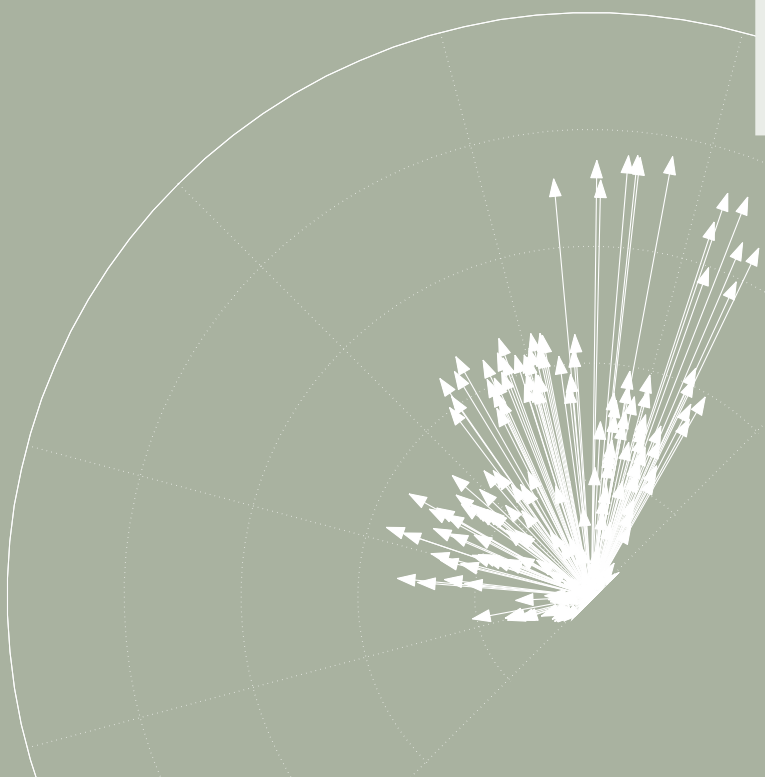
MANUAL DE COMPUTAÇÃO EVOLUTIVA E METAHEURÍSTICA

ANTÓNIO GASPAR-CUNHA
RICARDO TAKAHASHI
CARLOS HENGGELER ANTUNES
COORDENADORES



IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDITORA ufmg

CAPÍTULO 13

Distâncias Generalizadas: Espaços Combinatórios

Eduardo G. Carrano

Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais

Os algoritmos desenvolvidos para otimização de variáveis contínuas realizam a busca explorando propriedades favoráveis do espaço vetorial, no qual estão definidas as variáveis (Luenberger, 1969). Por exemplo, algoritmos de direção de busca se baseiam na possibilidade de definir uma direção que possibilita decrescimento da função, enquanto que, algoritmos de otimização unidimensional dependem da definição de uma distância ao longo de uma direção (Luenberger, 1984).

Mesmo no caso de problemas não-convexos, não-diferenciáveis ou multimodais, as características do espaço vetorial podem ser exploradas por algoritmos de otimização. Por exemplo, direções em que existam tendências de decrescimento, são exploradas de alguma forma por algoritmos evolucionários (AE's), como *Algoritmos Genéticos Reais* (Takahashi et al., 2003), *Algoritmos Particle Swarm Optimization* (Kennedy, 1997; Suganthan, 1999), *Algoritmos de Evolução Diferencial* (Storn e Price, 1997), etc. Uma vez definidos em problemas contínuos, estes algoritmos herdam diretamente o conceito de vizinhança inerente ao espaço vetorial, tornando a busca local implicitamente definida.

Estas operações baseadas nas propriedades do espaço vetorial garantem aos AE's contínuos uma certa generalidade: um mesmo algoritmo geralmente pode ser aplicado em problemas com funções objetivo de diferentes características.

Por outro lado, algoritmos evolucionários em que as soluções são representadas por códigos, como por exemplos algoritmos evolucionários voltados à otimização de problema discretos, não herdam a

capacidade de exploração do espaço vetorial. Uma particularidade destes algoritmos é que sua estrutura de vizinhança no espaço de busca é definida pela forma com que os operadores evolutivos percorrem as soluções, o que depende da representação adotada e a forma com que a operação é definida. A escolha do conjunto representação evolucionária/medida de distância ou representação evolucionária/operadores evolucionários afeta consideravelmente o desempenho do algoritmo de otimização (Moraglio et al., 2007; Carrano, Fonseca, Takahashi, Pimenta e Neto, 2007; Carrano et al., 2010). Essa alteração de desempenho pode ser creditada ao ordenamento relativo das soluções, que é dependente da medida de distância adotada ou os operadores utilizados, e afeta diretamente a forma com que o espaço é percorrido. O uso de uma medida de distância adequada para a representação adotada, e a escolha de operadores que se guiem eficientemente por essa distância, favorecem a otimização, pois permitem a realização de buscas aproximadamente convexas no espaço de busca, reduzindo a ocorrência de ótimos locais (Moraglio e Poli, 2004; Moraglio et al., 2007; Carrano, 2007; Carrano et al., 2010).

Este capítulo discute a generalização de AE's para o tratamento de problemas discretos utilizando operações inspiradas na otimização contínua. São relatados esforços no sentido de criar métricas de distância adequadas e operadores que percorrem as estruturas de vizinhança induzidas por essas métricas. Essas adaptações geralmente permitem a extensão de ferramentas que dependem de operações inicialmente restritas à espaços contínuos, como por exemplo exploração de vizinhanças, interpolações de soluções, etc. O capítulo apresenta a seguinte estrutura:

- A Sec. 1 discute o conceito de *panorama de aptidão*, identificando as relações que existem entre espaço de códigos / domínio da função / imagem da função.
- A Sec. 2 apresenta alguns exemplos de distâncias generalizadas propostas na literatura para problemas discretos.
- A Sec. 3 discute o conceito de operadores geométricos, proposto em (Moraglio e Poli, 2004, 2005; Moraglio et al., 2007), que visa unificar os conceitos que suportam o desenvolvimento de algoritmos evolucionários.
- Na Sec. 4 são apresentadas operações contínuas generalizadas para otimização de redes em árvore. Essas operações são utilizadas para construção de um algoritmo genético (GA), que pode ser aplicado na solução de problemas dessa classe.
- Na Sec. 5 é apresentado um estudo de caso, onde o GA proposto na Sec. 4 é aplicado na solução do *Optimal Communication Spanning Tree* (OCST).

1. Panorama de Aptidão

Seja um espaço de configuração qualquer $\mathcal{C} = (\mathcal{G}, Nhd)$, onde $\mathcal{G}$ é o conjunto de configurações sintáticas (genótipos no caso de algoritmos evolucionários) e $Nhd : G \to 2^G$ é uma função sintática de vizinhança que mapeia para cada configuração $G \in \mathcal{G}$ o conjunto de configurações vizinhas, que podem ser obtidos a partir da mesma utilizando um operador sintático de modificação unitária. A definição de modificação unitária é estabelecida através da escolha de uma função sintática de vizinhança, que identifica quais configurações diferem da configuração G em uma unidade. O operador sintático de modificação unitária deve ser simétrico ($y \in Nhd(x) \Leftrightarrow x \in Nhd(y)$) e conexo (uma configuração pode ser transformada em qualquer outra através da aplicação desse operador um número finito de vezes).

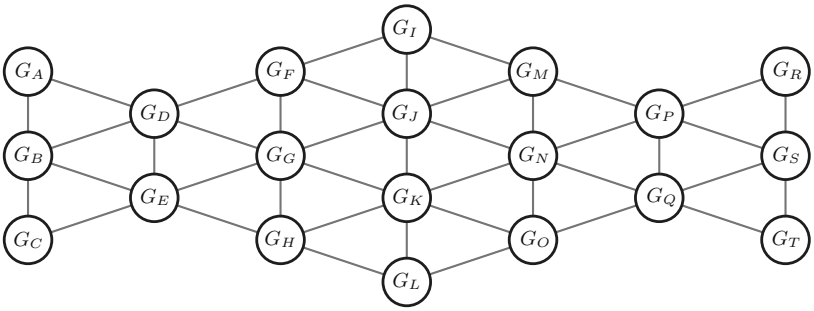
A função sintática de vizinhança confere ao espaço $\mathcal{C}$ uma estrutura de vizinhança que pode ser representada por um grafo não direcionado $W(V, E)$, onde V é o conjunto de vértices que representa as configurações e E é o conjunto de arestas que representa as relações entre essas configurações.

Uma mudança na função sintática de vizinhança adotada implica na alteração da estrutura de vizinhança do problema, e consequentemente altera a forma com que o operador de modificação unitária percorre o espaço. A Fig. 13.1 ilustra dois grafos induzidos por duas funções sintáticas de vizinhança distintas. Em ambos os casos G_G foi utilizado como exemplo para mostrar a variação do conjunto de vizinhos conforme muda a função sintática de vizinhança. Neste exemplo fica clara a alteração na forma de percorrer os espaço do operador de modificação unitária:

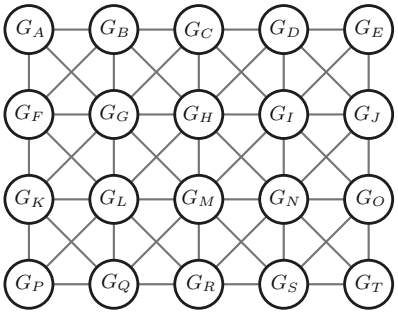
- Para a função nhd_1, a configuração G_E é vizinha de G_G, e portanto pode ser obtida através de uma única modificação unitária;

- Já para a função nhd_2, a configuração G_E não é vizinha de G_G, e são necessárias no mínimo três modificações unitárias para encontrar G_E a partir de G_G: $G_G \rightarrow G_C \rightarrow G_D \rightarrow G_E$.

Uma vez que a estrutura de vizinhança é simétrica e conexa, o espaço definido é um espaço métrico, munido de uma função de distância d, que é induzida pela função sintática de vizinhança (Bäck et al., 1997). Então, se torna equivalente definir $\mathcal{C} = (\mathcal{G}, Nhd)$ ou $\mathcal{C} = (\mathcal{G}, d)$. Entretanto, é importante notar que as distâncias originadas de um espaço de configuração sintático têm uma natureza sintática, e portanto podem não obedecer exatamente os axiomas que definem uma medida de distância em espaços vetoriais.



(a) $Nhd_1(G_G) = \{G_D, G_E, G_F, G_H, G_J, G_K\}$



(b) $Nhd_2(G_G) = \{G_A, G_B, G_C, G_F, G_H, G_K, G_L, G_M\}$

Figura 13.1: Exemplos de grafos induzidos por duas funções sintáticas de vizinhança distintas, Nhd_1 e Nhd_2.

Um "mapeamento de representação" é uma função $r : \mathcal{G} \rightarrow \mathcal{S}$ que associa cada configuração sintática G a uma solução formal $S \in \mathcal{S}$, onde $\mathcal{S}$ é o espaço onde está definido o domínio da função em estudo. Idealmente esse mapeamento deve ser bijetivo, mas existem casos em que $|\mathcal{G}| \neq |\mathcal{S}|$.

O *panorama de aptidão* (do inglês *fitness landscape*) F é um par $(\mathcal{C}, f)$, onde $\mathcal{C} = (\mathcal{G}, d)$ é um espaço de configurações e $f : \mathcal{G} \rightarrow \mathcal{R}$ é uma função que associa cada configuração sintática G ao seu valor de aptidão, definido em $\mathcal{R}$. A função de aptidão é um valor, geralmente positivo, que pode ou não coincidir com o valor de função objetivo correspondente ao genótipo G. Para efeito de simplicidade, será assumido neste texto que estes valores são idênticos, e portanto o valor de aptidão f é uma composição do "mapeamento de representações" r e a função objetivo g: $f = r \circ g$.[1]

Localidade no Espaço de Representações *vs.* Localidade no Domínio da Função: Reflexo na Imagem da Função

Como já foi discutido ao longo dessa sessão, a representação adotada para codificação das soluções e a função de distância escolhida no espaço de representações afetam o panorama da função objetivo e, consequentemente, o ordenamento relativo das soluções no espaço de busca do AE. Neste caso, é alterado o panorama da função objetivo, já que a mesma passa a ter como parâmetro de entrada não um ponto do domínio da função, mas uma representação, que mapeia um ponto no domínio da função através da função de mapeamento r:

$$f(S \in \mathcal{S}) = f(r(G \in \mathcal{G})) \tag{13.1}$$

A estrutura de vizinhança imposta pela escolha da representação e da função sintática de vizinhança atua fortemente na forma com que os operadores percorrem os espaço, o que pode ser favorável ou desfavorável ao processo de otimização. Um exemplo de como a representação medida de distância afetam a estrutura de vizinhança de um problema é apresentado na sequência, para um problema contínuo onde as soluções são representadas por cadeias binárias em um AE qualquer.

Exemplo 1

Seja o seguinte problema de minimização:

$$x^* = \arg\min_x \left[2 + (x-3)^2 + \sin(5 \cdot \pi \cdot x)\right] \quad , \quad 0 \leq x \leq 8 \tag{13.2}$$

Suponha que este problema seja resolvido utilizando um AE com representação binária de 5 bits por variável, e utilizando como operador de modificação unitário um operador que se orienta pela distância de Hamming (i.e. este operador é capaz de gerar a partir de um ponto P pontos cuja distância de Hamming em relação a P seja 1) (Hamming, 1950, 1980).

A distância de Hamming é brevemente apresentada aqui para facilitar o entendimento do exemplo. A mesma será discutida com mais detalhes na Sec. 2.

Distância de Hamming: Sejam duas cadeias binárias A e B, compostas de n bits cada uma. A distância de Hamming entre estas duas cadeias, $D_H(A, B)$, é determinada por:

$$D_H(A, B) = \sum_{i=1}^{n} \delta_H(A_i, B_i) \tag{13.3}$$

[1] Descrição adaptada de (Moraglio e Poli, 2004).

onde:

$$\delta_H(A_i, B_i) = \begin{cases} 0 & \text{se} \quad A_i = 0 \quad e \quad B_i = 1 \\ 0 & \text{se} \quad A_i = 1 \quad e \quad B_i = 0 \\ 1 & \text{se} \quad A_i = 0 \quad e \quad B_i = 0 \\ 1 & \text{se} \quad A_i = 1 \quad e \quad B_i = 1 \end{cases}$$

Uma vez que foi utilizada uma representação de 5 bits, o espaço de busca desse problema é composto por 32 soluções possíveis. A Fig. 13.2 ilustra essa função e os 32 pontos de espaço de busca. A representação binária, o valor real e o valor de função objetivo correspondentes à cada uma dessas soluções são apresentados na Tab. 13.1.

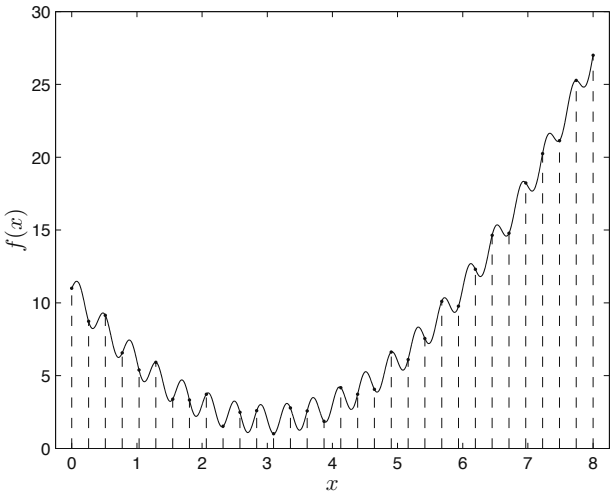


Figura 13.2: Exemplo 1: Função objetivo e pontos correspondentes à representação de 5 bits.

A análise de vizinhança desse conjunto de soluções pode ser realizada no domínio da função, $\mathbb{R}$, onde é avaliada a função objetivo, ou no espaço de genótipos, $\mathbb{B}^5$, onde as soluções são representados no AE[2].

Vizinhança no domínio da função: Devido à discretização proposta, no domínio da função cada solução candidata possui dois vizinhos, sendo um a esquerda e um a direita (exceto os dois extremos, que possuem um vizinho cada). Por exemplo, o ponto $x_4 = 1.0323$ é vizinho dos pontos $x_3 = 0.7742$ e $x_5 = 1.2903$. Este ponto constitui um mínimo local, uma vez que seu valor de função objetivo é menor que o de todos os seus vizinhos $f(x_4) = 5.3867$, $f(x_3) = 6.5599$ e $f(x_5) = 5.9115$. A distância euclideana entre x_4 e seus vizinhos é $D_E(x_3, x_4) = D_E(x_4, x_5) = 0.2581$.

Vizinhança no espaço de genótipos/representações: Uma vez que neste caso a estrutura de vizinhança é definida pela distância de Hamming, cada solução candidata possui 5 vizinhos, sendo cada vizinho obtido através da variação de cada um dos bits da cadeia individualmente. Neste espaço, o ponto $x_4 = 00100$ é vizinho de $x_5 = 00101$, $x_6 = 00110$, $x_0 = 00000$, $x_{12} = 01100$ e $x_{20} = 10100$. A distância de Hamming entre x_4 e seus vizinhos é necessariamente 1. Por outro lado, no espaço de genótipos x_4 não é vizinho de x_3, pois $D_H(x_3, x_4) = 3$. Outro fato importante

[2] $x \in \mathbb{B}$ se $x \in \{0, 1\}$.

a ser observado é que, este ponto não constitui um mínimo local no espaço de genótipos, uma vez que x_6 e x_{12} têm valor de função objetivo menor que o mesmo ($f(x_4) = 5.3867$, $f(x_6) = 3.3824$ e $f(x_{12}) = 1.0106$).

Tabela 13.1: Exemplo - Código binário, valor real correspondente e valor de função

	x_{bin}	x	$f(x)$		x_{bin}	x	$f(x)$
x_0	00000	0.0000	11.0000	x_{16}	10000	4.1290	4.1725
x_1	00001	0.2581	8.7274	x_{17}	10001	4.3871	3.7227
x_2	00010	0.5161	9.1377	x_{18}	10010	4.6452	4.0552
x_3	00011	0.7742	6.5599	x_{19}	10011	4.9032	6.6210
x_4	00100	1.0323	5.3867	x_{20}	10100	5.1613	6.0999
x_5	00101	1.2903	5.9115	x_{21}	10101	5.4194	7.5539
x_6	00110	1.5484	3.3824	x_{22}	10110	5.6774	10.1063
x_7	00111	1.8065	3.3234	x_{23}	10111	5.9355	9.7684
x_8	01000	2.0645	3.7238	x_{24}	11000	6.1935	12.2999
x_9	01001	2.3226	1.5211	x_{25}	11001	6.4516	14.6384
x_{10}	01010	2.5806	2.4752	x_{26}	11010	6.7097	14.7732
x_{11}	01011	2.8387	2.5973	x_{27}	11011	6.9677	18.2283
x_{12}	01100	3.0968	1.0106	x_{28}	11100	7.2258	20.2518
x_{13}	01101	3.3548	2.7773	x_{29}	11101	7.4839	21.1370
x_{14}	01110	3.6129	2.5769	x_{30}	11110	7.7419	25.2767
x_{15}	01111	3.8710	1.8608	x_{31}	11111	8.0000	27.0000

Com base no exemplo apresentado é possível perceber que a utilização de diferentes representações e medidas de distância afeta o comportamento da função, devido à alteração da estrutura de vizinhança do algoritmo. A análise desse aspecto é fundamental no estudo de problemas discretos, uma vez que, nestes problemas, as soluções candidatas são sempre representadas por códigos. O espaço em que estes códigos estão definidos não é necessariamente dotado de uma caracterização geométrica trivial, o que pode dificultar a construção de operadores que explorem adequadamente as propriedades do mesmo. O estudo de métricas de distância generalizadas, aplicáveis a estes problemas, pode facilitar substancialmente a construção de AE's eficientes para cada problema. Na Sec. 2 são apresentados alguns exemplos de métricas de distância aplicáveis a problemas discretos.

É importante destacar que existem vários trabalhos que estudam o estabelecimento de analogias entre os espaços contínuo e discreto. Como exemplo, pode-se citar:

- (Christodoulakis et al., 2005; Guo et al., 2008) utilizam métricas específicas para detectar similaridades de cadeias de caracteres.
- (Lafage e Lang, 2005) propõem uma família de medidas de distâncias que são utilizadas para comparação de preferências em tomadas de decisão.
- (Galton, 1999) propõe uma estrutura topológica para um espaço discreto genérico. A estrutura topológica proposta caracteriza uma topologia espacial viável do ponto de vista matemático.
- (Galton, 2000, 2003) introduzem o conceito de "deslocamentos aproximadamente contínuos" em espaços discretos.
- (Billera et al., 2001) apresentam um modelo que permite representar árvores filogenéticas binárias no espaço contínuo. A representação proposta permite o calculo de distâncias entre árvores filogenéticas, além de facilitar o processo de combinação das mesmas.
- (Moraglio e Poli, 2004) apresentam um ambiente (*framework*) para desenvolvimento de algoritmos evolucionários gerais, em que se torna necessária apenas a definição de uma representação evolucionária e uma medida de distância ou uma representação evolucionária e um operador de

mutação. Este trabalho é discutido na Sec. 3, juntamente com outros trabalhos dos mesmos autores.

- (Deza e Huang, 1998) apresentam métricas para estimar diferenças entre permutações.
- (Carrano et al., 2010) propõem uma representação que transforma redes genéricas em vetores embarcados no espaço contínuo $\mathbb{R}^n$. Com base nessa representação se torna possível estimar a posição relativa, produto escalar e a distância entre redes. Esses conceitos permitem a extensão de operadores de buscas local, interpolação e busca unidimensional para o problema de otimização de redes. Este trabalho é melhor discutido nas Secs. 2 e 4.

2. Exemplos de Medidas de Distância em Problemas Combinatórios

Conforme citado anteriormente, a existência de medidas de distância capazes de proporcionar um melhor ordenamento das soluções em problemas discretos pode ser útil na construção de AE eficientes. Isso serviu de motivação para vários estudos, cujo intuito principal era estabelecer medidas de distância adequadas aos mais diversos tipos de problemas discreto. No entanto, antes de introduzir algumas dessas métricas, é importante apresentar as propriedades necessárias para que uma métrica possa ser considerada uma medida de distância.

Definio 13.1 (Métrica de Distância) *Uma função D, que calcula um escalar que expressa a diferença entre dois vetores quaisquer $\overrightarrow{A}, \overrightarrow{B} \in \mathbb{X}^m$ (onde $\mathbb{X}^m$ é um espaço qualquer m-dimensional qualquer), define uma métrica de distância se, e somente se, a mesma cumpre os axiomas A.1, A.2, A.3 e A.4, listados abaixo:*

- *A.1: $D(\overrightarrow{A}, \overrightarrow{B}) \geq 0$*
- *A.2: $D(\overrightarrow{A}, \overrightarrow{B}) = 0$ se, e somente se, $\overrightarrow{A} = \overrightarrow{B}$ (identidade)*
- *A.3: $D(\overrightarrow{A}, \overrightarrow{B}) = D(\overrightarrow{B}, \overrightarrow{A})$ (simetria)*
- *A.4: $D(\overrightarrow{A}, \overrightarrow{C}) \geq D(\overrightarrow{A}, \overrightarrow{B}) + D(\overrightarrow{A}, \overrightarrow{C})$ (desigualdade triangular)*

Métricas em que o axioma A.2 é relaxado, de forma que a distância zero não implica em igualdade, mas a igualdade sempre implica em distância 0, são chamadas pseudo-métricas.

Algumas métricas de distância propostas na literatura para problemas discretos são apresentadas na sequência dessa seção:

distância de Hamming Generalizada

A distância de Hamming (Hamming, 1950, 1980), apresentada na Sec. 1 para estimar a diferença entre duas cadeias binárias, pode ser estendida para um caso mais geral, onde podem ser comparadas duas cadeias de qualquer tipo, desde que possuam o mesmo número de elementos. Essa extensão é apresentada em (13.4).

$$D_{HG}(A, B) = \sum_{i=1}^{n} \delta_{HG}(A_i, B_i) \tag{13.4}$$

onde:

$$\delta_{HG}(A_i, B_i) = \begin{cases} 0 & \text{se} \quad A_i = B_i \\ 1 & \text{se} \quad A_i \neq B_i \end{cases} .$$

Com essa adaptação se torna possível comparar vetores de qualquer tipo, como por exemplo cadeias de caracteres, inteiros, reais, binários, etc. A única restrição imposta é igualdade de comprimento dos dois vetores.

Exemplo 2

Sejam duas cadeias, X_1 e X_2, de seis números inteiros cada, conforme apresentado abaixo:

$$\begin{aligned} X_1 &= [1\ 3\ 2\ 4\ 6\ 5] \\ X_2 &= [2\ 3\ 4\ 1\ 6\ 5] \end{aligned}$$

Conforme a Eq. (13.4), a distância de Hamming Generalizada entre X_1 e X_2 é igual ao número de posições em que essas cadeias são diferentes, portanto:

$$D_{HG}(X_1, X_2) = 3$$

Distância *Labeling-Independent*

No problema de particionamento de grafos, que é um problema de agrupamento, cada solução é representada por um vetor X de tamanho n (n é o número de elementos que devem ser agrupados). Em cada posição i desse vetor se encontra um inteiro X_i, que indica a que grupo o termo i pertence.

Exemplo 3

Um exemplo de solução candidata para o problema de agrupamento é ilustrado abaixo:

$$X = [2\ 1\ 2\ 1\ 3\ 2]$$

Nesta solução existem seis elementos, que são divididos em 3 grupos distintos: os elementos 2 e 4 pertencem ao grupo 1; os elementos 1, 3 e 6 pertencem ao grupo 2 e o elemento 5 pertence ao grupo 3.

Este tipo de representação é altamente redundante, uma vez que qualquer solução que agrupe o elemento 5 em um grupo, os elementos 1, 3 e 6 em outro grupo e os elementos 2 e 4 em um terceiro representa, do ponto de vista prático, a mesma solução que X. A solução $Y = [3\ 2\ 3\ 2\ 1\ 3]$ é um exemplo de código que representa uma solução idêntica à X no domínio da função.

A alta redundância desse tipo de representação faz com que a distância de Hamming não seja adequada para comparar soluções em problemas de agrupamento, uma vez que a mesma geralmente vai detectar diferenças em soluções que do ponto de vista prático são idênticas. Visando contornar as limitações da distância de Hamming para este problema, (Moraglio et al., 2007) propõem a distância *labeling-independent*, conforme descrita abaixo:

Sejam A e B duas cadeias de tamanho fixo n, representadas no alfabeto K-ário ($A, B \in \mathcal{U} = \{1, 2, \ldots, K\}$), e δ_{LI} uma métrica definida em $\mathcal{U}$. A distância *labeling-independent* relacionada à métrica δ_{LI} é definida como segue:

$$D_{LI}(A, B) = \min_{\sigma \in \sum_K} \delta_{LI}(A, B_\sigma) \tag{13.5}$$

onde:
$\sum_K$ é o conjunto de todas as permutações possíveis de comprimento K;
B_σ é a cadeia B permutada conforme a permutação σ.

Continuação do Exemplo 3

No Exemplo 3, apresentado anteriormente nessa seção, os elementos estão divididos em 3 grupos distintos. Com isso, existem 6 permutações possíveis para cada solução: $\sigma_1 = [1\ 2\ 3]$, $\sigma_2 = [1\ 3\ 2]$, $\sigma_3 = [2\ 1\ 3]$, $\sigma_4 = [2\ 3\ 1]$, $\sigma_5 = [3\ 1\ 2]$ e $\sigma_6 = [3\ 2\ 1]$. Portanto, X possui seis permutações cujo sentido prático da solução é exatamente o mesmo: $X_{\sigma_1} = [1\ 2\ 1\ 2\ 3\ 1]$, $X_{\sigma_2} = [1\ 3\ 1\ 3\ 2\ 1]$, $X_{\sigma_3} = [2\ 1\ 2\ 1\ 3\ 2]$, $X_{\sigma_4} = [2\ 3\ 2\ 3\ 1\ 2]$, $X_{\sigma_5} = [3\ 1\ 3\ 1\ 2\ 3]$ e $X_{\sigma_6} = [3\ 2\ 3\ 2\ 1\ 3]$ representam a mesma solução.

A distância de Hamming entre X e cada uma dessas cadeias é:

$$D_{HG}(X,Y) = \begin{cases} 6 & \text{se} \quad Y = X_{\sigma_1} \\ 6 & \text{se} \quad Y = X_{\sigma_2} \\ 0 & \text{se} \quad Y = X_{\sigma_3} \\ 6 & \text{se} \quad Y = X_{\sigma_4} \\ 6 & \text{se} \quad Y = X_{\sigma_5} \\ 6 & \text{se} \quad Y = X_{\sigma_6} \end{cases}$$

Com base nesses valores, fica fácil constatar que a distância de Hamming é ineficiente para esse tipo de problema: $D_{HG}(X,Y)$ assume valor 0 apenas para o caso em que Y é exatamente igual à X ($Y = X_{\sigma_3}$), e assume valor máximo, 6, para todas as outras permutações de X. Este não é comportamento esperado para este tipo de problema, uma vez que, do ponto de vista prático, X, X_{σ_1}, X_{σ_2}, X_{σ_3}, X_{σ_4}, X_{σ_5} e X_{σ_6} representam a mesma solução.

Já a distância *labeling-independent* entre X e qualquer uma de suas permutações é necessariamente 0. Caso seja considerada uma permutação efetivamente distinta de X, como por exemplo $Z = [1\ 1\ 1\ 2\ 2\ 3]$, essa distância é não nula. Neste caso, a permutação de Z que mais se aproxima de X é $Z_{\sigma_3} = [2\ 2\ 2\ 1\ 1\ 3]$, e portanto:

$$D_{LI}(X,Y) = 2$$

A distância *labeling-independent* elimina completamente o problema da redundância da representação utilizada em problemas de agrupamento, uma vez que ela testa todas as $K!$ permutações possíveis da cadeia B. (Moraglio et al., 2007) demonstram que a distância *labeling-independent* obedece todos os axiomas que caracterizam uma pseudo-métrica, e pode ser calculada de forma eficiente utilizando o Método Húngaro (Kuhn, 2004).

Distância de Levenshtein (*Edit Distance*)

A distância de Levenshtein, também conhecida como distância de edição (do inglês *edit distance*), é utilizada para detecção de similaridades em cadeias de caracteres (Guo et al., 2008). Esta distância calcula o mínimo de operações de edição necessárias para transformar uma cadeia de caracteres A em outra cadeia de caracteres B. As operações de edição compreendem inserções, apagamentos e substituições, sendo que todas possuem peso 1 (Christodoulakis et al., 2005). Matematicamente, a distância de Levenshtein pode ser definida como:

$$D_{LV}(A,B) = \min_{\sigma_A \in \sum_A\ ,\ \sigma_B \in \sum_B} \delta_{LV}\left(A_{\sigma_A}, B_{\sigma_B}\right) \tag{13.6}$$

onde:
$\sum_A$ e $\sum_B$ são os conjuntos de todos os alinhamentos das cadeias A e B respectivamente;
S_{σ_i} é a cadeia S alinhada conforme o alinhamento σ_i.

Suponha que se calcular a distância de Levenshtein entre as cadeias *VINTNER* e *WRITERS*. Neste caso é necessário determinar o número mínimo de operações de edição necessárias para transformar

a palavra VINTNER na palavra WRITERS, ou vice-versa. O alinhamento que apresenta a menor número de operações para estas duas palavras é:

$$A^* : \begin{cases} \texttt{V-INTNER-} \\ \texttt{WRI-T-ERS} \end{cases}$$

Neste caso são necessárias 5 operações: uma substituição (`V` pelo `W` na primeira posição), duas inserções (`R` na segunda posição e `S` na última posição) e dois apagamentos (`N` na quarta posição e `N` na sexta posição). Portanto, a distância de Levenshtein entre essas cadeias de caracteres é 5:

$$D_{LV}(\mathit{VINTNER}, \mathit{WRITERS}) = 5.$$

Com base neste exemplo, fica fácil perceber que outros alinhamentos conduziriam à outras distâncias, necessariamente maiores que a encontrada para o alinhamento ótimo A^*. Outros possíveis alinhamentos para essas duas cadeias são apresentados abaixo:

$$A_1 : \begin{cases} \texttt{VINTNER} \\ \texttt{WRITERS} \end{cases} \quad A_2 : \begin{cases} \texttt{VINTNER-} \\ \texttt{-WRITERS} \end{cases} \quad A_3 : \begin{cases} \texttt{-VINTNER-} \\ \texttt{WRIT--ERS} \end{cases}$$

Para estes três alinhamentos o número de operações de edição necessárias é sempre seis.[3]

A distância de Levenshtein cumpre todos os axiomas estabelecidos no início da Sec. 2 para definição de medidas de distância. Apesar do alto número de alinhamentos possíveis, segundo (Moraglio et al., 2006) o alinhamento ótimo da distância de Levenshtein pode ser calculado de forma eficiente utilizando programação dinâmica.

Distância *T-norma*

(Carrano et al., 2010) propõem um procedimento que permite transformar soluções de problemas combinatórios modelados por redes em vetores esparsos, embutidos no espaço vetorial $\mathbb{R}^m$ de Hilbert. A transformação das soluções é baseada em uma definição de norma, chamada de *T-norma* (do inglês *T-norm*) que é possível devido à definição de um produto escalar estabelecido adequadamente. Esse procedimento é apresentado na sequência.

Seja $G(\mathcal{V}, \mathcal{A}, \mathcal{B})$ um grafo não direcionado com $|\mathcal{V}| = n$ vértices, $|\mathcal{A}| = m$ arestas e $|\mathcal{B}| = b$ tipos de arestas distintos e ordenados (1 se refere ao tipo de aresta mais fraco e b o tipo de aresta mais forte). Seja N um sub-grafo de G que contém necessariamente todos os vértices do conjunto $\mathcal{V}$. É possível representar N como um vetor $\overrightarrow{N}$ embutido no espaço $\mathbb{R}^m$ de Hilbert, conforme proposto em (13.7).

$$\overrightarrow{N} = \sum_{i=1}^{m} w_i^N \cdot (p + \phi(N_i)) \cdot \overrightarrow{e_i} \tag{13.7}$$

onde:
w_i^N é o *peso topológico* da aresta i na árvore N. Este peso é um valor real positivo se a conexão i existe ou 0 caso contrário;
N_i é o tipo de aresta da conexão i na árvore N, com $0 \leq N_i \leq b$. Os tipos de arestas devem estar ordenados de forma que as conexões "mais fortes" recebam valores maiores de N_i;
$\phi(\cdot)$ é uma função monotônica crescente, com $\phi(1) = 0$ no caso de problemas *multi-branch*, e é a função constante nula $\phi(\cdot) \equiv 0$ em problemas *mono-branch*;
p é um fator constante tal que $p \geq \phi(b)$;
$\overrightarrow{e_i}$ é o i-ésimo vetor da base canônica.

[3] Exemplo extraído de (Moraglio et al., 2006).

Esta expressão atribui à cada aresta possível no grafo uma dimensão do espaço $\mathbb{R}^m$. As referências (Carrano, 2007; Carrano et al., 2010) mostram que os vetores definidos por essa expressão obedecem os axiomas necessários para definição de vetores em espaços vetoriais (Lima, 1995).

Esse procedimento permite o tratamento tanto de problemas *mono-branch*, em que existe apenas um tipo de aresta possível, quanto de problemas *multi-branch*, em que uma mesma aresta pode assumir mais de um tipo. Problemas *mono-branch* ocorrem em casos em que se necessita definir apenas se duas arestas estão conectadas ou não, como por exemplo em redes de computadores e redes de sensores sem fio. Problemas *multi-branch* surgem em casos em que dois vértices podem ser conectados fisicamente por arestas de diferentes tipos, como por exemplo redes de energia elétrica, onde existem condutores de diferentes bitolas, ou redes de água, onde podem ser utilizados tubos de diferentes diâmetros.

No caso de problemas *mono-branch*, em que $\phi(\cdot) \equiv 0$, o valor de p pode ser definido, sem perda de generalidade, como $p = 1$, reduzindo (13.7) para:

$$\overrightarrow{N} = \sum_{i=1}^{m} w_i^N \cdot \overrightarrow{e_i} \tag{13.8}$$

Se a conexão i não existe, então w_i^N recebe 0. Caso contrário, é atribuído ao *peso topológico* w_i^N um valor que pode ser interpretado como a importância topológica da conexão i na rede N. Por exemplo, em redes estruturadas em árvores, é esperado que mudanças próximas à raiz tenham maior impacto no fluxo da rede que mudanças próximas as folhas. Portanto, deve-se associar valores maiores de w_i^N para as conexões mais próximas à raiz. No caso geral, as regras específicas para definir w_i^N devem ser elaboradas tendo em conta as características de cada problema.

Em (13.7), a função crescente $\phi(N_i)$, que tem como argumento o inteiro N_i, expressa a força do tipo de conexão da aresta i (tipos de arestas mais fortes são associados à valores mais altos de N_i e, portanto, de $\phi(N_i)$ também). A desigualdade $p \geq \phi(b)$ faz com que o efeito de acrescentar ou retirar uma aresta seja maior que o efeito de mudar um tipo de aresta, na avaliação da expressão $(p + \phi(N_i))$.

Sejam agora A e B duas redes quaisquer, tais que:

$$\begin{aligned} \overrightarrow{A} &= \sum_{i=1}^{m} w_i^A \cdot (p + \phi(A_i))\, \overrightarrow{e_i} \\ \overrightarrow{B} &= \sum_{i=1}^{m} w_i^B \cdot (p + \phi(B_i))\, \overrightarrow{e_i} \end{aligned}$$

O vetor diferença que representa a posição relativa de A em relação a B pode ser definido como:

$$\begin{aligned} \overrightarrow{(B,A)} &= \overrightarrow{A} - \overrightarrow{B} \\ &= \sum_{i=1}^{m} \left[w_i^A \cdot \phi(A_i) - w_i^B \cdot \phi(B_i) + p \cdot \left(w_i^A - w_i^B \right) \right] \overrightarrow{e_i} \end{aligned} \tag{13.9}$$

Um produto escalar dos vetores $\overrightarrow{A}$ e $\overrightarrow{B}$ pode ser definido por:

$$\overrightarrow{A} \cdot \overrightarrow{B} = \sum_{i=1}^{m} \left[w_i^A \cdot (p + \phi(A_i)) \right] \cdot \left[w_i^B \cdot (p + \phi(B_i)) \right] \tag{13.10}$$

Este produto escalar define uma norma:

$$\overrightarrow{A} \cdot \overrightarrow{A} = \| \overrightarrow{A} \|^2 \tag{13.11}$$

que, por sua vez, define uma distância:

$$D_{TN}(A,B) = \sqrt{\sum_{i=1}^{m} \left(w_i^B \cdot \phi(B_i) - w_i^A \cdot \phi(A_i) + p \cdot \left(w_i^B - w_i^A \right) \right)^2} \tag{13.12}$$

Exemplo 5

Seja o grafo $G(\mathcal{V}, \mathcal{A}, \mathcal{B})$, com $|\mathcal{A}| = 10$, $|\mathcal{V}| = 19$ e $\mathcal{B} = \{1\}$, apresentado na Fig. 13.3 e na Tab. 13.2. Busca-se nesse grafo redes com topologia de árvore, assumindo o vértice 1 como raiz.

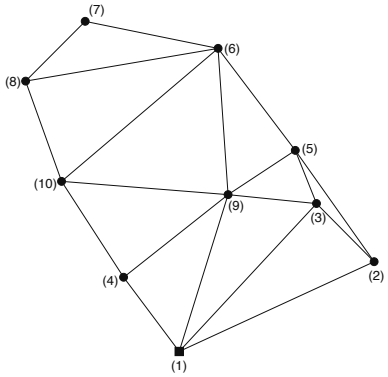


Figura 13.3: Exemplo 5: Grafo de busca com 10 vértices e 19 arestas.

Tabela 13.2: Exemplo 5: Componentes do espaço de arestas possíveis

Componente	$\overrightarrow{e_1}$	$\overrightarrow{e_2}$	$\overrightarrow{e_3}$	$\overrightarrow{e_4}$	$\overrightarrow{e_5}$	$\overrightarrow{e_6}$	$\overrightarrow{e_7}$	$\overrightarrow{e_8}$	$\overrightarrow{e_9}$	$\overrightarrow{e_{10}}$
do vértice:	1	1	1	1	2	2	3	3	4	4
para o vértice:	2	3	4	9	3	5	5	9	9	10

Componente	$\overrightarrow{e_{11}}$	$\overrightarrow{e_{12}}$	$\overrightarrow{e_{13}}$	$\overrightarrow{e_{14}}$	$\overrightarrow{e_{15}}$	$\overrightarrow{e_{16}}$	$\overrightarrow{e_{17}}$	$\overrightarrow{e_{18}}$	$\overrightarrow{e_{19}}$
do vértice:	5	5	6	6	6	6	7	8	9
para o vértice:	6	9	7	8	9	10	8	10	10

As Figs. 13.4a e 13.4b representam duas árvores X_1 e X_2, possíveis em G, que foram geradas aleatoriamente. Os pesos dos vértices nessas redes foram calculados de forma inversamente proporcional à soma dos custos das arestas no caminho entre o vértice e raiz, conforme definido em (13.13).

$$s_x^N = 1 - \frac{{d_{x,root}}^N}{\max_j \left({d_{j,root}}^N\right)} \tag{13.13}$$

Onde:
${d_{x,root}}^N$ é o custo somado das arestas no caminho entre a aresta x e a raiz *root* na árvore N.

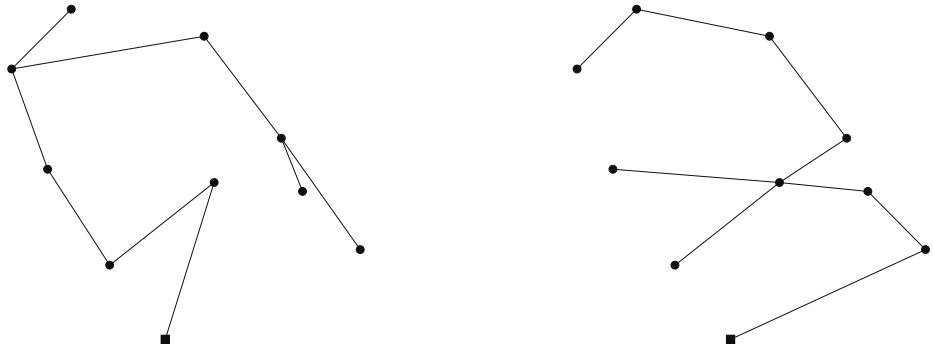
Com base nessa equação é possível encontrar o peso de todas os vértices em X_1 e X_2:

$$s^{X_1} = [\,1,00 \;\; 0,00 \;\; 0,08 \;\; 0,69 \;\; 0,14 \;\; 0,28 \;\; 0,37 \;\; 0,45 \;\; 0,82 \;\; 0,57\,]$$

$$s^{X_2} = [\,1,00 \;\; 0,74 \;\; 0,64 \;\; 0,37 \;\; 0,44 \;\; 0,27 \;\; 0,11 \;\; 0,00 \;\; 0,54 \;\; 0,35\,]$$

O peso das arestas é calculado com base nos pesos dos nós, como mostra a Eq. (13.14):

$$w_i^N = \frac{s_a^N + s_b^N}{2} \tag{13.14}$$



(a) X_1 (b) X_2

Figura 13.4: Exemplo 5: Árvores aleatórias.

O pesos dos nós s^{X_1} e s^{X_2} são inseridos em (13.14), o que leva ao peso topológico das arestas em X_1 e X_2:

$$\overrightarrow{w^{X_1}} = 0,91\overrightarrow{e_4} + 0,07\overrightarrow{e_6} + 0,11\overrightarrow{e_7} + 0,75\overrightarrow{e_9} + 0,63\overrightarrow{e_{10}} + 0,21\overrightarrow{e_{11}} + 0,36\overrightarrow{e_{14}} + 0,41\overrightarrow{e_{17}} + 0,51\overrightarrow{e_{18}}$$

$$\overrightarrow{w^{X_2}} = 0,87\overrightarrow{e_1} + 0,69\overrightarrow{e_5} + 0,59\overrightarrow{e_8} + 0,45\overrightarrow{e_9} + 0,35\overrightarrow{e_{11}} + 0,49\overrightarrow{e_{12}} + 0,19\overrightarrow{e_{13}} + 0,05\overrightarrow{e_{17}} + 0,44\overrightarrow{e_{19}}$$

Como o caso considerado é *mono-branch*, estes pesos são então utilizados na Eq. (13.8) para encontrar os vetores $\overrightarrow{X_1}$ e $\overrightarrow{X_2}$, que representam as redes X_1 e X_2 respectivamente:

$$\overrightarrow{X_1} = 2,73\overrightarrow{e_4} + 0,21\overrightarrow{e_6} + 0,34\overrightarrow{e_7} + 2,26\overrightarrow{e_9} + 1,88\overrightarrow{e_{10}} + l0,63\overrightarrow{e_{11}} + 1,09\overrightarrow{e_{14}} + 1,22\overrightarrow{e_{17}} + 1,53\overrightarrow{e_{18}}$$

$$\overrightarrow{X_2} = 2,62\overrightarrow{e_1} + 2,07\overrightarrow{e_5} + 1,76\overrightarrow{e_8} + 1,36\overrightarrow{e_9} + 1,05\overrightarrow{e_{11}} + 1,46\overrightarrow{e_{12}} + 0,56\overrightarrow{e_{13}} + 0,16\overrightarrow{e_{17}} + 1,32\overrightarrow{e_{19}}$$

Com os vetores $\overrightarrow{X_1}$ e $\overrightarrow{X_2}$ é possível calcular a posição relativa da rede X_1 em relação à X_2, usando (13.9):

$$\begin{aligned}\overrightarrow{(X_2, X_1)} \;=\; & 2,62\overrightarrow{e_1} + 2,73\overrightarrow{e_4} - 2,07\overrightarrow{e_5} + 0,21\overrightarrow{e_6} + 0,34\overrightarrow{e_7} - \\ & - 1,76\overrightarrow{e_8} + 0,90\overrightarrow{e_9} + 1,88\overrightarrow{e_{10}} - 0,42\overrightarrow{e_{11}} - 1,46\overrightarrow{e_{12}} - \\ & - 0,56\overrightarrow{e_{13}} + 1,09\overrightarrow{e_{14}} + 1,06\overrightarrow{e_{17}} + 1,53\overrightarrow{e_{18}} - 1,32\overrightarrow{e_{19}}\end{aligned}$$

Finalmente, o número escalar que define a distância entre X_1 e X_2 pode ser estimado calculado a norma Euclideana de $\overrightarrow{(X_2, X_1)}$, conforme definido em (13.12):[4]

$$D_{TN}(A, B) = \sqrt{\sum_{i=1}^{19} \left(\overrightarrow{(X_2, X_1)}_i\right)^2} = 5.93$$

Em (Carrano, 2007; Carrano et al., 2010) é demonstrado que o produto escalar e a norma definidos obedecem todos os axiomas de produtos escalares e métricas de distância respectivamente. Também é mostrado que o processo de transformação de redes proposto permite a extensão de entidades inerentes a espaços contínuos, como por exemplo, direções, distâncias, vizinhanças, projeções, etc., que ficam definidas para redes em geral. Mais detalhes sobre como essa metodologia é explorada para construção de operadores evolucionários são apresentados na Sec. 4.

[4] Exemplo adaptado de (Carrano et al., 2010).

3. Operadores Topológicos / Geométricos

A referência (Moraglio e Poli, 2004) constitui o marco inicial de um estudo que objetiva esclarecer as relações entre representações, operadores genéticos, estruturas de vizinhança e métricas de distância em algoritmos evolutivos. Com base nesse estudo é proposta uma nova classe de operadores, chamados de operadores topológicos e re-batizados de operadores geométricos em referências subsequentes. São propostos dois operadores, ε-mutação topológica uniforme e cruzamento topológico uniforme, que se enquadram nesta classe. Os pontos mais relevantes deste trabalho são discutidos na sequência dessa seção.

Operadores Topológicos

Antes de apresentar os operadores topológicos é importante esclarecer algumas definições de operadores evolucionários:[5]

Um operador evolucionário g-ário[6] OP utiliza g soluções pais, $p_1, p_2, \ldots, p_g$ para construir uma solução filha c, de acordo com alguma função de distribuição de probabilidade condicional dada:

$$Pr\{OP(p_1, \ldots, p_g) = c\} = f_{OP}(c|p_1, \ldots, p_g) \tag{13.15}$$

onde:
$Pr\{x\}$ é a probabilidade de ocorrência do evento x.

Definio 13.2 *O conjunto imagem de um operador OP é o conjunto de todos os possíveis filhos que podem ser obtidos por OP, quando os pais são $p_1, p_2, \ldots, p_g$, com probabilidade não nula:*

$$Im[OP(p_1, \ldots, p_g)] = \{c \in \mathcal{S} | f_{OP}(c|p_1, \ldots, p_g) > 0\} \tag{13.16}$$

Definio 13.3 *Um operador unário M é uma ε-mutação topológicamutação!topológica se, e somente se, $Im[M(p)] \subseteq B(p, \varepsilon)$, onde ε é o menor real possível para o qual essa afirmação é verdadeira e $B(p, \varepsilon)$ é uma vizinhança de tamanho ε, definida por uma métrica de distância D considerada.*

Definio 13.4 *Um operador binário CX é um cruzamento topológico se, e somente se, $Im[CX(p_1, p_2)] \subseteq [p_1; p_2]$. Isto significa que em um cruzamento topológico a solução filha se encontra necessariamente entre os indivíduos pais, tendo em conta uma métrica de distância D considerada.*

Tendo em conta essas definições se torna possível estabelecer dois operadores topológicos:

Definio 13.5 (ε-mutação topológica uniforme) *A ε-mutação topológica uniforme $UM\varepsilon$ é uma ε-mutação topológica onde todo filho z que se encontra a uma distância ε do pai p tem a mesma probabilidade de ser encontrado:*

$$f_{UM\varepsilon}(z|p) = Pr\{UM\varepsilon = z|P = p\} = \frac{\zeta(z \in B(p, \varepsilon))}{|B(p, \varepsilon)|} \tag{13.17}$$

$$Im[UM\varepsilon(x)] = \{z \in \mathcal{S} | f_{UM\varepsilon}(z|p) > 0\} = B(p, \varepsilon) \tag{13.18}$$

onde:
ζ é um função que retorna 1 se o argumento é verdadeiro ou 0 caso contrário.

[5] Definições extraídas de (Moraglio e Poli, 2004).
[6] A mutação é sempre um operador unário, enquanto o cruzamento é geralmente um operador binário.

Definio 13.6 (Cruzamento topológico uniforme) *O cruzamento topológico uniforme UX é um cruzamento topológico onde todo filho z que se encontra entre os pais p_1 e p_2 tem a mesma probabilidade de ser encontrado:*

$$f_{UX}(z|p_1,p_2) = Pr\{UX = z|P1 = p_1, P2 = p_2\} = \frac{\zeta(z \in [p_1;p_2])}{|\,[p_1;p_2]\,|} \tag{13.19}$$

$$Im[UM(p_1,p_2)] = \{z \in \mathcal{S}|f_{UX}(z|p_1,p_2) > 0\} = [p_1;p_2] \tag{13.20}$$

Uma característica importante da mutação topológica uniforme e o cruzamento topológico uniforme é que os mesmos são operadores definidos sem qualquer referência à representação adotada, e portanto são extensíveis à qualquer tipo de representação.

(Moraglio e Poli, 2004) demonstram que o espaço de configurações $\mathcal{C}$ pode ser definido de forma equivalente pelo par formado por $\mathcal{G}$ com qualquer uma das seguintes entidades: *(a)* A função de vizinhança Nhd; *(b)* O grafo de vizinhança $W(V,E)$; *(c)* A função de distânciadistância!função de d; *(d)* A mutação topológica uniforme UM; e, *(e)* O cruzamento topológico uniforme UX.

Isso torna possível a definição de um espaço métrico a partir da definição apenas da representação, que define $\mathcal{G}$, e um operador de mutação a de P_2 ou remover uma aresta não existente em P_2 do grafo atual (inicialmente P_1). Este método descreve um caminho que conecta as duas soluções, e se assemelha ao procedimento de *path-relinking* (Glover et al., 2000).

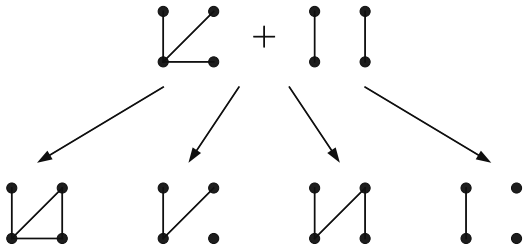


Figura 13.5: Exemplo de operador de cruzamento topológico, definido a partir de ε-mutações topológicas

A proposta dos operadores topológicos é extremamente útil na construção de algoritmos evolucionários para problemas discretos, uma vez que a mesma cria uma *framework* geral de desenvolvimento de algoritmos e permite a construção dos mesmos tendo como base apenas uma representação e um operador de mutação, ou uma representação e uma medida de distância. Segundo os autores, este *framework* traz avanços no estudo da unificação da teoria de operadores evolucionários:

Generalização: A mutação topológica e o cruzamento topológico estabelecem procedimentos gerais para construção de operadores de mutação e cruzamento, baseados em simples operações de modificação unitária.

Unificação: Estudos indicam que vários operadores propostos para diferentes representações obedecem as propriedades de operadores topológicos.

Independência de representação: Todas as definições apresentadas são independentes da representação evolucionária adotada. Isso constitui um avanço em relação à estudos anteriores, que geralmente desenvolvem operadores e ferramentas que são aplicados somente à representações específicas. O *framework* desenvolvido pode ser estendido para qualquer representação, desde que haja um operador de mutação, estrutura de vizinhança ou medida de distância definida.

Clarificação: As conexões entre representações, operadores, vizinhanças e distâncias se tornam mais claras com o uso de operadores topológicos.

Análise: Dado uma representação e um conjunto de operadores existentes, fica fácil analisar se este conjunto se enquadra nas definições topológicas. Caso sim, as propriedades conhecidas dos operadores topológicos são válidas para os operadores analisados.

É importante salientar que, mesmo com o uso de operadores topológicos, o desempenho de um algoritmo evolutivo continua estritamente relacionado com a escolha do conjunto representação / medida de distância adotada (Carrano, Fonseca, Takahashi, Pimenta e Neto, 2007). Isso faz com que, mesmo que sejam utilizados operadores topológicos, ainda seja necessário um estudo das representações e operadores mais adequados para o problema.

Aplicações de Operadores Topológicos / Geométricos

Vários estudos foram desenvolvidos utilizando os conceitos estabelecidos para os operadores topológicos (Moraglio e Poli, 2004). Nestes trabalhos foi priorizada a análise de operadores de cruzamento, que foram re-batizados de cruzamentos geométricos. A escolha pelo estudo dos operadores de cruzamento é justificada pelos autores pelo fato desses operadores constituírem um ponto crítico no desenvolvimento de AG's. Uma breve descrição das principais referências que utilizam essa classe de operadores é apresentada na sequência:

- (Moraglio e Poli, 2004) apresentam os primeiros avanços no estudo de operadores geométricos. São apresentadas as definições iniciais sobre as quais os operadores geométricos estão estabelecidos (vide o início dessa sessão). É apresentada uma análise sobre os operadores binários clássicos e é mostrado que os mesmos são operadores topológicos sob a distância de Hammingdistância!de Hamming.
- (Moraglio e Poli, 2005) utilizam três métricas específicas para permutações para construir três cruzamento geométricos aplicáveis à problemas de otimização de permutações.
- (Moraglio et al., 2006) propõem um cruzamento geométrico aplicável ao problema de comparação de cadeias de DNA. Este operador se baseia em uma medida de comparação de cadeias de caracteres para gerar a orientação do espaço métrico gerado para o problema.
- (Moraglio et al., 2007) apresentam três cruzamentos para o problema de particionamento de grafos. Estes operadores são baseados na distância *Labeling-Independent* e na distância de Hamming (vide Sec. 2).
- (Moraglio e Togelius, 2009) propõem um algoritmo *Particle Swarm Optimization* (PSO) com inércia utilizando os princípios de operadores geométricos. Este algoritmo foi comparado com três outros na solução de problemas clássicos de otimização.

4. Operações Contínuas Generalizadas para Otimização de Redes em Árvore

O procedimento de transformação de redes em vetores embutidos no espaço contínuo (Sec. 2) é utilizado em (Carrano, 2007; Carrano et al., 2010) para a construção de várias operações aplicadas à problemas de otimização de árvores. Estas operações são extensões de operações contínuas, que só são possíveis devido a metodologia proposta em (Carrano, 2007; Carrano et al., 2010). As mesmas são brevemente descritas ao longo dessa sessão.

Interpolação em Linha

O conceito de posição relativa apresentado na Eq. (13.9) pode ser usado para definir uma "interpolação em linha" para duas redes em árvore. Sejam S e D duas árvores definidas em um grafo de busca $G(\mathcal{V}, \mathcal{A}, \mathcal{B})$ (S é a rede de partida e D é a rede direção). É possível gerar redes correspondentes à pontos sobre uma "trajetória aproximada", que começa em S e termina em D, conforme descrito abaixo:

O problema é definido por:
Dadas as redes de partida e destino S e D, um escalar $\gamma \in [0,1]$ e uma tolerância $\epsilon > 0$ busca-se uma árvore R, associada ao vetor $\overrightarrow{R}$ que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + \gamma \cdot (\overrightarrow{S,D})$, tal que: $\left\| \overrightarrow{(R,P)} \right\| \leq \epsilon$.

Tal rede pode ser gerada através do seguinte procedimento:

1. Definir o contador $j = 1$, e a rede resultante atual $R_j = S$;
2. Encontrar uma nova rede R_{j+1} que é obtida a partir da remoção de uma aresta de S e a inserção de uma aresta de D em R_j, de forma que $\left\| \overrightarrow{(R_{j+1},P)} \right\| < \left\| \overrightarrow{(R_j,P)} \right\|$ e R_{j+1} mantenha factibilidade;
3. Se $\left\| \overrightarrow{(R_{j+1},P)} \right\| < \epsilon$, ir para o passo 4. Caso contrário, fazer $j = j + 1$ e ir para o passo 2;
4. Se o problema é *mono-branch*, fazer $R = R_{j+1}$ e parar; caso contrário, vá para o passo 5;
5. Se o problema é *multi-branch*, encontrar uma rede R_{j+2} que tem a mesma topologia de R_{j+1}, mas tem tipos de conexões tais que $\left\| \overrightarrow{(R_{j+2},P)} \right\|$ atinge o mínimo valor possível, com a topologia da rede fixa e os tipos de conexões variáveis. Fazer $R = R_{j+2}$ e parar.

A ideia por trás deste processo é fazer com que R caminhe ao longo do segmento $\overrightarrow{(S,D)}$, de forma que as componentes do vetor $\overrightarrow{(R,D)}$ decresçam progressivamente, enquanto que as componentes do vetor $\overrightarrow{(S,R)}$ cresçam progressivamente, e a soma das normas destes vetores seja próxima a norma de $\overrightarrow{(S,D)}$. Este processo iterativo é necessário para garantir que R_j caminhe sobre o conjunto de soluções factíveis entre S e D. Após a definição da topologia de R (após o passo 3), o ajuste dos tipos de conexões em problemas *multi-branch* é direto, e pode ser realizado de forma não-iterativa.

Este procedimento faz com que R possua apenas arestas do conjunto $S \cup D$. Isso faz desse procedimento um operador de cruzamento geométrico, conforme descrito na Sec. 3. No entanto, o mesmo ainda apresenta evoluções em relação ao cruzamento geométrico básico:

- O peso topológico da aresta w_i^N presente na *T-norma* faz com que diferentes perturbações unitárias[7] tenham diferentes impactos no valor calculado da distância. Com isso, diferentes escolhas de arestas a serem removidas/inseridas podem levar a soluções que distanciam ou aproximam de P, que é o vetor ideal que se encontra sobre o segmento $\overrightarrow{(S,D)}$ para o escalar γ. Da forma com que está proposta, a interpolação em linha busca encontrar soluções que se aproximem o máximo possível de P, o que faz que essa interpolação se aproxime muito da interpolação de vetores no espaço $\mathbb{R}^n$, cujas propriedades são conhecidas e favoráveis para a otimização.

- O fato dessa interpolação trabalhar de forma semelhante à interpolação contínua permite a extensão de outras ferramentas que dependem de interpolações, como por exemplo buscas unidimensionais, que serão apresentadas na próxima seção.

[7] Uma perturbação unitária pode ser compreendida como a remoção de uma aresta de S e inclusão de uma aresta de D em R_j

- Uma vez que a distância medida entre as soluções é continua, a interpolação permite uma definição mais adequada do parâmetro γ, que pode assumir uma conotação aproximadamente contínua (dois valores próximos de γ tendem a conduzir a duas árvores levemente diferentes). Isso não seria possível se o algoritmo se orientasse pela distância de Hamming, uma vez que a mesma tem natureza absolutamente discreta.

Busca Unidimensional

A operação de interpolação descrita torna possível a implementação de métodos otimização unidimensional. Esse procedimento pode ser implementado utilizando o algoritmo de seção áurea conforme descrito abaixo:

O problema é definido por:
Dadas as redes de partida de destino S e D e uma tolerância $\epsilon > 0$, busca-se uma árvore R, associada ao vetor $\overrightarrow{R}$ que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + \alpha^* \cdot (\overrightarrow{S,D})$, tal que: $\left\|(\overrightarrow{R,P})\right\| \leq \epsilon$. α^* é determinado otimizando a função $f(\cdot)$ a partir de S no segmento $(\overrightarrow{S,D})$: $\alpha^* = \arg\min_{\alpha} f\left(\left\lceil \overrightarrow{S} + \alpha \cdot (\overrightarrow{S,D}) \right\rceil\right)$, onde $\left\lceil \overrightarrow{X} \right\rceil$ é a árvore cujo vetor associado é o mais próximo possível do vetor $\overrightarrow{X}$.

Tal rede pode ser gerada através do seguinte procedimento:

1. Definir $a = 0$, $b = 1$, $x_a = 0,382$ e $x_b = 0,618$;
2. Encontrar uma árvore R_a que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + x_a \cdot (\overrightarrow{S,D})$;
3. Encontrar uma árvore R_b que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + x_b \cdot (\overrightarrow{S,D})$;
4. Encontrar $f_a = f(R_a)$ e $f_b = f(R_b)$;
5. Enquanto $(b - a) > \epsilon$:
   (a) Se $f_a < f_b$:
      i. Fazer $b = x_b$, $x_b = x_a$, $x_a = b - 0,618 \cdot (b - a)$, $f_b = f_a$;
      ii. Encontrar uma árvore R_a que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + x_a \cdot (\overrightarrow{S,D})$;
      iii. Encontrar $f_a = f(R_a)$.
   (b) Caso contrário:
      i. Fazer $a = x_a$, $x_a = x_b$, $x_b = a + 0,618 \cdot (b - a)$, $f_a = f_b$;
      ii. Encontrar uma árvore R_b que é aproximadamente situada na posição espacial $\overrightarrow{P} = \overrightarrow{S} + x_b \cdot (\overrightarrow{S,D})$;
      iii. Encontrar $f_b = f(R_b)$.
6. Se $f_a \leq f_b$, faça $R = R_a$; caso contrário, faça $R = R_b$.

Assim como no espaço de variáveis contínuas, esta otimização unidimensional garante encontrar a rede ótima R no segmento que une S à D, dado que a função objetivo $f(R)$ é unimodal neste segmento. No caso de comportamento não-unimodal, os limites unimodais máximo e mínimo definem os limites de erro na minimização.

Geração de Pontos Aleatórias a Distâncias Pré-definidas

A geração aleatória de um ponto que se encontra a uma distância pré-definida de um dado ponto é uma operação fundamental em vários algoritmos estocásticos. Esta operação pode ser realizada utilizando os conceitos propostos conforme descrito abaixo:

O problema é definido por:
Dada a rede inicial S e um escalar γ, que define a distância desejada, busca-se uma árvore R cuja distância em relação à S seja tão próximo quanto possível de γ: $\left\|\overrightarrow{(R,P)}\right\| \approx \gamma$.

Tal rede pode ser gerada através do seguinte procedimento:

1. Definir $\overrightarrow{R_x} = \overrightarrow{S}$;
2. Determinar $\mathcal{O}$, que é o conjunto de arestas que podem ser removidas de R_x tal que a rede resultante (após ser corrigida para manutenção da factibilidade) cumpra $\left\|\overrightarrow{(R_x,S)}\right\| \leq \gamma$;
3. Se $|\mathcal{O}| > 0$, ir para o passo 4; caso contrário ir para o passo 7;
4. Escolher aleatoriamente uma conexão $o \in \mathcal{O}$ e remover de R_x;
5. Determinar $\mathcal{I}$, que é o conjunto de conexões que re-estabelecem a factibilidade de R_x e cumprem $\left\|\overrightarrow{(R_x,S)}\right\| \leq \gamma$;
6. Escolher aleatoriamente uma conexão $i \in \mathcal{I}$ e inserir em R_x;
7. Fazer $R = R_x$;
8. Se o problema é *mono-branch*, parar; caso contrário, vá para o passo 9;
9. Determinar o conjunto de conexões de R que podem ter seus tipos mudados, de forma que a rede resultante tenha o menor valor possível para $\left|\left\|\overrightarrow{(R,S)}\right\| - \gamma\right|$. Realizar estas alterações.

Este procedimento permite a definição de um operador de mutação, que é capaz e realizar não só buscas locais, mas também buscas locais quando necessário (em algoritmos cujo raio de mutação é inversamente proporcional à sua aptidão por exemplo). A utilização da *T-norma* também permite relacionar de forma mais adequada a remoção de uma aresta com seu impacto na função objetivo.

Vizinhança e Busca Local

O conceito de distância proporcionado pela *T-norma* induz imediatamente uma estrutura de vizinhança, que pode ser definida como:

$$\mathcal{V}\left(\overrightarrow{X_0},\epsilon\right) = \left\{X \;\middle|\; \left\|\overrightarrow{(X_0,X)}\right\| < \epsilon\right\} \tag{13.21}$$

em torno do vetor $\overrightarrow{X_0}$ que representa a árvore X_0, com raio ϵ. Neste caso a busca local fica imediatamente definida como a busca dentro de uma dada vizinhança.

Vários métodos de busca local podem ser propostos para o espaço vetorial onde as redes estão embutidas. Uma possível estratégia seria:

O problema é definido por:
Dada a rede inicial S e um escalar γ, que define o raio de vizinhança, busca-se uma árvore R que representa o mínimo local na região em torno de S.

Tal rede pode ser gerada através do seguinte procedimento:

1. Fazer $R = S$ e $f(R) = f(S)$;

2. Determinar $\mathcal{O}$, que é o conjunto de arestas que podem ser removidas de R tal que a rede resultante R_x (após ser corrigida para manutenção da factibilidade) cumpra $\left\|\overrightarrow{(R_x, S)}\right\| \leq \gamma$;
3. Para cada $o \in \mathcal{O}$:
   (a) Fazer $R_x = R$;
   (b) Remover a aresta o de R_x;
   (c) Determinar $\mathcal{I}$, que é o conjunto de conexões que re-estabelecem a factibilidade de R_x e mantém $\left\|\overrightarrow{(R_x, S)}\right\| \leq \gamma$;
   (d) Para cada $i \in \mathcal{I}$:
      i. Inserir a aresta i em R_x e encontrar $f(R_x)$;
      ii. Se $f(R_x) < f(R)$:
         A. Fazer $R = R_x$ e $f(R_x) = f(R)$.
      iii. Remover i de $\mathcal{I}$.
   (e) Remover o de $\mathcal{O}$.
4. Se o problema é *mono-branch*, parar; caso contrário, ir para o passo 05;
5. Determinar $\mathcal{M}$, que é o conjunto de arestas que podem ter seus tipos modificados em R tal que a rede resultante R_x cumpra $\|\overrightarrow{(R_x, S)}\| \leq \gamma$;
6. Para cada $m \in \mathcal{M}$:
   (a) Para cada $b \in \mathcal{B}$:
      i. Fazer $R_x = R$;
      ii. Substituir a aresta m em R_x por uma aresta do tipo b e encontrar $f(R_x)$;
      iii. Se $f(R_x) < f(R)$:
         A. Fazer $R = R_x$ e $f(R_x) = f(R)$.
   (b) Remover m de $\mathcal{M}$.

Este procedimento, apesar de garantir a obtenção do ótimo local na vizinhança de S, é muito caro computacionalmente, e pode se tornar inviável mesmo em problemas com um número moderado de arestas. Este custo pode ser reduzido utilizando uma busca local aproximada, conforme descrito abaixo:

O problema é definido por:
Dada a rede inicial S, o raio de vizinhança γ e um número máximo de avaliações sem melhoria N_{EV}, busca-se uma árvore R que ao menos aproxima o mínimo local na região em torno de S.

Tal rede pode ser gerada através do seguinte procedimento:

1. Fazer $R = S$ e $f(R) = f(S)$;
2. Fazer $n = 0$;
3. Calcular $\rho = \mathrm{U}(0,1) \cdot \gamma$ onde $\mathrm{U}(0,1)$ é um número aleatório com distribuição uniforme no intervalo $[0,1]$;
4. Gerar uma rede R_x a partir de R, tal que $\left\|\overrightarrow{(R_x, S)}\right\| \approx \rho$ (esta rede é gerada utilizando o procedimento de geração de árvores aleatórias a distâncias pré-definidas) e encontrar $f(R_x)$;
5. Se $f(R_x) < f(R)$:
   (a) Fazer $R = R_x$ e $f(R_x) = f(R)$ e voltar ao passo 2.
6. Caso contrário, fazer $n = n + 1$;
7. Se $n \geq N_{EV}$, então parar; caso contrário, voltar ao passo 3.

Este método permite o controle do custo computacional do método de busca local, mesmo em problemas com muitos vértices. Ainda pode ser acrescentado um parâmetro que controle o número máximo de avaliações de função executado pelo método. No entanto, é importante salientar que este método não garante a obtenção do ótimo local exato, uma vez que o mesmo não percorre todos os pontos da vizinhança de S.

Como no caso contínuo, o conceito de busca local proposto tem como intenção encontrar o mínimo local da função. No espaço vetorial onde as redes estão definidas, o mínimo local fica definido como o ponto que apresenta valor mínimo de função objetivo dentro da vizinhança considerada, ao invés de uma vizinhança arbitrariamente pequena, que é geralmente considerada em problemas contínuos.

Algoritmo Genético Proposto

As operações apresentadas nesta seção foram utilizadas para criação de 5 operadores genéticos, sendo 3 de cruzamento e 2 mutação:

crv1: Dadas duas soluções pais p_1 e p_2, a operação de interpolação em linha é utilizada para gerar duas soluções filhas c_1 e c_2, para dois aleatórios uniformes $\gamma_1, \gamma_2 \in [0, 1]$.

crv2: Dadas duas soluções pais, p_1 e p_2, a operação de busca unidimensional é utilizada para gerar um filho c_1. O segundo filho c_2 é gerado utilizando a operação de interpolação em linha, considerado um aleatório uniforme $\gamma \in [0, 1]$.

crv3: Dada uma solução pai p e a melhor solução encontrada até o momento p^*, a operação de busca unidimensional é utilizada para gerar uma solução filha c.

mut1: Dada uma solução pai p e um raio de mutação γ, a operação de geração de pontos aleatórios a distância pré-definidas é utilizada para gerar uma solução filha c.

mut2: Dada uma solução pai p e um raio de vizinhança γ, a operação de busca local (com custo computacional controlado) é utilizada para gerar uma solução filha c.

Estes operadores foram divididos em duas classes: binários, que são operadores que geram duas soluções filhas (crv1 e crv2); e, unários, que são operadores que geram uma solução filha (crv3, mut1 e mut2). Os mesmos foram utilizados na construção de um algoritmo genético, denominado *GANet*, cujo esquema básico é apresentado na sequência:

inicialização:

- gerar uma população inicial factível;
- definir $p_{bin} = 0.80$ e $p_{un} = 0.45$;
- avaliar a função objetivo para cada objetivo da população inicial;
- encontrar o valor de aptidão para cada indivíduo, utilizado ranking linear.

enquanto não critério de parada

operações binárias:

- dividir a população em pares (aleatoriamente, com distribuição uniforme);
- **para** cada par de indivíduos:
  - gerar um número aleatório $0 \leq r_{bin} \leq 1$ com distribuição uniforme;
  - **se** $r_{bin} \leq p_{bin}$ **então**

    escolher aleatoriamente entre **crv1** e **crv2** e executar o operador.

operações unárias:

- **para** cada indivíduo:
  - gerar um número aleatório $0 \leq r_{un} \leq 1$ com distribuição uniforme;
  - **se** $r_{un} \leq p_{un}$ **então**
    escolher aleatoriamente entre **mut1**, **mut2** e **crv3** e executar o operador.

avaliação de função e seleção:

- avaliar a função objetivo para cada indivíduo novo gerado;
- encontrar o valor de aptidão para todos indivíduos, utilizando ranking linear;
- realizar a seleção utilizando a Amostragem Estocástica Universal (do inglês *Stochastic Universal Sampling* ou simplesmente SUS) (Baker, 1987).

end enquanto

Aplicações bem sucedidas deste algoritmo, ou variantes do mesmo, podem ser encontradas em:

- (Carrano, Guimaraes, Takahashi, Neto e Campelo, 2007), onde um Algoritmo Imunológico Artificial (que utiliza os operadores propostos) é utilizado para projetar sistemas de distribuição de energia elétrica considerado incertezas na evolução de carga;
- (Pereira et al., 2009), onde três variações do algoritmo proposto são aplicadas na solução do *Degree-Constrained Minimum Spanning Tree Problem* (DCMST);
- (Carrano et al., 2010), onde este algoritmo é aplicado na solução do *Optimal Communication Minimum Spanning Tree* (OCST) e do *Quadratic Minimum Spanning Tree* (QMST).

No presente trabalho, este algoritmo foi aplicado na solução de um problema clássicos de projeto de redes: o *Optimal Communication Spanning Tree*. O algoritmo *GANet* foi comparado com outros cinco GA's clássicos, desenvolvidos para otimização de problemas de árvores *mono-branch*. Os resultados observados são apresentados na Sec. 5.

5. Estudo de Caso - Optimal Communication Spanning Tree (OCST)

No problema *Optimal Communication Spanning Tree*, ou OCST (Hu, 1974), busca-se à árvore geradora de mínimo custo, onde este custo é baseado nos requisitos de comunicação entre os pares de nós do sistema (Soak et al., 2006). Este problema foi inicialmente provado como NP-difícil (NP-hard) (Garey e Johnson, 1979) e posteriormente como MAX SNP-difícil (MAX SNP-hard) (Papadimitriou e Yannakakis, 1991; Soak et al., 2006). A formulação do problema é apresentada abaixo.

A função objetivo é definida como:

$$\min \sum_{i,j \in \mathcal{V}} R_{i,j} \cdot C^X_{i,j} \tag{13.22}$$

onde:
$C^X_{i,j}$ é a soma dos custos das arestas no caminho $i - j$;
$R_{i,j}$ são os requisitos de comunicação entre i e j;
$\mathcal{V}$ é o conjunto de vértices.
A única restrição é a restrição topológica que exige que a rede seja uma árvore.

O algoritmo proposto na seção anterior foi utilizado para solução de instâncias euclideanas de 25 nós (300 variáveis) e 50 nós (1225 variáveis) do OCST. Essas instâncias foram geradas considerando

grafos completos, com os nós gerados aleatoriamente dentro de um quadrado 50×50. Os requisitos de comunicação entre os nós foram gerados aleatoriamente, com distribuição uniforme, no intervalo $[0, 200]$.

O *GANet* e outros cinco GA's, baseados em representações específicas para redes em árvores, foram aplicados na solução dessas instâncias:

$\mathcal{A}_1$: *GANet* (Carrano, 2007; Carrano et al., 2010);

$\mathcal{A}_2$: GA utilizando representação *Characteristic Vector* (Rothlauf, 2005);

$\mathcal{A}_3$: GA utilizando representação *Prüfer Numbers* (Prüfer, 1918);

$\mathcal{A}_4$: GA utilizando representação *Network Random Keys* (Routhlauf et al., 2002);

$\mathcal{A}_5$: GA utilizando representação *Edge Sets* (Raidl e Julstrom, 2003);

$\mathcal{A}_6$: GA utilizando representação *Node Biased* (Palmer e Kershenbaum, 1994a);

Detalhes sobre estes cinco últimos algoritmos podem ser encontrados em (Carrano, Fonseca, Takahashi, Pimenta e Neto, 2007).

Todos os algoritmos, incluindo o *GANet*, foram testados utilizando o mesmo conjunto de parâmetros:

- Número de execuções: 30 execuções por instância;
- Tamanho da população: 50 indivíduos;
- p_{crz} / p_{bin}: 0.80 por par;
- p_{mut} / p_{un}: 0.45 por indivíduo;
- Método de seleção: SUS com ranking linear ($s = 2$);
- Critério de parada: 1.500 avaliações de função sem melhoria para o melhor indivíduo da população.

Foram consideradas três critérios de mérito (C_{MER}) para avaliação dos algoritmos:

- *bst*: valor de função objetivo da melhor solução encontrada dentre 30 execuções do algoritmo;
- f^*: média ($\overline{X}$) e desvio padrão (s) da variável aleatória definida pelo valor de função objetivo da melhor solução encontrada em cada execução do algoritmo;
- n_{FE}: média ($\overline{X}$) e desvio padrão (s) da variável aleatória definida pelo número de avaliações de função gasto para atingir a melhor solução encontrada em cada execução do algoritmo.

Os resultados observados nos seis algoritmos, para as instâncias de 25 e 50 nós, são apresentados na Tab. 13.3.

Sob o ponto de vista do valor de função objetivo observado, é possível ver que $\mathcal{A}_1$ apresenta o melhor desempenho, com uma variação muito baixa entre a melhor solução obtida e a média observada. No caso de 50 nós, $\mathcal{A}_1$ foi o único a obter uma solução com valor de função objetivo abaixo de $1,115E+7$. Outro aspecto interessante nessa instância é que a média de $\mathcal{A}_1$ é melhor que a melhor execução de todos os outros algoritmos estudados.

Já sob o ponto de vista de avaliações de função é possível observar que $\mathcal{A}_1$ é mais rápido que todos os outros algoritmos para a instância de 25 nós, e ligeiramente mais lento que $\mathcal{A}_3$ e $\mathcal{A}_6$ para a instância de 50 nós. No entanto, essa menor velocidade na instância de 50 nós pode ser justificada pela maior capacidade de convergência do algoritmo: uma vez que $\mathcal{A}_1$ atinge melhores soluções que os outros algoritmos, é razoável que o mesmo gaste mais avaliações de função que eles, uma vez que melhores soluções são mais difíceis de ser alcançadas.[8]

[8] Descrição e resultados adaptados de (Carrano et al., 2010).

Tabela 13.3: OCST: Resultados obtidos

Inst.		OCST: 25 nós		OCST: 50 nós	
Alg.	C_{MER}	$\overline{X}$	s	$\overline{X}$	s
$\mathcal{A}_1$	bst	2,5620E+6	-	1,1087E+7	-
	f^*	2,5624E+6	0,0061	1,1099E+7	0,0040E+6
	n_{FE}	1,0326E+4	1,8556E+3	6,5959E+4	2,0396E+4
$\mathcal{A}_2$	bst	2,5620E+6	-	1,1415E+7	-
	f^*	2,5764E+6	0,1089	1,1926E+7	0,3523E+6
	n_{FE}	3,2167E+4	5,4320E+3	7,6265E+4	1,5854E+4
$\mathcal{A}_3$	bst	2,6360E+6	-	1,1809E+7	-
	f^*	2,8545E+6	1,9991	1,2971E+7	0,8964E+6
	n_{FE}	1,2711E+4	2,6864E+3	6,1056E+4	1,3665E+4
$\mathcal{A}_4$	bst	2,6039E+6	-	1,1782E+7	-
	f^*	2,8027E+6	1,5887	1,4197E+7	1,6379E+6
	n_{FE}	2,4872E+4	8,1407E+3	7,2492E+4	1,6789E+4
$\mathcal{A}_5$	bst	2,5620E+6	-	1,1182E+7	-
	f^*	2,5742E+6	0,0989	1,1640E+7	0,2277E+6
	n_{FE}	2,0735E+4	2,9372E+3	9,0303E+4	1,3878E+4
$\mathcal{A}_6$	bst	2,5620E+6	-	1,1322E+7	-
	f^*	2,6242E+6	0,3163	1,1653E+7	0,3070E+6
	n_{FE}	1,5632E+4	4,8127E+3	5,3659E+4	1,0203E+4

Testes de Panorama de Aptidão

O OCST foi utilizado para estudar como a *T-norma* afeta o ordenamento relativo das soluções no espaço de busca, e qual o impacto disso na função objetivo. O panorama de aptidão gerado pelo ordenamento relativo proporcionado pela *T-norma* é comparada com o panorama de aptidão observado quando as soluções são ordenadas conforme a distância de Hamming. Essa comparação é feita através do seguinte procedimento:

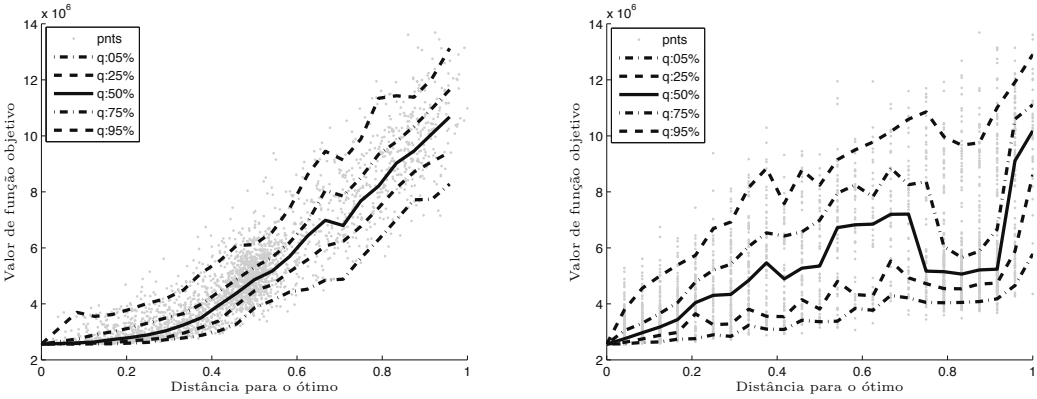
1. Gerar um conjunto de k redes factíveis aleatórias;
2. Encontrar o distância entre cada rede e a melhor solução conhecida para o problema usando a *T-norma* e a distância de Hamming;
3. Ordenar as redes em ordem crescente de distância para o ótimo;
4. Normalizar a distância para ambas as métricas;
5. Dividir o intervalo de distâncias, que estão normalizadas entre 0 e 1, em l sub-intervalos;
6. Para cada intervalo, encontrar os quantis $q_{0.05}$, $q_{0.25}$, $q_{0.50}$, $q_{0.75}$ e $q_{0.95}$ do valor de função objetivo para estimar a dispersão do conjunto de soluções.

As Figs. 13.6a e 13.6b mostram o ordenamento das soluções oferecido pela *T-norma* e a distância de Hamming respectivamente, para a instância de 25 nós do OCST. Já as Figs. 13.7a e 13.7b apresentam estes resultados para o OCST de 50 nós.

Deve-se notar que a amplitude coberta pelo valor de função objetivo em cada sub-intervalo é muito maior para a distância de Hamming que para a *T-norma*. Isso significa que, no espaço métrico induzido pela T-norma, as redes estão espalhadas em torno do ótimo seguindo um padrão próximo ao de bacias de atração circulares: o valor de função objetivo é aproximadamente o mesmo para redes que estão localizadas à uma dada distância da rede ótima em todas as direções espaciais. Por outro lado, a distância de Hamming define conjuntos de redes que são "equidistantes" para a rede ótima, e apresentação valores de função objetivo consideravelmente diferentes. Isto pode ser interpretado,

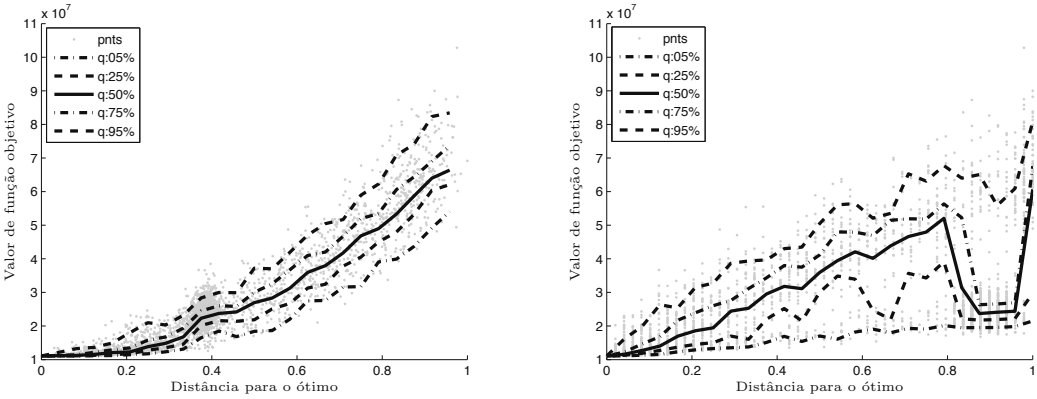
utilizando uma analogia do espaço contínuo, como "curvas de nível" da função objetivo que são consideravelmente diferentes de círculos, uma vez que um círculo cruza várias curvas de nível.

Estas observações suportam a interpretação de que a representação vetorial proposta gera coordenadas que são favoráveis para o processo de otimização.



(a) Distância *T-norma* (b) distância de Hamming

Figura 13.6: OCST (25 nós): Teste de panorama de aptidão.



(a) Distância *T-norma* (b) distância de Hamming

Figura 13.7: OCST (50 nós): Teste de panorama de aptidão.