de avaliação, a aptidão a é definida pelo valor numérico que é atribuído a cada indivíduo (fenótipo) $\mathbf{x}_i$ indicando o quão bom é em relação a outros indivíduos da população $\mathbf{X}$ para a solução, como na Equação *1.33*.

$$a(f(x_i), \mathbf{X}) = \text{aptidão de } \mathbf{x}_i \tag{1.29}$$

Observe que, para muitos casos, a aptidão é a própria função de avaliação, isto significa que $a(f(x_i), \mathbf{X}f(x_i)$ que geralmente pode criar confusão entre os dois. Por isso, deve ficar claro que a função avaliação depende apenas do indivíduo x_i e do problema de otimização, enquanto que a aptidão de um mesmo indivíduo deve depender dela e do resto de indivíduos da população $\mathbf{X}$ em relação ao problema de otimização.

Exemplo 7.3 Assumindo a utilização de uma codificação/decodificação de

binário para inteiro $\gamma : \mathbb{B}^{12} \mapsto \mathbb{Z}^2$ com cadeias binárias de comprimento $l_1 + l_2 = 6 + 6$ para fenótipo (x_1, x_2) e uma função de aptidão igual a $f(x_1, x_2) = x_1^2 + x_2^2$, temos que

$$f(010011_2, 010110_2) = 19^2 + 22^2 = 845 \tag{1.30}$$

Não há qualquer limitação sobre a complexidade da função de avaliação, pode ser tão simples como a equação polinomial do exemplo *7.3*, ou ser tão complexa exigindo a execução de vários módulos de simulação para avaliar os valores das variáveis especificadas no indivíduo. A Figura *1.8* ilustra o quão sério é o processo de avaliação para o problema f_6.

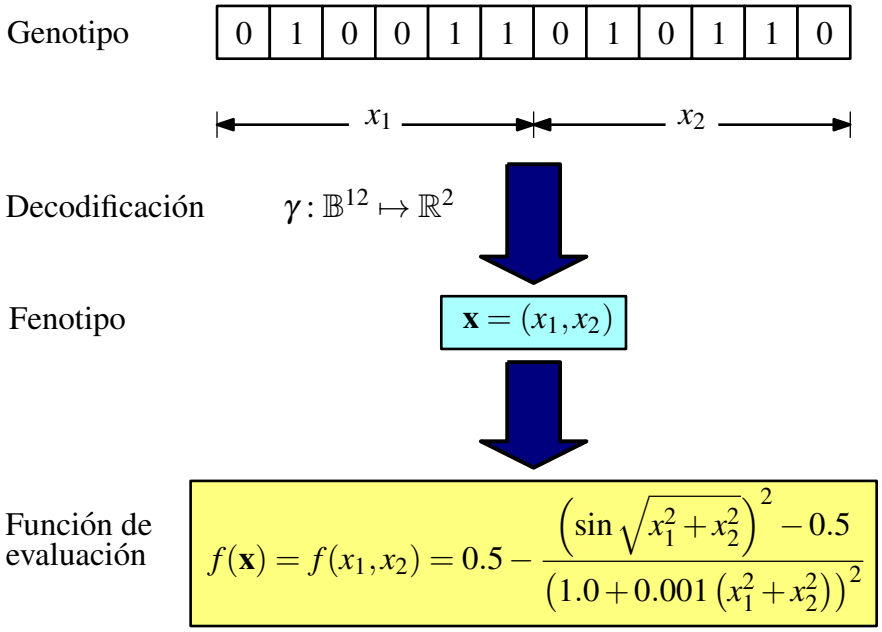

Figura 1.8: Cromossoma e função de avaliação.

O objetivo desta fase é obter, a partir de uma população $\mathbf{X}_t$ na geração t, um vetor de aptidões $\mathcal{A}_t$ que vai ser usado na seguinte etapa do processo evolutivo: a seleção.

1.4.6 Estratégias de Seleção

Uma vez criada uma população inicial de m indivíduos $\mathbf{X}_0 = \{x_i\}_{i=1}^m$ e calculada aptidão de cada um de seus indivíduos $\mathbf{x}_i$ armazenada no vetor

de aptidões $\mathcal{A}_t$, os indivíduos são selecionados para serem reproduzidas. Esta seleção é geralmente baseada na capacidade de cada indivíduo para $a(f(\mathbf{x}_i), \mathbf{X}_t)$ o que representa atualmente uma *probabilidade* de seleção de cada indivíduo $\mathbf{x}_i$ com respeito a outros indivíduos na população. A seleção normalmente envolve um sorteio usando probabilidade, que o caracteriza como um processo probabilístico.

As técnicas de seleção utilizados em algoritmos genéticos podem ser classificados em 2 grupos:

- Seleção proporcional
- Seleção por torneio

Seleção Proporcional Que são algumas das estratégias para a seleção originalmente propostas por Holland [Holland1975] cuja característica é que os indivíduos são selecionados de acordo com a sua contribuição de aptidão com respeito ao total de aptidão da população. Dentro deste grupo, temos as seguintes estratégias

- Seleção por Roleta
- Resto Estocástico,
- Deterministic Sampling.
- Universal Estocástico

Além disso, existem outras técnicas sendo proporcionais, implementam os seguintes adiantamentos

- Escalonamento-σ
- Usando hierarquias
- Seleção de Boltzmann

Seleção por Roleta Entre todas as estratégias de seleção, a mais utilizada por sua simplicidade e facilidade de compreensão é a seleção de roleta. Seu princípio de funcionamento baseia-se na formação de um vetor de aptidões acumuladas onde cada indivíduo trás o seu peso. Um sorteio é então realizado gerando uma posição entre 0 e o valor acumulado total e se seleciona o indivíduo que, doou a parte em que o ponto é encontrado. Desta forma, indivíduos cuja contribuição para a aptidão foi maior, terão mais chance de serem selecionados. O algoritmo é descrito no procedimento abaixo:

Algoritmo 8 Seleção por Roleta

$\mathbf{X}_t$, A_t
calcular $\mathcal{B}_t = \{b_1, \ldots, b_m\}$
$j = 0$
gerar $u_j = b_m \mathbf{U}(0, 1)$
$u_j \notin [x_j, x_{j+1}]$
$j = j + 1$
selecionar $x_j \in \mathbf{X}_t$

onde: $\mathcal{A}_t$ é o vetor de aptidões de $\mathbf{X}_t$ e $\mathcal{B}_t$ com componentes $\{b_1, \ldots, b_m\}$ é o vetor de aptidões acumuladas de $\mathbf{X}_t$

indivíduo i	fenótipo	$a_i \in \mathcal{A}_t$	$b_i \in \mathcal{B}_t$	a_i/b_m (%)
$\mathbf{x}_1$	11010110	254	254	24,49%
$\mathbf{x}_2$	10100111	47	301	4,53%
$\mathbf{x}_3$	00110110	457	758	44,07%
$\mathbf{x}_4$	01110010	194	952	18,71%
$\mathbf{x}_5$	11110010	85	1037	8,20%
Total		**1037**		$\mathbf{100,00}\%$

Tabela 1.3: Exemplo de roleta habilidades de aplicação

que é calculada como se segue:

$$\begin{array}{rl} b_i = a_i & i = 1 \\ b_i = b_{i-1} + a_i & i > 1 \end{array} \tag{1.31}$$

b_m é o valor acumulado total de aptidão, $\mathbf{U}(0, 1)$ é um gerador de números aleatórios entre 0 e 1.

Uma vez que um indivíduo x_i e a sua aptidão $a(x_i, \mathbf{X}_t)$, sua probabilidade de seleção $p_{\mathbf{x}_i}$ para o modelos de seleção por roleta será:

$$p_{\mathbf{x}_i} = \frac{a(\mathbf{x}_i, \mathbf{X}_t)}{\sum_{k=1}^{m} a_k} \tag{1.32}$$

Na Tabela *1.4.6* um conjunto de tais valores é mostrado para um algoritmo genético com cromossomos binários.

Cuja roleta resultante para aplicar o algoritmo de seleção é ilustrado pela Figura *1.9*:

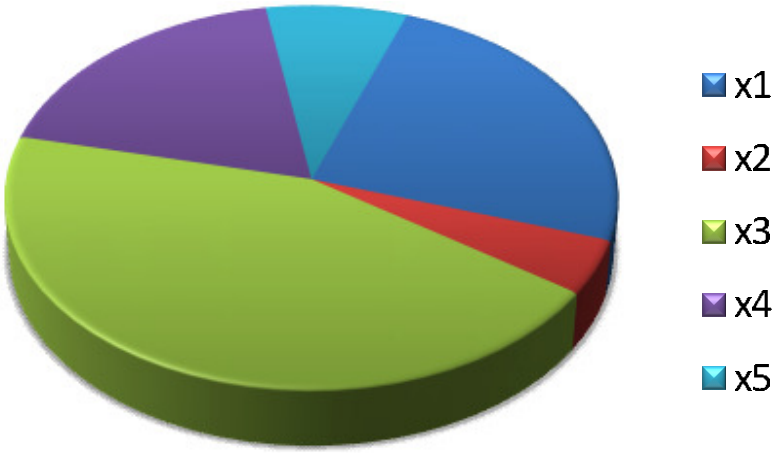

Figura 1.9: Exemplo de roleta para as pessoas e habilidades Tabela *1.4.6*.

Análise da seleção por Roleta A roleta tem algumas deficiências descritas a seguir:

- A sua complexidade computacional é $O(n^2)$. O algoritmo é ineficiente quando o tamanho m da população cresce.
- É sujeito a erro de amostragem, isto é, o valor esperado $E_s[\mathbf{x}_i]$ de seleção, que depende da aptidão relativa (ou probabilidade) de seleção de um indivíduo, nem sempre cumpre, podendo haver diferenças grandes. Isto é visualizado na possibilidade de que o pior indivíduo pode ser escolhido repetidamente.
- Você pode usar busca binária em vez de busca linear, que exige memória adicional e uma primeira varredura com a complexidade $O(n)$. Com isso, a complexidade total da roleta se torna $O(n \log n)$.

Resto Estocástico É uma alternativa que visa obter uma melhor aproximação dos valores esperados de seleção e os indivíduos $E_s[\mathbf{x}_i]$ definido pela Equação *1.37*:

$$E_s[\mathbf{x}_i] = m\frac{a(f(\mathbf{x}_i), \mathbf{X}_t)}{b_m} \tag{1.33}$$

onde: $a(\cdot)$ é a aptidão do indivíduo $\mathbf{x}_i$, b_m é aptidão acumulada da população e m o número de elementos na população.

O que se faz é:

1. Atribuir de forma determinística a contagem dos valores esperados, é deixar a parte inteira de $e_t = m(a_i/b_m)$ em seguida,

2. Usando um esquema probabilístico e proporcional, preencha os indivíduos remanescentes por arredondamento decimal usando a parte restante.

O esquema de completamento de indivíduos desaparecidos tem duas variantes:

Sem substituição onde todo excesso é usado para um sorteio que determina se um indivíduo é selecionado ou não.

Com substituição onde o excedente é usado para dimensionar uma roleta e se usa esta técnica para a seleção.

A Tabela *1.4.6* ilustra um exemplo de utilização do resto estocástico.

Indivíduo i	fenótipo	$a_i \in \mathcal{A}_t$	e_t	$\mathbf{int}(e_t)$	$e_t - \mathbf{int}(e_t)$
$\mathbf{x}_1$	110100	220	1,23	1	0,23
$\mathbf{x}_2$	011010	140	0,78	0	0,78
$\mathbf{x}_3$	111001	315	1,76	1	0,76
$\mathbf{x}_4$	001101	42	0,23	0	0,23
Total		717	4,00	2	

Tabela 1.4: Exemplo de Resto Estocástico

Essa metodologia permite reduzir questões de amostragem global da roleta, embora possa também causar convergência prematura devido à pressão mais elevada seletiva.

Seleção do Torneio A seleção do torneio ou *Tournament Selection* foi proposta em [Wetzel1983] sua idéia básica é selecionar indivíduos comparando diretamente suas aptidões como descreve o algoritmo 9:

Algoritmo 9 Seleção por Torneio

$\mathbf{X}_t$, $\mathcal{A}_t$
bagunçar $\mathbf{X}_t$
escolher k indivíduos
comparar os k indivíduos na sua capacidade $a(\cdot)$
selecionar o indivíduo com melhor aptidão

onde k geralmente tem o valor de $k = 2$.

A seleção torneio é bastante simples e é determinista porque os indivíduos menos aptos nunca serão selecionados (característica extintiva).

Para compensar este efeito, uma versão do torneio probabilístico cuja diferença principal surge quando se escolhe o vencedor. Em vez de selecionar o indivíduo mais apto, um sorteio é feito $\mathbf{U}(t) < p$, onde $\mathbf{U}(t)$ é um gerador de números aleatórios entre $[0, 1)$ e $0,5 < p \leq 1$. Se o resultado for cumprido, ele seleciona o elemento mais adequado, caso contrário, selecione o menos apto. O valor p permanece fixo durante o processo evolutivo. Esta variante probabilística reduz a pressão seletiva eliminando a caracterização extintiva, porque as vezes os elementos menos aptos poderiam ganhar o torneio e serem selecionados.

Análise do Torneio

- A versão determinista garante que o melhor indivíduo seja selecionado
- Cada competição requer seleção aleatória de k indivíduos, o que é um valor constante, de modo que você pode dizer que a sua complexidade é $O(1)$.
- são necessários m competências para conseguir selecionar uma nova população completa para uma geração. Por isso, a complexidade desse algoritmo é $O(m)$.
- A técnica é eficiente e fácil de implementar.
- Não há necessidade de subir a função de aptidão, são feitas comparações diretamente.
- Na versão determinista, você pode introduzir uma pressão seletiva forte, uma vez que os indivíduos menos aptos não tem nenhuma chance de sobrevivência.
- A pressão de seleção é ajustável através da variação do tamanho k dos indivíduos que competem.
  - Para $k = 1$, a seleção é equivalente a um passeio aleatório sem pressão seletivo
  - Para $k = m$, a seleção é completamente determinista sempre escolhendo o melhor elemento da população
  - Para $2 \leq k \leq 5$, é considerado uma seleção suave
  - Para $k \geq 10$ é considerada uma seleção dura.

1.4.7 Operadores Genéticos

Os operadores são os mecanismos que manipulam o genótipo **g** e têm influência sobre como a informação genética é transmitida de pais para filhos. Os principais operadores recaem nas seguintes categorias:

Cruzamentos: (ou*crossover*) que são dois novos indivíduos a partir da troca de peças ou de recombinação de informações dois indivíduos selecionados para operar (em sentido amplo, o número de ancestrais poderia ser maior do que dois) neste tipo de operador a transmissão de características hereditárias ocorre. Os tipos de cruzamentos mais conhecidos são o ponto de um ponto, cruzamento de dois pontos e cruzamento uniforme.

Mutações: a formação de um novo indivíduo a partir de alterações nos genes de um indivíduo selecionado.

Cruzamento de um ponto Neste operador, a partir de um par de genótipos selecionados $\mathbf{g}_1$ e $\mathbf{g}_2$ e selecionando um corte de aleatório, gerar dois descendentes $\mathbf{g}'_1$ e $\mathbf{g}'_2$ trocando algumas de suas informações como mostrado na Figura *1.10*

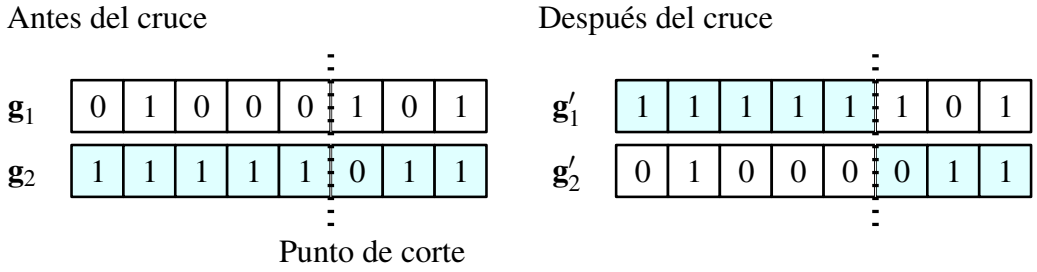

Figura 1.10: Cruzamento de um ponto.

Cruzanento de dois pontos Neste caso, dois pontos de corte são usados. É obtido um segmento cromossômico dos pais $\mathbf{g}_1$ e $\mathbf{g}_2$ para ser trocado como ilustrado na Figura *1.11*.

Cruzamento Uniforme Dado os cromossomos selecionados $\mathbf{g}_1$ e $\mathbf{g}_2$ este operador varre suas respectivas estruturas usando um sorteio $\mathbf{U}(t)_i$ para determinar qual gene g_{1_i} ou g_{2_i} dos pais contribuirá para formar os genes da

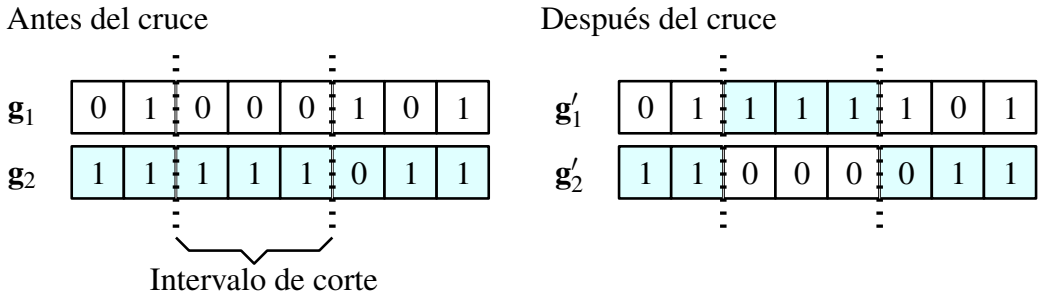

Figura 1.11: Cruzamento de dois pontos.

descendência $\mathbf{g}'_1$ e $\mathbf{g}'_2$ como mostrado na Equação *1.38* e ilustrada na Figura *1.12*.

$$g'_{1_i} = \begin{cases} g_{1_i} & \text{se} \quad \mathbf{U}(t)_i \leq 0,5 \\ g_{2_i} & \text{caso contrário} \end{cases} \qquad g'_{2_i} = \begin{cases} g_{1_i} & \text{se} \quad \mathbf{U}(t)_i > 0,5 \\ g_{2_i} & \text{caso contrário} \end{cases} \tag{1.34}$$

Mutação Os operadores de mutação genética selecionam um gene $g_i \in \mathbf{g}$ e o alteram por outro valor. Sendo de um indivíduo com genótipo binário, um gene seria um *bit* e o seu valor de alteração é 1 para 0 ou vice-versa.

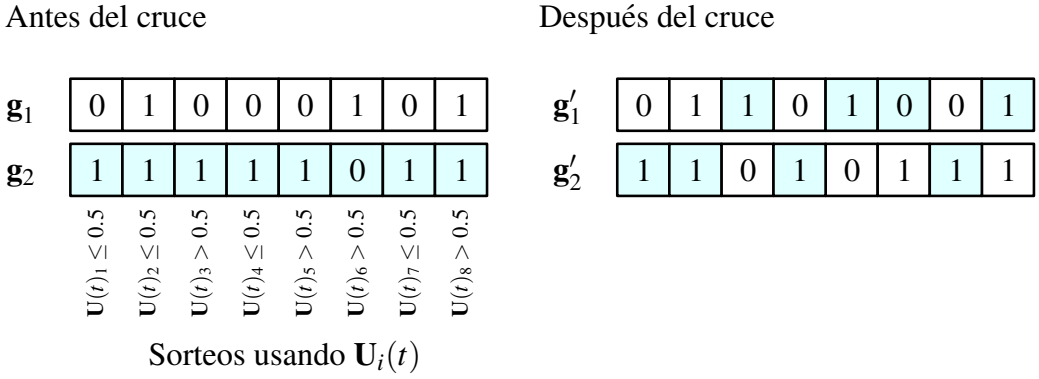

Figura 1.12: *Crossover* uniforme.

O operador de mutação pode sofrer algumas variações para selecionar e alterar o *bit* como descrito:

- Opção 1: Aplicar o sorteio a todos os genes no cromossomo com probabilidade de mutação p_m e para cada gene selecionado, coloque um valor aleatório.

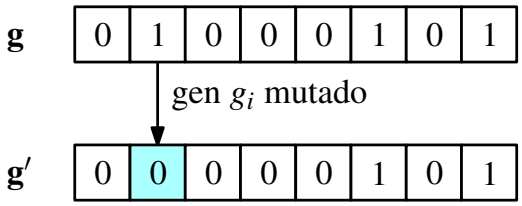

Figura 1.13: Mutação de um *bit*.

- Opção 2: Aplicar sorteio e colocar o bit complementar no gene sorteado.
- Opção 3: Escolha aleatoriamente um único gene no cromossomo e mute seu conteúdo através de um valor aleatório.

O Dilema *Exploiting-Exploring* Na maioria dos sistemas de aprendizagem existe uma necessidade de construir sobre o conhecimento existente (*exploiting*) e também olhar para novos conhecimentos (*exploring*), que são características ou comportamentos que, a priori, parecem contraditórias, caracterizando um dilema ou paradoxo. Em geral, qualquer algoritmo evolutivo, não é estranho a esta situação.

Em um modelo evolutivo, o *exploiting* se parece com o aproveitamento de informações que a população atual $\mathbf{X}_t$ tem sobre o espaço de problema $\mathbb{X}$ e em determinar os locais mais promissores e interessantes para visitar, ou seja, *sacar o jogo* para as informações que temos sobre a população atual. Além disso, o *exploring* é exibido na necessidade de atingir as regiões no espaço $\mathbb{X}$ nunca antes visitados para ver se aparece qualquer uma nova informação promissora, que é o *salto para o desconhecido*. Este recurso também dá-lhe a opção de não ficar preso em ótimos locais. Neste sentido operadores de crossover fornecem recursos exploiting e operadores de mutação fornecem a característica exploring.

$$\text{Crossover} \rightarrow \text{Exploiting}$$

$$\text{Mutação} \rightarrow \text{Exploring}$$

Uma consequência deste efeito, é a necessidade de ajustamento das probabilidades cruzamento e mutação durante o ajuste de um algoritmo evolutivo à procura de um melhor desempenho na resolução de um determinado problema de pesquisa ou otimização.

1.4.8 Ajustes da aptidão

Quando o cálculo de aptidão é feito, podemos encontrar dois problemas extremos:

Pouca pressão seletiva: o que acontece quando as habilidades, produto da avaliação dos cromossomos possuem valores numéricos semelhantes, fazendo qualquer algoritmo de seleção ineficiente. Nesta situação, o processo evolutivo tem um comportamento muito próximo ao de seleção aleatória.

Superindividuo: quando há um ou alguns indivíduos da população com elevada aptidão um em relação a outros indivíduos da população. Neste caso, o processo de seleção para selecionar tenderá para somente esses indivíduos se reproduzir, portanto, estes irão invadir populações sucessivas que causam perda de diversidade genética e convergência prematura.

Para evitar esses dois efeitos indesejados, você pode fazer ajustes competências da população, tais como:

- Aptidão = Avaliação
- *Windowing*
- Padronização linear

como descrito abaixo

Aptidão = Avaliação É o caso mais simples, em que não há necessidade de ajustes na função de avaliação $f(\mathbf{x}_i)$ para se ter aptidão, ou seja, não se deseja um problema de superindividuo em competição próxima. É utilizar a definição da equação de aptidão *29* igualando à própria avaliação do indivíduo:

$$a(x_i, \mathbf{X}_t) = f(\mathbf{x}_i) \tag{1.35}$$

Windowing Quando ocorre o problema de concorrência devido à indivíduos terem avaliações muito semelhantes, é necessário o ajuste da aptidão utilizando a seguinte expressão:

$$a(f(\mathbf{x}_i), \mathbf{X}_t) = f(\mathbf{x}_i) - f_{min}(\mathbf{X}_t) + a_{min} \tag{1.36}$$

onde $f_{min}(\mathbf{X}_t)$ é a avaliação mínima encontrada na população, a_{min} (opcional) é uma capacidade mínima para garantir a reprodução de cromossomos menos aptos.

Normalização linear É o tipo de ajustamento utilizado em algoritmos genéticos, que procura mitigar o efeito de superindividuos e de indivíduos com próxima competição, manutendo um equilíbrio de pressão seletiva, tal como descrito na continuação.

Seja uma população $\mathbf{X}_t$ com m indivíduos ordenada pelas avaliações $\mathcal{F}_t$. Por uma normalização linear, habilidades são ajustadas a partir de um valor mínimo v_{min} para um valor máximo v_{max} com passos fixados, como mostrado no algoritmo 10.

Algoritmo 10 Normalização Linear

$\mathbf{X}_t$, $\mathcal{F}_t$, v_{min}, v_{max}
ordenar $\mathbf{X}_t$ decrescente de acordo $\mathcal{F}_t$
$i = 0$ $i < M$
criar $\mathcal{A}_t = \{a_i\}_{i=1}^{m}$ usando: $a_i = v_{min} + \frac{v_{max} - v_{min}}{m-1}(i-1)$
$i = i + 1$

onde $\mathcal{F}_t$ é o vetor de avaliações da população $\mathbf{X}_t$, v_{min} é o valor mínimo de aptidão padronizada.

v_{max} é o valor máximo de aptidão normalizada, a_i é a aptidão normalizada linearmente para o indivíduo $\mathbf{x}_i$ da população $\mathbf{X}_t$ ordenada.

Significativamente, a pressão seletiva é diretamente relacionada com $|v_{max} - v_{min}|$.

Exemplo 7.4 Seja uma população $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{6}$ cujas avaliações $f(\mathbf{x}_i)$ são especificadas na segunda coluna da Tabela *1.4.8*.

$\mathbf{x}_i$	$f(\mathbf{x}_i)$	a_i $\langle v_{min}, v_{max}\rangle = \langle 10, 60\rangle$	a_i $\langle v_{min}, v_{max}\rangle = \langle 1, 101\rangle$
$\mathbf{x}_6$	200	60	101
$\mathbf{x}_5$	9	50	81
$\mathbf{x}_4$	8	40	61
$\mathbf{x}_3$	7	30	41
$\mathbf{x}_2$	4	20	21
$\mathbf{x}_1$	1	10	1

Tabela 1.5: Exemplo de Normalização linear

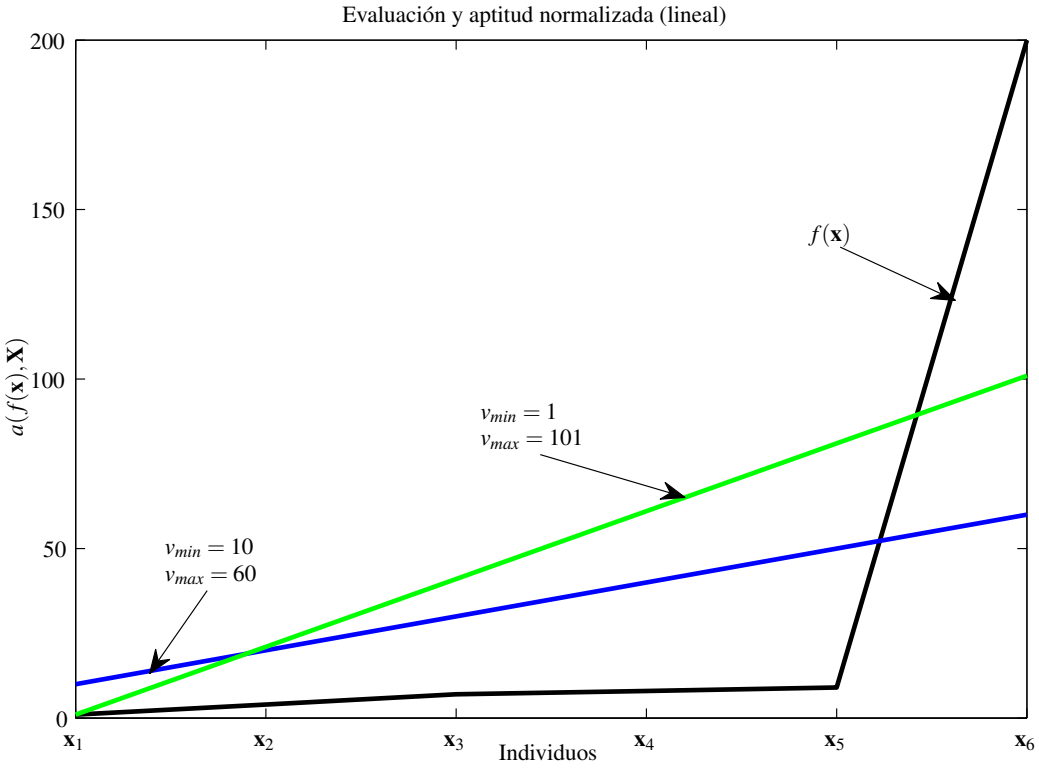

Figura 1.14: Exemplo de Normalização linear.

Olhando a Tabela *1.4.8*, e a Figura *1.14*, note que $\mathbf{x}_6$ é um superindividuo com uma avaliação $f(\mathbf{x}_6) = 200$ é muito elevada em relação a outros indivíduos. Deve também ser notado que existe uma estreita competição entre o indivíduo $\mathbf{x}_3$, $\mathbf{x}_4$ e $\mathbf{x}_5$ cujos valores de avaliação são muito próximos.

Através da aplicação da normalização linear, os efeitos indesejáveis destes dois casos são atenuados, e a pressão seletiva pode ser definida pela diferença entre v_{min} e v_{max}.

Para além da normalização linear, podem ser aplicados outros padrões à aptidão dos indivíduos, tais como o ajuste geométrico, exponencial, logarítmica, etc.

1.4.9 Ajustes da Seleção

Uma questão a se observar é chamada *conflito de gerações* e refere-se aos elementos de transição da população $\mathbf{X}_t$ para a população $\mathbf{X}_{t+1}$.

O algoritmo genético convencional considera ao fazer a seleção, toda uma população nova é obtida que posteriormente se operará (cruzamentos) e mutações. Isso caracteriza um comportamento extintivo em alguns casos, não pode ser muito benéfica, pois poderia, eventualmente, ter indivíduos valiosos perdidos. Para resolver esta situação são usadas duas estratégias.

Elitismo Consiste em armazenar uma cópia do melhor indivíduo da população $\mathbf{x}_{best} \in \mathbf{X}_t$ para colocar na população $\mathbf{X}_{t+1}$ depois de fazer todo processo de evolução (seleção, reprodução, avaliação). Esta estratégia reduz o efeito aleatório do processo evolutivo e também garante que através das gerações, o melhor indivíduo em $\mathbf{x}_{best} \in \mathbf{X}_{t+1}$ sempre será igual ou melhor do que o melhor indivíduo $\mathbf{x}_{best} \in \mathbf{X}_t$ preservando os melhores indivíduos para continuar jogando.

Steady-state Neste caso, não é uma substituição total dos indivíduos na população $\mathbf{X}_t$ para gerar a população $\mathbf{X}_{t+1}$. A metodologia de realização estado estacionário (*Steady-State*) é aquele com nos mostra o algoritmo 11.

Algoritmo 11 Steady-state

selecionar $k < m$ indivíduos de $\mathbf{X}_t$
operar esses k indivíduos (cruzamento, mutação)
remover os k piores elementos da $\mathbf{X}_t$
obter $\mathbf{X}_{t+1}$ inserindo os n indivíduos gerados

onde k é a fração de indivíduos a ser substituído, $k < m$, $k = GAP \times m$.

A Figura *1.15* ilustra a operação de estado estacionário.

Além disso, você pode fazer a substituição parcial da população e excluir indivíduos repetidos: O estado estacionário sem repetidos. Com esta estratégia pode-se obter as seguintes vantagens:

- Evitar repetições são mais frequentes em populações com steady-state que são mais estáticas.
- busca mais eficiente em paralelismo, como se garantem diferentes elementos (diversidade).

Neste caso, cada indivíduo repetido é tratado das seguintes maneiras

1. Substituído por um novo, gerado aleatoriamente (verificando não ser repetido) ou

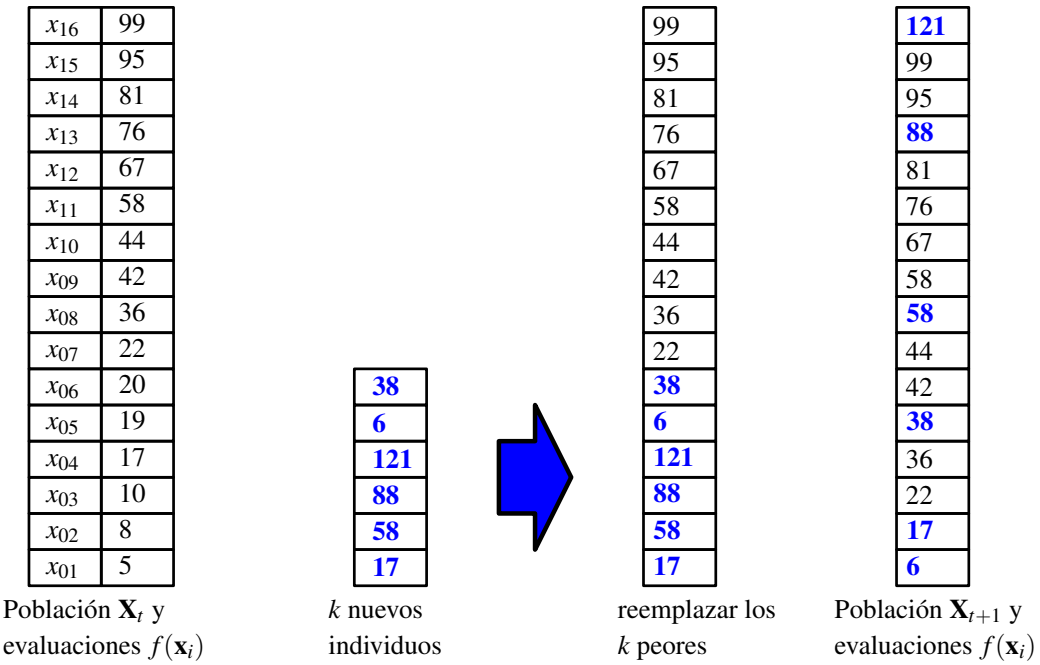

Figura 1.15: Procesimento *Steady-State*.

2. Operar o indivíduo (cruzamento ou mutação) até que o resultado é um indivíduo diferente

Há um aumento do custo computacional devido à verificação de indivíduos repetidos.

Medidas de controle Uma forma de visualizar o comportamento de um algoritmo evolutivo é ver como as pessoas se comportam durante a execução do processo evolutivo, ou seja, para as gerações até que sejam dadas a conclusão de desempenho. O mais comum é a obtenção de curvas de desempenho das execuções de um algoritmo. Os seguintes tipos de curva são conhecidos:

- Curva *on-line*
- Curva *off-line*
- Curva *best-so-far*

Curva on-line Uma curva *on-line* permite rápida obtenção de boas soluções e da convergência dos indivíduos na população. Dado um experimento que

começou com uma população inicial $\mathbf{X}_0$, um ponto da correspondente curva on-line para a geração t é dado por:

$$v_{on}(t) = \frac{1}{t+1} \sum_{i=0}^{t} \bar{f}(\mathbf{X}_t) \tag{1.37}$$

onde $\bar{f}(\mathbf{X}_t)$ é a média das avaliações de todos os indivíduos $x_i \in \mathbf{X}_t$, na geração t. A partir da Equação **??** pode-se inferir que um ponto da curva *online* representa a média de todos os indivíduos que foram avaliados para terminar a geração de t.

Curva off-line A curva off-line de exibição permite a obtenção de boas soluções, sem se importar com o tempo necessário para encontrá-los. Como experiência, para a geração t o valor do ponto na curva *off-line* é dado por:

$$v_{off}(t) = \frac{1}{t+1} \sum_{i=0}^{t} f(\mathbf{x}_{best})(t) \tag{1.38}$$

onde $f(\mathbf{x}_{best})(t)$ é a melhor avaliação do elemento de geração t. A partir da Equação **??** pode-se inferir que a curva off-line traça a média os melhores elementos desde a geração 0 até a geração t.

Curva best-so-far É a curva mais básica em que todo o valor do ponto da geração t é simplesmente o valor $\mathbf{x}_{best}(t)$, sem cálculo de médias, como expressa na Equação **??**.

$$v_{best}(t) = f(\mathbf{x}_{best})(t) \tag{1.39}$$

onde $f(\mathbf{x}_{best})(t)$ é a melhor avaliação do elemento de geração t. Como um algoritmo genético é essencialmente um processo estocástico de pesquisa, é necessário realizar muitas execuções do mesmo problema com os mesmos parâmetros para uma visão comportamental mais real da população para as gerações. Então, é comum realizar muitas execuções e calcular cada uma das curvas off-line, on-line, e a curva de elementos best-so-far, assim em seguida, mostrar as curvas caminhos com metade dos pontos nas gerações $0, \ldots, T$.

A Figura *1.16* ilustra um exemplo de curvas best-so-far, off-line e on-line para a implementação de um algoritmo evolutivo com 200 gerações.

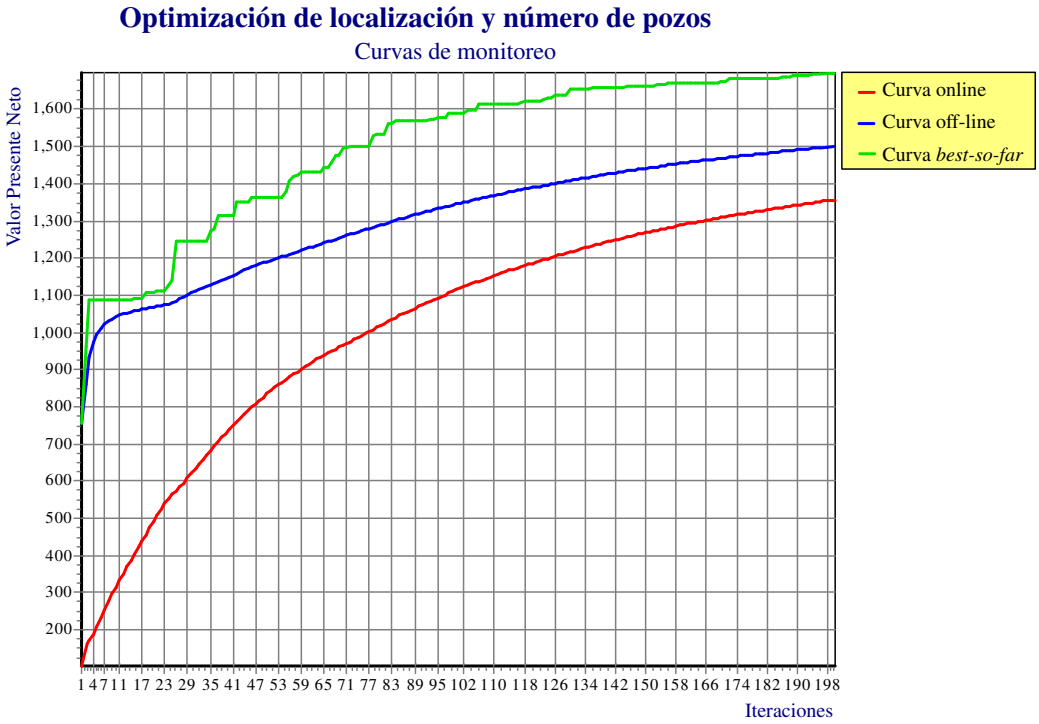

Figura 1.16: Exemplo de curvas *best-so-far*, *off-line* e *online*.