**UNIVERSIDADE FEDERAL DE MINAS GERAIS**
**Programa de Pós-graduação em Engenharia Elétrica**

# Restauração de Sistemas de Distribuição de Energia Elétrica Utilizando Evolução Diferencial com Árvore de Ancestralidade

Ricardo Sérgio Prado

Texto submetido à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais para a obtenção do título de Doutor em Engenharia Elétrica

*Orientador:*
Frederico Gadelha Guimarães

Escola de Engenharia
Departamento de Engenharia Elétrica

Belo Horizonte, Dezembro de 2013

*"O ensinamento adequado é facilmente reconhecido. Podemos reconhecê-lo sem dúvida, quando ele nos desperta a sensação de termos ouvido algo que sempre soubemos"*

Frank Herbert, em Duna.

Aos meus pais, Rubens e Dinah, *in memoriam*.

# *Resumo*

Problemas de Restauração de Sistemas de Distribuição de Energia (SDE), como restauração de serviço, redução de perdas resistivas e planejamento da expansão do sistema, são normalmente formulados como problemas de otimização multiobjetivo com várias restrições inerentes ao sistema. Vários Algoritmos Evolucionários (AE) têm sido propostos na resolução desses tipos de problemas, mas a grande maioria deles ainda demandam um tempo computacional muito grande quando aplicados em grandes SDE (milhares de barramentos e chaves). Neste trabalho é apresentada uma nova abordagem para a restauração de serviço em SDE de larga escala utilizando um algoritmo discreto para Evolução Diferencial baseado em uma Lista de Movimentos com Árvore de Ancestralidade (DE-Tree), no qual a Árvore de Ancestralidade é utilizada para construir a lista de movimentos. A Representação Nó-Profundidade (RNP) é utilizada para representar computacionalmente a topologia do sistema elétrico e dois operadores, o *Preserve Ancestor Operator (PAO)* e o *Change Ancestor Operator(CAO)*, são utilizados na evolução da população. A abordagem proposta mostra que a Evolução Diferencial é adequada na resolução de problemas de otimização de SDE preservando o mecanismo de autoadaptação da mutação diferencial. Resultados apresentados na reconfiguração de SDE sugerem a adequação e a rápida convergência da abordagem proposta.

# *Abstract*

Problems in Power Distribution System Restoration (PDSR), such as service restoration, power loss reduction, and expansion planning, are usually formulated as multi-objective and multi-constrained optimization problems. Several Evolutionary Algorithms (EAs) have been developed to deal with PDSR problems, but the majority of EAs still demand high running time when applied to large-scale Distribution Systems (thousands of buses and switches). This work presents a new approach for service restoration in large-scale distribution systems that employs a Discrete Differential Evolution based on List of Movements with ancestor Tree (DE-Tree), in which the Ancestor Tree is used to obtain the list of movements. The Node-Depth Encoding (NDE) is used to computationally represent the electrical topology of the system and its operators, the Preserve Ancestor Operator (PAO) and the Change Ancestor Operator (CAO), are used to evolve the population. The proposed approach makes Differential Evolution suitable for treating combinatorial optimization problems related to PDSR preserving the self-adaptive differential mutation mechanism. Results presented on Distribution System Reconfiguration Problems indicates the adequacy and fast convergence of the proposed approach.

# *Agradecimentos*

Ao Prof. Frederico Gadelha Guimarães pela valorosa orientação prestada no desenvolvimento deste trabalho, pela amizade e pelos jantares gastronômicos.

Ao Prof. Oriane Magela Neto (*in memoriam*), colega de outrora, pela amizade e orientação inicial deste trabalho, sempre atencioso e que sempre nos contagiou, a todos, com seu sorriso (gargalhada) sempre alegre.

Aos professores e funcionários do PPGEE da UFMG que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Ao Instituto Federal de Minas Gerais - Campus Ouro Preto pelo apoio dispensado ao longo destes anos de doutoramento.

Aos colegas e amigos Sílvia, Rodrigo, Betânia, Lucas e Érica, Tatiana, Denise, Alan, André, pela amizade e os momentos agradáveis que desfrutamos juntos durante esses anos.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Abreviações

**AE** Algoritmos Evolucionários

**AG** Algoritmo Genético

**AMA** Abordagem por Matriz de Adjacências

**AMP** Abordagem por Matriz de Permutação

**CE** Computação Evolutiva

**DE-Tree** Evolução Diferencial com Árvore de Ancestralidade

**DE** Evolução Diferencial

**EE** Estratégias Evolucionárias

**FBT** Transformação Direta e Reversa(do inglês: *Forward/Backward Transformation*)

**IRP** Indexação por Posição Relativa (do inglês: *Indexing by Relative Position*)

**MPF** Ordenação de nós pelo Modelo Pai-Filho

**NA** Chaves seccionadoras normalmente abertas

**NF** Chaves seccionadoras normalmente fechadas

**PE** Programação Evolutiva

**PG** Programação Genética

**PSO** Otimização por Enxame de Partículas (do inglês: *Particle Swarm Optimization*)

**RNP** Representação Nó-Profundidade

**RS** Restauração de Serviço

**SDE** Sistema de Distribuição de Energia

**TSP** Problema do Caixeiro Viajante (do inglês: *Traveling Salesman Problem*)

# Símbolos

| | |
|---|---|
| $x$ | Variável real que representa cada parâmetro do sistema. |
| $\mathbf{X}$ | Vetor de variáveis reais representando uma solução. |
| $\mathcal{X}$ | Região de domínio, espaço dos parâmetros ou espaço de busca de otimização. |
| $f(.)$ | Função objetivo. |
| $b_{L,j}$ | Limite inferior de cada parâmetro do sistema. |
| $b_{U,j}$ | Limite superior de cada parâmetro do sistema. |
| $N_p$ | Número de indivíduos da população. |
| $V_{g,i}$ | Vetor mutante $i$ da geração $g$ |
| $X_{g,r_0}$ | Vetor base da equação da mutação diferencial da geração $g$ |
| $X_{g,r_1}$ | Vetor 1 da equação da mutação diferencial da geração $g$ |
| $X_{g,r_2}$ | Vetor 2 da equação da mutação diferencial da geração $g$ |
| $\mathbf{F}$ | Fator de multiplicação escalar da equação da mutação diferencial |
| $U_{g,i}$ | Vetor experimental da geração $g$ |
| $\mathcal{C}_r$ | Probabilidade de cruzamento |
| $\mathcal{N}$ | Estrutura de vizinhança |
| $\mathbf{P}$ | Matriz de Permutação |
| $\mathbf{P_F}$ | Matriz de Permutação ponderada pelo escalar $\mathbf{F}$ |
| $\mathbf{G}$ | Grafo representando uma estrutura em árvores. |

# Capítulo 1

# Introdução

## 1.1 Apresentação

Para lidar com certos problemas nas áreas da engenharia e da computação, que pelos métodos tradicionais são considerados intratáveis ou que demandam um tempo de execução computacional elevado, engenheiros e pesquisadores têm buscado na natureza modelos que sirvam de inspiração. Esta inspiração em processos naturais tem possibilitado criar modelos computacionais com larga aplicação na resolução de problemas práticos na indústria e comércio. Na área de otimização de sistemas, muitos processos observados na natureza têm sido utilizados como modelos na tentativa de solucionar tais problemas. Um olhar mais aguçado entre otimização de sistemas e evolução biológica levou ao desenvolvimento de um paradigma importante nas técnicas de inteligência computacional: a Computação Evolutiva (CE), que se utiliza dos princípios da evolução dos seres vivos, como seleção natural e herança genética, na resolução de problemas complexos de otimização. O paradigma da CE data da década de 1950, quando o uso dos princípios da teoria da evolução das espécies de Charles Darwin foram utilizados pela primeira vez na resolução de problemas de engenharia. Na década seguinte, vertentes distintas dessas técnicas foram desenvolvidas: a Programação Evolutiva (PE) [Fogel and Fogel 1996] desenvolvida por Lawrence J. Fogel nos Estados Unidos e, quase simultaneamente na Alemanha, as Estratégias Evolucionárias (EE) apresentadas por I. Rechenberg e H. P. Schwefel [Beyer and Schwefel 2002]. Uma década mais tarde, John Henry Holland e Ann Arbor apresentam o método Algoritmo Genético (AG) utilizado na solução de problemas práticos de otimização [Das and Suganthan 2011]. Estas três áreas seguem separadamente até que, na década de 1990, são unificadas como dialetos diferentes de uma mesma tecnologia denominada, então, por Computação Evolutiva. Também nesta mesma década, seguindo em linhas gerais o mesmo paradigma, surge a Programação

Genética (PG), uma nova vertente proposta por J.R. Koza [Koza 1990] e o algoritmo Evolução Diferencial (DE)[1] [Storn and Price 1995], tema em que este trabalho discorre.

Muitos algoritmos híbridos posteriormente desenvolvidos incorporam muitas das características e técnicas acima citadas e, consequentemente, são difíceis de serem classificados. Assim, todos esses métodos são referidos como Algoritmos Evolucionários (AEs) e esta divisão é meramente histórica. Hoje em dia, as meta-heurísticas inspiradas na natureza são constituídas pelos AEs (compreendendo os AGs, PE, EEs, PG, DE, etc.) bem como os algoritmos de Otimização por Enxame de Partículas (PSO)[2] [Kennedy and Eberhart 1995], os Sistemas Imunológicos Artificiais [de Castro and Timmis 2002], os algoritmos de Estimação de Distribuição [Larranaga and Lozano 2002] e muitos outros.

O DE surgiu na década de 1990 como um AE utilizado na otimização de sistemas a variáveis contínuas e é um algoritmo de otimização importante e poderoso em sistemas mono-objetivo [Price et al. 2005, Mezura-Montes and Coello 2006] e em sistemas multi-objetivo [Xue et al. 2003, Batista et al. 2009]. Atualmente, ele tem sido utilizado com algum sucesso em um grande número de problemas de natureza combinatória [Lichtblau 2002, Onwubolu and Davendra 2009, Price et al. 2005], em que o conjunto de soluções factíveis é um subconjunto dos números inteiros não-negativos e, portanto, pertencentes a uma classe de problemas do mundo real de natureza estritamente discreta.

## 1.2 Contexto Histórico

O algoritmo DE surgiu do algoritmo Recozimento Simulado Genético[3] desenvolvido por Kenneth Price. Este algoritmo era um algoritmo combinatório baseado em população que realizava um "recozimento" dirigido pelo desempenho médio dessa população. Logo após o desenvolvimento desse algoritmo, Price foi contactado por Rainer Storn que estava interessado em solucionar o problema de ajuste dos parâmetros do Polinômio de Tchebychev aplicando o algoritmo de recozimento simulado genético. Após alguns experimentos, Price modificou o algoritmo utilizando variáveis reais ao invés de codificação em *bit-string* e operadores aritméticos ao invés de operadores lógicos. Essas modificações transformaram o recozimento genético combinatório em um otimizador a variáveis contínuas. A descoberta do algoritmo DE aconteceu quando Price surgiu com a ideia de usar as diferenças entre vetores como uma maneira de perturbar e evoluir a população (método, esse, denominado então por *mutação diferencial*), um cruzamento por recombinação discreta e uma seleção em pares. Price e Storn detectaram que a mutação

[1] do inglês *Differential Evolution*
[2] do inglês *Particle Swarm Optimization*
[3] do inglês Genetic Simulated Annealing

diferencial, com recombinação discreta e seleção em pares, não necessitava de um fator de recozimento. Assim, o mecanismo de recozimento foi removido surgindo o algoritmo DE [Feoktistov 2006].

O primeiro artigo sobre o DE apareceu como um relatório técnico escrito por R. Storn e K. V. Price em 1995 [Storn and Price 1995]. Um ano depois ele participou do *First International Contest on Evolutionary Optimization (1st ICEO)*, realizado em conjunto com o *IEEE International Conference On Evolutionary Computation (CEC)* em Nagoya, Japão. O DE terminou o confronto em terceiro lugar e foi considerado o melhor algoritmo evolucionário na resolução de funções de testes contínuas (os dois primeiros lugares foram dados a algoritmos não evolutivos pois, apesar de não serem aplicáveis a vários tipos de problemas, resolviam as funções testes mais rápido que o DE). Em 1997, Price apresenta o DE no *Second International Contest on Evolutionary Optmization* [Price 1997] e acabou sendo um dos melhores entre os algoritmos concorrentes. Dois artigos consecutivos [Price and Storn 1997a] e [Price and Storn 1997b] descrevendo o algoritmo em detalhes são, então, publicados. Na competição de otimização de problemas contínuos clássicos de 10 dimensões no CEC de 2005, o DE assegurou a segunda colocação e uma variante auto-adaptativa do DE, denominada SaDE[4] [Qin and Suganthan 2005], assegurou o terceiro lugar, embora tenha tido um desempenho insuficiente em problemas acima de 30 dimensões. Outras variantes do DE continuaram a assegurar as melhores colocações nas subsequentes competições do CEC como as do CEC-2006 em otimização de parâmetros reais com restrições ($1^o$ lugar), CEC-2007 competição em otimização multiobjetivo ($2^o$ lugar), CEC-2008 competição em otimização global em larga escala ($3^o$ lugar), CEC-2009 competição em otimização multiobjetivo ($1^o$ lugar, dado a um algoritmo DE baseando no MOEA/D[5] para problemas sem restrições) e CEC-2009 competição em computação evolucionária em ambientes dinâmicos e incertos ($1^o$ lugar) [Das and Suganthan 2011].

## 1.3 Conceitos Básicos

Muitas vezes, em engenharia, nos deparamos com o problema de otimização que consiste em encontrar a melhor solução dentro de certas restrições e flexibilidades existentes em um determinado sistema. O objetivo da otimização é, então, encontrar um conjunto de valores dos parâmetros do sistema para o qual o seu desempenho é o melhor sobre estas restrições. Em um sistema não-linear a variáveis contínuas, a solução de um problema

[4] Self-adaptive Differential Evolution
[5] A Multiobjective Evolutionary Algorithm Based on Decomposition

de otimização irrestrito para um sistema mono-objetivo é definida como:

$$x^* = \arg \min_x f(x) \tag{1.1}$$

onde:

$x^*$ é a solução ótima. É o argumento que minimiza a função $f(x)$.

$f(.) : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ é a função de custo ou função objetivo a ser minimizada ou maximizada[6].

$\mathcal{X}$ é a região de domínio, espaço dos parâmetros ou espaço de busca de otimização. Ela é definida como:

$$\mathcal{X} = \{x \in \mathbb{R}^n : b_{L,j} \leq x_j \leq b_{U,j};\ j = 1, \ldots, n\}$$

em que $b_{L,j}$ e $b_{U,j}$ são, respectivamente, os limites inferior e superior de cada parâmetro $j$ do sistema.

É importante destacar que solução ótima pode ser local, se ela for a melhor solução dentro de uma determinada vizinhança, ou pode ser global, caso em que é a melhor solução encontrada para o sistema.

O algoritmo DE, assim como outros AEs, é baseado em populações e, portanto, utiliza um conjunto de vetores $X$ de soluções candidatas aleatoriamente geradas dentro da região de domínio $\mathcal{X}$ como população inicial. Para um sistema multivariável, a solução $x$, ou indivíduo da população, é representada por um vetor dado por:

$$\mathbf{X}_i = \left[x_{g,i,1}, x_{g,i,2}, x_{g,i,3}, \ldots, x_{g,i,n}\right]^T \tag{1.2}$$

onde o subscrito $g$ indica a geração em que cada indivíduo pertence, $i$ indica o indivíduo dentro da população e o terceiro subscrito indica um dos $j$ parâmetros da solução.

O objetivo da otimização passa, então, a ser o de encontrar o vetor de parâmetros $X^*$ o qual minimiza a função objetivo $f(.)$, isto é:

$$f(X^*) < f(X) \quad \forall \quad X \in \mathcal{X} \tag{1.3}$$

Uma vez criada a população inicial, o algoritmo DE realiza o processo de mutação para evoluir a população e gerar novas soluções ótimas. Basicamente, a evolução diferencial proposta por Price e Storn consiste em, através da equação da mutação diferencial,

[6] No restante deste texto, sem perda de generalidade, o problema de otimização será o de minimização da função objetivo, a não ser que seja dito o contrário.

criar indivíduos mutantes. Os indivíduos mutantes resultantes, por sua vez, passam por um processo de recombinação com os indivíduos da população corrente gerando uma população de *indivíduos experimentais*. Estes competem com os indivíduos da população corrente por um lugar na próxima geração [Price et al. 2005]. Cada uma destas etapas é apresentada nas Seções de 1.3.1 a 1.3.3.

### 1.3.1 Mutação Diferencial

Após gerada a população inicial, o algoritmo DE realiza a *mutação diferencial* como o principal processo de evolução da população. A mutação diferencial consiste em, para cada indivíduo corrente $X_{g,i}$ da população (também denominado *vetor alvo*), gerar um indivíduo mutante $V_{g,i}$. Para tanto, três indivíduos distintos entre si e entre o vetor alvo são escolhidos aleatoriamente dentro da população para fazerem parte da equação da mutação diferencial (Eq. 1.4): o primeiro denominado *vetor base* ($X_{g,ro}$) e os outros dois *vetores diferença* ($X_{g,r1}$ e $X_{g,r2}$, respectivamente). O vetor mutante é obtido da soma vetorial entre a diferença entre os vetores diferença, ponderada pelo escalar $\mathbf{F}$, e o vetor base.

$$V_{g,i} = X_{g,r_0} + \mathbf{F}.(X_{g,r_1} - X_{g,r_2}) \tag{1.4}$$

onde: $X_{g,r_0} \neq X_{g,r_1} \neq X_{g,r_2} \neq X_{g,i}$.

O fator escalar $\mathbf{F}$ é um número real positivo que controla a amplitude da diferença vetorial e, consequentemente, o tamanho do passo da mutação no espaço de busca e a taxa em que a população evolui. O intervalo indicado para o fator escalar é $\mathbf{F} \in (0, 1+)$. Embora não haja um limite superior para $\mathbf{F}$, raramente valores maiores que 1 são utilizados [Price et al. 2005]. Por outro lado, quando $\mathbf{F} = 1$, combinações distintas dos vetores diferença geram vetores mutantes idênticos, o que reduz o número de vetores mutantes pela metade [Price et al. 2005]. Normalmente, a escolha do melhor valor é feita por tentativa e erro numa bateria de testes que demandam um tempo considerável. Valores típicos encontrados na literatura estão entre $\mathbf{F} = 0,6$ e $\mathbf{F} = 0,9$. A Figura 1.1 ilustra como é obtido um vetor mutante $V_{g,i}$ em um sistema bidimensional.

A equação da mutação diferencial gera vetores mutantes que serão maiores ou menores dependendo da distribuição da população corrente no espaço de busca do problema. Essa diversidade de vetores afeta tanto o tamanho do passo de convergência como a sua orientação para as curvas de nível da função objetivo [Storn and Price 1995]. Esta propriedade do DE é mostrada nas Figuras 1.2 à 1.4[7] para as curvas de nível da função *peaks* (Equação 1.5) utilizada como exemplo.

[7] Figuras reproduzidas de: [Guimarães 2009]

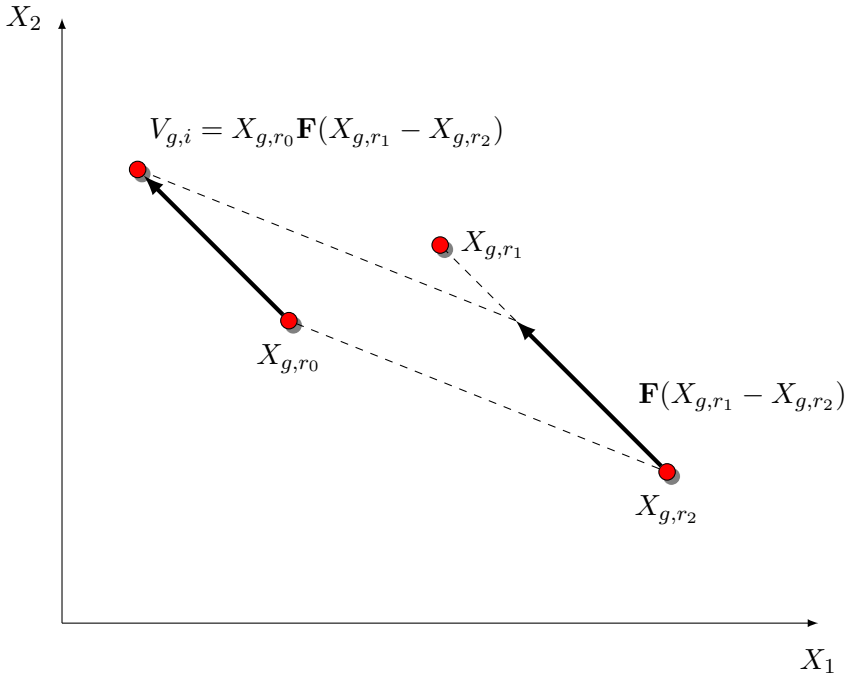



FIGURA 1.1: Mutação Diferencial: a diferença ponderada, $F(X_{g,r_1} - X_{g,r_2})$, é adicionada ao vetor base, $X_{g,r_0}$, para produzir o vetor mutante, $V_{g,i}$. (Figura reproduzida de: [Price et al. 2005] pág. 39).

$$f(x_1, x_2) = 3(1 - x_1)^2 . \exp\left(x_1^2 + (x_2 + 1)^2\right) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) . \exp\left(x_1^2 + x_2^2\right) - \frac{1}{3} . \exp\left((x_1 + 1)^2 + x_2^2\right) \tag{1.5}$$

A Figura 1.2(a) mostra a distribuição da população nas curvas de nível da função *peaks* para a geração 1 e, na Figura 1.2(b), são traçados, em um diagrama polar, todas as combinações possíveis que podem ser obtidas para os vetores diferença dessa população. Observa-se que, inicialmente, os tamanhos dos vetores diferença são grandes devido a distribuição, ainda aleatória, da população e estão uniformemente distribuídos em todas as direções.

À medida que a população evolui e converge para as bacias de atração, observa-se que todas as combinações possíveis dos vetores diferença tendem a ter tamanhos equivalentes às distâncias existentes entre essas bacias de atração e, também, direcionam-se para elas. Essa característica inerente ao DE é mostrada na Figura 1.3 para a geração 10.

A Figura 1.4(a) mostra que, para esse exemplo, após 20 gerações a população converge para a bacia de ótimo global e, (b), que os tamanhos dos vetores diferença limitam-se a esta bacia de atração e sua distribuição orienta-se para ela. A partir desse estágio,

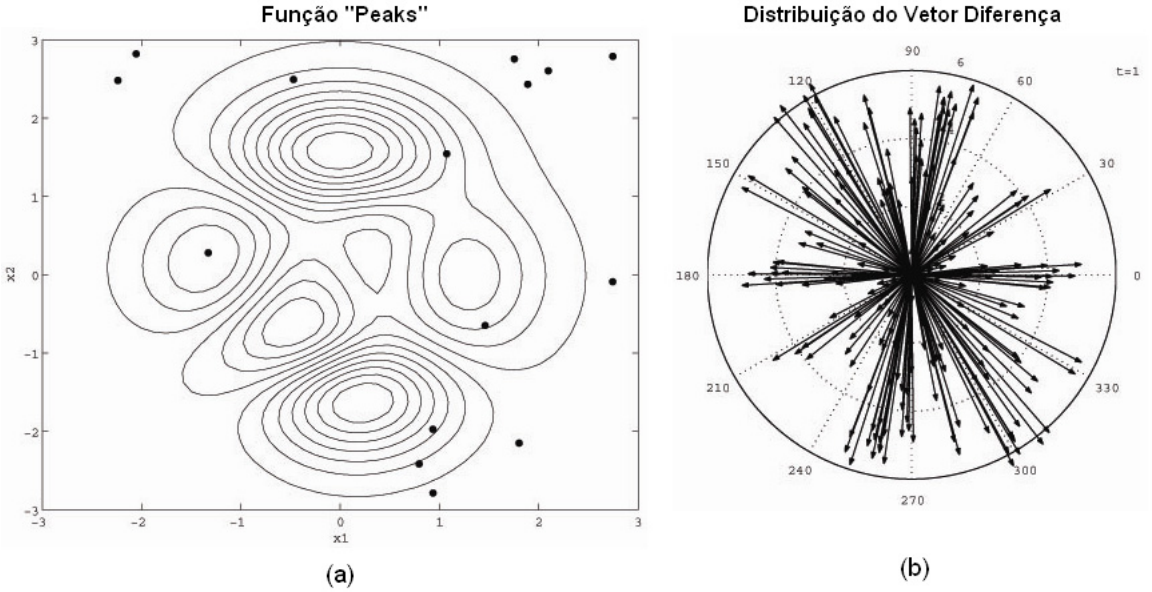



FIGURA 1.2: Geração 1: População ainda aleatória (a) e vetores diferença grandes e uniformemente distribuídos (b).

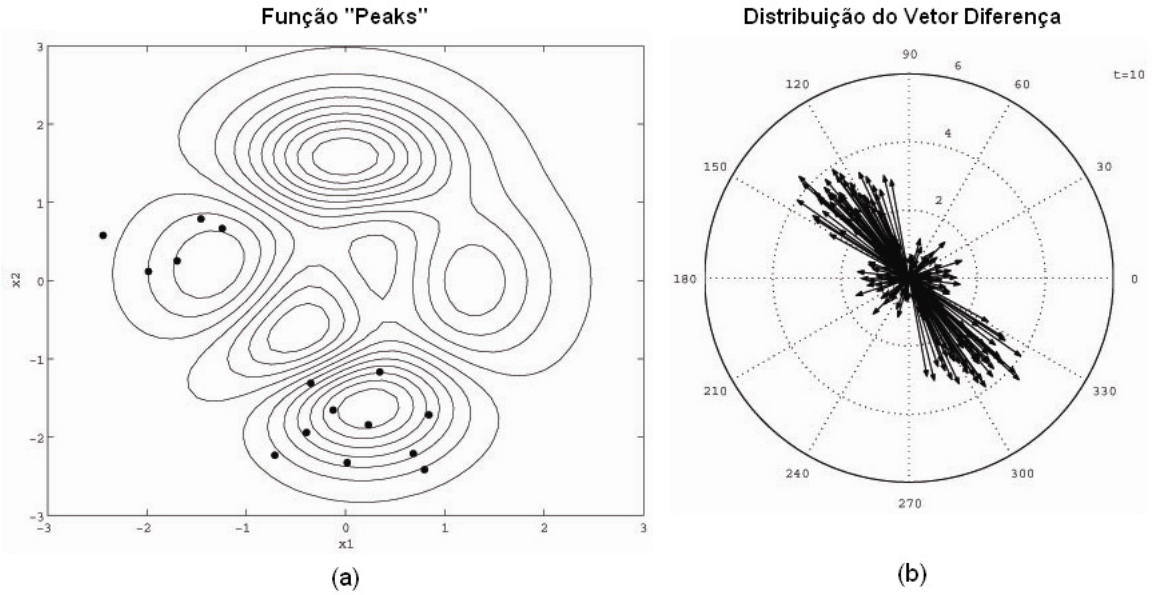



FIGURA 1.3: Geração 10: Distribuição da População em torno das bacias de atração (a) e vetores diferença direcionados para as bacias de atração (b).

com toda a população dentro da bacia de ótimo global e o tamanho dos vetores diferença estarem limitados a esta região, o DE inicia uma busca local pelo valor ótimo.

Essa característica do DE em obter a diferença entre dois indivíduos da população e usá-la para definir as direções de busca e ajustar o passo da mutação (inicialmente grande e, após a convergência pequeno) faz dele um algoritmo eficiente em otimização de sistemas a variáveis contínuas.

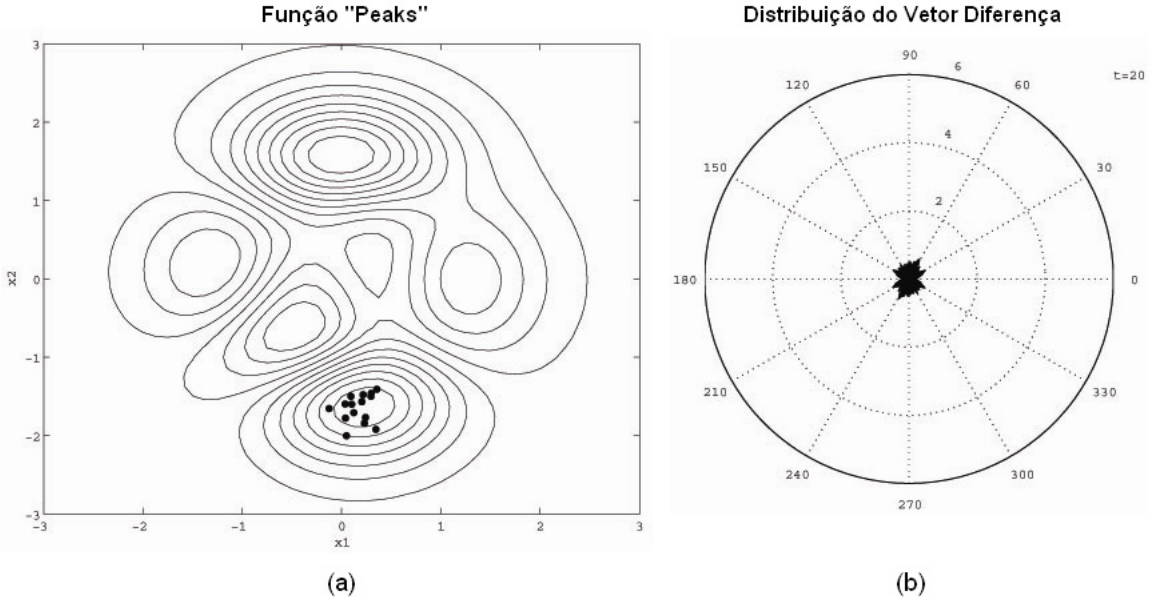



FIGURA 1.4: Geração 20: População atraída para o mínimo global (a) e tamanho dos vetores diferença e sua distribuição propícios a uma busca local (b)

### 1.3.2 Cruzamento

Após a etapa de mutação diferencial, o DE realiza a etapa de cruzamento para gerar os vetores experimentais. Nesta etapa, cada indivíduo alvo $X_{g,i}$ da população corrente e seu vetor mutante correspondente $V_{g,i}$ se recombinam para produzir um vetor experimental $U_{g,i}$ [Price et al. 2005]:

$$U_{g,i} = U_{g,i,j} = \begin{cases} \mathcal{C}ruzamento(V_{g,i,j}, X_{g,i,j}), & \text{se } rand(0,1) \leq \mathcal{C}_r \text{ ou } j = j_{rand} \\ X_{g,i,j}, & \text{c. c.} \end{cases} \tag{1.6}$$

onde:

$j$: $j^{ésimo}$ parâmetro do vetor correspondente,
$\mathcal{C}_r$: probabilidade de cruzamento,
$j_{rand}$: parâmetro do vetor mutante escolhido aleatoriamente.

Na sua versão clássica, o algoritmo DE utiliza uma probabilidade de cruzamento $\mathcal{C}_r \in [0,1]$. A probabilidade de cruzamento $\mathcal{C}_r$ controla a fração dos parâmetros do vetor mutante que serão copiados para a solução corrente. Um parâmetro do vetor mutante escolhido aleatoriamente, $j_{rand}$, é copiado para o vetor experimental para garantir que o vetor experimental não seja uma cópia da solução corrente.

Nos AEs, o cruzamento é realizado através da recombinação discreta entre dois vetores pais da mesma população para gerar dois vetores filhos. Diferentemente, o DE realiza a recombinação entre um vetor da população corrente (o vetor alvo) e o vetor mutante correspondente para gerar um único filho: o vetor experimental.

### 1.3.3 Seleção

O método de seleção utilizado no DE é semelhante ao método $(\mu + \lambda)$ utilizado nas EE, em que, os melhores $\mu$ vetores da combinação dos $\mu$ pais com os $\lambda$ filhos compõem a próxima geração. A diferença está no fato de que, ao invés de *ranquear* a população formada por pais e filhos, DE realiza uma competição entre cada vetor pai com o seu vetor filho correspondente selecionando um deles para a próxima geração (método de seleção $(1 + 1)$). Essa técnica de comparar cada vetor experimental (vetor filho) com o vetor corrente de mesmo índice (vetor pai) garante manter a melhor solução obtida para o indivíduo daquele índice. Este método de seleção garante a diversidade dos indivíduos e evita a convergência prematura da população, o que é importante no DE, já que a distribuição espacial da população influencia diretamente o desempenho do algoritmo [Storn and Price 1995].

Baseada na função objetivo do vetor alvo $X_{g,i}$ e do vetor experimental $U_{g,i}$, a seleção é realizada da seguinte maneira: se a função objetivo do vetor experimental for menor ou igual à do vetor alvo, o vetor experimental o substitui na próxima geração; caso contrário, o vetor alvo é preservado e vai para a próxima geração:

$$X_{g+1,i} = \begin{cases} U_{g,i}, & \text{se } f(U_{g,i}) \leq f(X_{g,i}) \\ X_{g,i}, & \text{c. c.} \end{cases} \tag{1.7}$$

Uma vez que a nova geração esteja completa, o processo de mutação, recombinação e seleção é repetido até que um determinado critério de parada estabelecido seja alcançado. O Algoritmo 1 apresenta o pseudocódigo do algoritmo DE na sua forma clássica. Esta versão clássica do algoritmo DE é também referida na literatura como **DE/ rand/1/bin**[8]. Modificações na etapa de mutação e na escolha do tipo de cruzamento podem gerar variações interessantes no algoritmo DE. Além da escolha aleatória para o vetor base, apresentada na sua versão clássica, outras duas versões são apresentadas em [Price et al. 2005]. Estas versões são a *best* e a *target-to-best*. Na primeira,

[8] Convencionalmente, o primeiro termo, **DE**, refere-se ao algoritmo, o segundo e o terceiro termos, **rand/1**, ao modo como a equação da mutação diferencial é implementada (vetor base escolhido aleatoriamente e número de vetores diferença que foram utilizados na equação da mutação diferencial: 1 vetor) e o quarto e último termo, **bin**, refere-se ao tipo de recombinação utilizado no cruzamento (recombinação binomial).

o vetor base é o indivíduo com a melhor função objetivo da população corrente. Na segunda o vetor base é uma recombinação direta entre o vetor alvo e o melhor indivíduo da população corrente conforme mostrado na Equação 1.8:

$$X_{g,r_0} = X_{g,i} + \lambda.(X_{g,best} - X_{g,i}). \tag{1.8}$$

**Algorithm 1** *Pseudocódigo DE Clássico. Listagem reproduzida de [Price et al. 2005] pág. 42*

**Input**: Np // Número de indivíduos da população
**Input**: **D** // dimensão do problema
**Input**: **F** // fator escalar da equação da mutação diferencial
// Geração da população de vetores experimentais
**repeat**
  **for** $(i = 0; i < Np; i++)$ **do**
    **repeat**
      $X_{r_0} = floor(rand(0,1) \times Np)$;
    **until** $(r_0 == i)$;
    **repeat**
      $X_{r_1} = floor(rand(0,1) \times Np)$;
    **until** $((r_1 == r_0)\, OR\, (r_1 == i))$;
    **repeat**
      $X_{r_1} = floor(rand(0,1) \times Np)$;
    **until** $((r_2 == r_1)\, OR\, (r_2 == r_0)\, OR\, (r_2 == i))$;
    $j_{rand} = floor(\mathbf{D} \times rand(0,1))$;
    // Cria vetores experimentais
    **for** $(j = 0; j < \boldsymbol{D}; j++)$ **do**
      **if** $(rand(0,1) \leq \mathcal{C}_r)\, OR\, (j == j_{rand})$ **then**
        $U_{j,i} = X_{j,r_0} + \mathbf{F} \times (X_{j,r_1} - X_{j,r_2})$
      **else**
        $U_{j,i} = X_{j,i}$;
  // Seleção da próxima geração
  **for** $(i = 0; i < Np; i++)$ **do**
    **if** $f(U_i) \leq f(X_i)$ **then**
      $X_i = U_i$;
**until** *(critério-de-parada não encontrado)*;

De maneira a não aumentar o número de parâmetros iniciais, o fator de multiplicação escalar $\lambda$ na Equação 1.8 é escolhido ser o mesmo fator de multiplicação escalar **F** da equação de mutação diferencial. Com estas modificações, duas novas estratégias são obtidas para o DE:

$$DE/best/1/bin \Rightarrow V_{g,i} = X_{g,best} + \mathbf{F}.(X_{g,r_1} - X_{g,r_2})$$

$$DE/target-to-best/1/bin \Rightarrow V_{g,i} = X_{g.i} + \mathbf{F}.(X_{g,best} - X_{g,i}) + \mathbf{F}.(X_{g,r_1} - X_{g,r_2}).$$

Outra modificação na equação da mutação diferencial sugerida por Price e Storn está no número de vetores diferença que participam da mutação. Os autores sugerem, além de 1 vetor diferença conforme apresentado na versão clássica, 2 ou até 3 vetores diferença na equação da mutação diferencial como mostrado abaixo:

$$DE/rand/2/bin \Rightarrow V_{g,i} = X_{g,r_0} + \mathbf{F}.(X_{g,r_1} - X_{g,r_2}) + \mathbf{F}.(X_{g,r_3} - X_{g,r_4})$$

$$DE/rand/3/bin \Rightarrow V_{g,i} = X_{g,r_0} + \mathbf{F}.(X_{g,r_1} - X_{g,r_2}) + \mathbf{F}.(X_{g,r_3} - X_{g,r_4}) + \mathbf{F}.(X_{g,r_5} - X_{g,r_6})$$

Outras versões para o DE podem ser obtidas combinando estas variações na equação da mutação diferencial com outros tipos de cruzamentos. Utilizando-se, por exemplo, uma recombinação exponencial apresentada em [Storn and Price 1995] com a versão clássica do DE, obtém-se a versão **DE/rand/1/exp**.

## 1.4 Objetivos

Em linhas gerais, o principal objetivo desta tese é desenvolver uma meta-heurística para o algoritmo DE no domínio das variáveis discretas. Esta meta-heurística deve preservar, de uma maneira análoga, o interessante mecanismo de busca observado na evolução diferencial no domínio das variáveis contínuas. Para tanto, os operadores aritméticos da equação da mutação diferencial no domínio das variáveis contínuas são redefinidos para o domínio das variáveis discretas, tornando a mutação diferencial mais significativa e genérica no contexto dos problemas de natureza combinatória. Os objetivos específicos podem ser divididos em três partes principais:

- Elaboração de uma revisão das principais abordagens apresentadas na literatura para o algoritmo DE aplicados em problemas combinatórios, enumerando suas vantagens e desvantagens.
- Implementação e realização de testes do algoritmo proposto em um problema real de Restauração de sistema de distribuição de energia elétrica.

## 1.5 Motivação

No âmbito dos problemas de otimização combinatória, devido à sua importância tanto no mundo científico bem como no mundo industrial, novas adaptações têm sido propostas na literatura para o algoritmo DE com o intuito de observar aquelas características obtidas no domínio das variáveis contínuas. Recentemente, várias adaptações do algoritmo DE aplicado a problemas de natureza combinatória têm sido apresentadas na literatura. Diversas dessas abordagens consistem simplesmente em aplicar a equação da mutação diferencial definida no domínio das variáveis contínuas diretamente em problemas de natureza combinatória. Todavia, em problemas de natureza combinatória, a diferença vetorial no espaço das variáveis contínuas não tem uma correspondência clara no espaço das variáveis discretas. Isto deve-se ao fato de que, diferenças de números inteiros que são meramente rótulos e não representam nenhuma quantidade numérica, não geram soluções viáveis e nem representam direções significativas no espaço das variáveis discretas. Exemplos dessas abordagens são a Indexação Por Posição Relativa (IRP)[9] [Price et al. 2005] e a Transformação Direta e Reversa (FBT)[10] [Onwubolu and Davendra 2009] aplicadas em alguns problemas combinatórios com permutação. Outras abordagens empregam esquemas mais sofisticados redefinindo ou conceituando o que seria a diferença vetorial no domínio das variáveis discretas. Exemplos dessas abordagens são a Abordagem Por Matriz de Permutação e a Abordagem Por Matriz de Adjacências apresentadas em [Price et al. 2005]. Tais abordagens são em geral aplicáveis especificamente a uma classe de problemas combinatórios.

Isto posto, uma abordagem para a evolução diferencial onde os operadores aritméticos da mutação diferencial são definidos especificamente para problemas combinatórios mostra-se interessante. Estes operadores especiais no contexto das variáveis discretas, mas análogos àqueles aplicados às variáveis contínuas, devem ser definidos de modo a preservar as características que tornaram o DE um otimizador importante no domínio das variáveis contínuas. Isso permitiria sua aplicação direta em problemas combinatórios sem a necessidade de rotinas especiais de reparação das soluções não triviais ou fora do espaço de busca do problema.

[9] do inglês *Indexing by Relative Position*
[10] do inglês *Forward/Backward Transformation*

## 1.6 Estrutura do Trabalho

Os tópicos apresentados neste trabalho estão organizados conforme mostrado a seguir:

**Capítulo 2 - DE Para Otimização Combinatória:** Discorre-se acerca dos principais trabalhos relacionados à evolução diferencial no domínio das variáveis discretas. Nele, apresenta-se um estudo crítico detalhado considerando as abordagens mais citadas na literatura, a saber, Indexação por Posição Relativa, Transformação Direta e Reversa, Matriz de Permutação e Matriz de Adjacências. Ao final do capítulo a abordagem por Lista de Movimentos é proposta e as possíveis definições para esta lista são apresentadas para algumas estruturas de dados comumente utilizadas na representação de soluções de problemas discretos.

**Capítulo 3 - Restauração de Sistemas de Distribuição de Energia Elétrica:** Apresenta-se o problema de Restauração de sistemas de distribuição de energia elétrica com contingências devido a faltas no setor primário da rede. Com o objetivo de reconfigurar o sistema elétrico, propõe-se uma versão do algoritmo DE com lista de movimentos, o algoritmo DE-Tree. Para esta versão do algoritmo, desenvolveu-se uma estrutura de dados, denominada de Árvore de Ancestralidade, que possibilita armazenar a sequência das manobras de chaves necessária para reconfigurar o sistema, além de guardar as informações acerca dos movimentos que serão utilizados na equação da mutação diferencial.

**Capítulo 4 - Resultados Experimentais:** Descrevem-se os resultados das simulações realizadas em um sistema de distribuição de energia elétrica real de grande porte, com o intuito de averiguar o desempenho do algoritmo proposto. Testes são realizados para o caso de contingências devido a uma única falta e três faltas simultâneas no sistema elétrico.

**Capítulo 5 - Conclusões:** Encerra-se este texto apresentando uma discussão geral sobre o trabalho e enumerando sugestões para trabalhos futuros.

# Capítulo 2

# DE Para Otimização Combinatória

Do ponto de vista prático, um problema de otimização consiste em encontrar a melhor configuração de um dado sistema de modo que ele opere de forma rápida e eficiente em um tempo reduzido. Teoricamente, consiste em determinar os valores extremos de uma função, isto é, o máximo ou mínimo valor que uma função pode assumir em um dado intervalo. Neste cenário, a otimização pode ser dividida em duas categorias: aquela onde as soluções são codificadas como variáveis reais e aquela onde as soluções são codificadas como variáveis discretas. Nesta última, encontra-se a classe de problemas designados como Problemas de Otimização Combinatória. Problemas de otimização combinatória ocorrem em áreas tão diversas como as de projetos de sistemas de distribuição de energia, roteamento de veículos, alocação de pessoas ou máquinas a tarefas, empacotamento de caixas em contêineres, sequenciamento de genes e DNA, dentre outras. Tais problemas e suas variantes podem ser enquadrados e tratados em uma das seguintes classes de problemas teóricos de natureza combinatória: Problema do Caixeiro Viajante, Problema de Roteamento de Veículos, Problema de Empacotamento, Problema de Programação de Tarefas, Problema da Mochila, etc. Para tanto, para cada tipo de problema considerado, o espaço de busca deve ser bem definido. Na classe de problemas combinatórios, os espaços de busca são representados por um conjunto finito de soluções cujos parâmetros podem ser do tipo arranjo, grupamento, ordenações, permutações ou mesmo por uma estrutura em grafo e, portanto, necessitam ser tratados de uma maneira especial, diferente daquelas utilizadas em problemas estritamente de natureza contínua.

Muitos algoritmos têm sido propostos na área da otimização para solucionar problemas de natureza estritamente combinatória. Estes algoritmos são classificados como *Exatos* ou *Aproximados*. Os algoritmos exatos garantem encontrar, para um número

finito de instâncias, uma solução ótima. Entretanto, o tempo computacional necessário gasto para encontrar tal solução ótima pode ser exponencial no pior caso, o que é impraticável em problemas reais. Assim, os algoritmos aproximados, apesar de não garantirem encontrar a solução ótima mas sim uma boa solução em um tempo computacional menor, têm sido objeto de pesquisa nos últimos anos.

Dos métodos aproximados podem-se destacar os *Algoritmos Construtivos* e os *Algoritmos de Busca Local*. Os algoritmos construtivos consistem em inserir componentes em uma solução, inicialmente vazia, de forma incremental até que uma solução completa seja obtida. Eles são tipicamente os métodos aproximados mais rápidos apesar de muitas vezes retornarem soluções de qualidade inferior quando comparados aos algoritmos de busca local. Já os algoritmos de busca local partem de uma solução inicial e iterativamente substituem a solução atual por uma solução melhor em uma dada *Estrutura de Vizinhança* apropriadamente predefinida. Esta estrutura de vizinhança pode ser formalmente definida como [Blum and Roli 2003]:

> *Definição* 2.1. Seja $\mathcal{X}$ o espaço de busca de um problema e $x$ uma solução deste problema. Uma estrutura de vizinhança é uma função $\mathcal{N}(x) \subseteq \mathcal{X}$ que associa à cada solução $x \in \mathcal{X}$ uma solução $x' \in \mathcal{N}(x)$. $\mathcal{N}(x)$ denota o conjunto de soluções $x'$ que pode ser obtido de $x$.

A transição de uma solução $x$ para uma solução vizinha $x'$ é designada como um *movimento*. Os movimentos possíveis em uma estrutura de vizinhança estão relacionados com a estrutura de dados utilizada na representação do problema. Para uma solução representada por um vetor os movimentos possíveis seriam, por exemplo, uma troca de seus elementos. Esta troca de elementos do vetor podeira gerar um vetor vizinho com características melhores que a do original e, assim, o substituiria no conjunto de soluções. Em uma estrutura em grafos, um movimento para gerar uma estrutura de vizinhança seria a adição ou remoção de arestas e, assim, para cada tipo de problema uma estrutura de vizinhança tem que ser definida.

A partir da definição de estrutura de vizinhança é possível definir o conceito de soluções que localmente são mínimas para uma dada estrutura de vizinhança:

> *Definição* 2.2. Uma solução $\widehat{x}$ é localmente mínima (ou mínimo local) em relação a uma vizinhança $\mathcal{N}(x)$ se: $\forall\, x \in \mathcal{N}(\widehat{x}) : f(\widehat{x}) \leq f(x)$.

Dentro da classe dos algoritmos aproximados estão as *meta-heurísticas*, uma classe de algoritmos que combina as heurísticas básicas com procedimentos avançados visando melhorar a eficiência e eficácia na exploração do espaço de busca do problema. Dentro das meta-heurísticas estão incluídos, dentre outros, os algoritmos *Colônia de Formigas*, *Busca Local Iterativa*, *Recozimento Simulado*, *Busca Tabu* e os *Algoritmos Evolutivos*.

A meta-heurística DE, uma classe dos AE, devido ao seu grande sucesso na otimização de problemas no espaço das variáveis contínuas tem recebido adaptações no seu mecanismo de mutação diferencial para atender a otimização de problemas estritamente de natureza combinatória. Este capítulo apresenta e discute algumas destas abordagens propostas na literatura especializada e, ao final, apresenta uma nova abordagem denominada por *Lista de Movimentos*, que visa preservar o interessante mecanismo de busca do DE no domínio das variáveis discretas.

## 2.1 Indexação por Posição Relativa

Em [Lichtblau 2002] é apresentada uma versão discreta para a evolução diferencial aplicada em alguns problemas de otimização combinatória baseados em permutação usando a abordagem denominada Indexação Por Posição Relativa (IRP[1]). Nesta abordagem, os indivíduos da população são representados por vetores cujos parâmetros são *arrays* de números inteiros. O princípio básico é transformar cada parâmetro dos vetores no domínio dos inteiros para o domínio dos reais no intervalo $[0, 1]$. Para tanto, os vetores são normalizados dividindo-se cada parâmetro pelo maior dentre eles. Após esta normalização (conversão dos parâmetros inteiros em números reais), a equação da mutação diferencial (Eq. 1.4) é aplicada diretamente como no caso do domínio contínuo. A conversão de volta para o domínio dos inteiros é obtida usando-se uma indexação baseada na posição relativa de cada elemento dentro do vetor. Os elementos no vetor são ordenados de acordo com seus valores. Assim, o menor número real obtido é relacionado ao menor valor inteiro, o próximo menor valor real ao próximo menor valor inteiro e assim sucessivamente, até todos os elementos no vetor serem convertidos para o domínio dos números inteiros novamente. Um exemplo ilustrativo desta abordagem para o Problema do Caixeiro Viajante (TSP[2]), formulado no Apêndice A, é apresentado no Apêndice A.1.

Essa abordagem sempre resultará em soluções válidas, desde que os valores convertidos para o domínio dos números reais não sejam idênticos. Se isso ocorrer, o vetor mutante resultante terá valores inteiros idênticos e, portanto, ele deve ser reparado ou

[1] do inglês *Indexing by Relative Position*
[2] do inglês *Traveling Salesman Problem*

descartado. No exemplo apresentado no Apêndice A.1 é fácil observar que esta abordagem tão somente embaralha os parâmetros do vetor e uma mutação idêntica àquela obtida na mutação diferencial não é obtida. Além disso, esta abordagem só é aplicável tão somente em problemas combinatórios com permutação e, mesmo assim, ela não é capaz de identificar a geração de permutações idênticas [Price et al. 2005].

## 2.2 Transformação Direta e Reversa

A Transformação Direta e Reversa (FBT[3]) apresentada em [Onwubolu and Davendra 2009], também referida como Abordagem de Onwubolu, utiliza o mesmo princípio de Lichtblau no que diz respeito a transformar os parâmetros de números inteiros dos vetores em números reais, aplicar a equação de mutação diferencial e retorná-los para o domínio dos inteiros. A diferença está na forma como isso é feito. Nesta abordagem, uma "transformação direta" é utilizada para transformar os elementos inteiros do vetor para o domínio das variáveis contínuas. Isso é feito utilizando a expressão:

$$x_i^{pf} \quad = \quad -1 \; + \; x_i(1+\alpha) \tag{2.1}$$

onde:

> $x_i$ é o $i^{ésimo}$ parâmetro do vetor 9sobrescrito $pf$ denota a representação dos elementos do vetor em ponto-flutuante).
> $\alpha$ é um número POSITIVO muito pequeno.

Uma vez que os elementos dos vetores estão no domínio dos números reais, a equação da mutação diferencial é aplicada. Em seguida, a operação inversa ("transformação reversa") que converte os valores reais de volta para o domínio dos inteiros (Eq. 2.2) é aplicada em cada elemento do vetor mutante para obter sua representação discreta.

$$x_i \quad = \quad round\Big[(1 \; + \; x_i^{pf})(2-\alpha)\Big], \tag{2.2}$$

onde a função *round* arredonda seu argumento para o inteiro mais próximo.

Essa abordagem geralmente gera soluções inválidas, ora devido à geração de soluções fora do espaço de busca, ora devido à geração de soluções com valores repetidos e, portanto, necessitam de um mecanismo de reparo adequado. [Price et al. 2005] relatam que, apesar dos bons resultados de trabalhos apresentados por Onwubolu usando esta abordagem, há motivos para acreditar que os sucessos obtidos são consequência direta

[3]do inglês *Forward-Backward Transformation*

dos mecanismos de reparos adotados e da prudência na escolha das heurísticas utilizadas. O Apêndice A.2 apresenta um exemplo ilustrativo da aplicação dessa abordagem no TSP.

## 2.3 Abordagem por Matriz de Permutação - AMP

Essa abordagem, apresentada por Price e Storn em [Price et al. 2005], faz uma analogia entre a diferença vetorial no domínio dos números reais e a matriz de permutação na análise combinatória. Assim no domínio dos números reais a diferença entre dois vetores resulta em um vetor, no campo da análise combinatória, duas permutações poderiam definir um mapeamento que também é uma permutação. Este mapeamento é representado por uma matriz de Permutação. Seguindo essa analogia, a mutação diferencial equivaleria, nos problemas combinatórios com permutação, à matriz de permutação (ou *Permutação Diferencial*).

Sejam $\mathbf{P}$ uma matriz de permutação que mapeia a permutação dos elementos de um dado vetor em outro e, $x_{g,r1}$ e $x_{g,r2}$, os dois vetores da equação da mutação diferencial aleatoriamente escolhidos da população. A matriz de permutação $\mathbf{P}$ que mapeia $x_{g,r_2}$ em $x_{g,r_1}$ satisfaz a relação:

$$x_{g,r1} = \mathbf{P}\, x_{g,r_2} \tag{2.3}$$

Pode-se dizer, portanto, que a matriz de permutação $\mathbf{P}$ "leva" $x_{g,r_2}$ à $x_{g,r_1}$. No contexto da evolução diferencial, esta matriz é considerada a "diferença" obtida de duas soluções candidatas. A equação análoga à equação da mutação diferencial (Eq 1.4) é, então, escrita como:

$$v_{g,i} = \mathbf{P_F}.x_{g,r_0} \tag{2.4}$$

onde $\mathbf{P_F}$ é a matriz de permutação modificada pelo parâmetro escalar $\mathbf{F}$, significando aqui, uma probabilidade de se utilizar uma parcela da permutação representada pela matriz de permutação original $\mathbf{P}$. Portanto, esta abordagem aplica algumas permutações ao vetor base aleatoriamente selecionadas do conjunto de permutações da matriz $\mathbf{P}$.

Segundo [Price et al. 2005], esta abordagem tende a estagnar a exploração do espaço de busca,uma vez que os movimentos derivados de permutações são raramente produtivos. Além disso, esse método não é capaz de identificar se duas soluções são equivalentes. Essa abordagem sempre gera soluções válidas pois todas a operações são derivadas de permutações, porém, está restrita a ser aplicada a esta classe de problemas. Um exemplo ilustrativo desta abordagem é apresentado no Apêndice A.3 para o TSP.

## 2.4 Abordagem por Matriz de Adjacências - AMA

Storn [Price et al. 2005, Onwubolu and Davendra 2009], propôs a Abordagem por Matriz de Adjacências (AMA), em que os vetores soluções da equação da mutação diferencial são codificados em matrizes de adjacências. Como na AMP, esta abordagem também estabelece uma analogia do que seria a diferença vetorial na análise combinatória. Aqui, entretanto, a diferença vetorial é a "diferença" entre a representação dos dois vetores $x_{g,r_1}$ *e* $x_{g,r_2}$ pelas suas respectivas matrizes de adjacências.

Sejam $x_{g,r_0}$, $x_{g,r_1}$ *e* $x_{g,r_2}$ as matrizes de adjacências dos vetores da equação da mutação diferencial. Assim, usando aritmética Módulo 2[4] e uma vez que as operações de adição e subtração nesta aritmética são idênticas, a equação da mutação diferencial é dada por:

$$v_{g,i} = x_{g,r_0} \oplus \mathbf{F}.(x_{g,r_1} \oplus x_{g,r_2}) \tag{2.5}$$

onde: o operador $\oplus$ indica a operação lógica XOR.

Essa abordagem é restrita a problemas combinatórios com permutação. Porém, rotinas especiais de reparo devem ser aplicadas às soluções obtidas pois há casos em que a diferença entre duas soluções (duas matrizes de adjacências) nem sempre gera uma solução que é uma matriz de adjacências. Além disso, soluções idênticas mas com seus parâmetros simplesmente "rotacionados" dentro do vetor geram matrizes de adjacências idênticas. Consequentemente, a matriz diferença entre estas soluções será sempre zero [Price et al. 2005]. Na ocorrência desses casos, a solução obtida deverá ser reparada ou descartada. No Apêndice A.4 um exemplo, também para o TSP, é apresentado.

## 2.5 Lista de Movimentos: Abordagem Proposta

Ao longo das seções deste capítulo apresentaram-se as principais abordagens encontradas na literatura para a utilização do DE no domínio das variáveis discretas. Essas abordagens podem ser classificadas em dois principais grupos: um grupo onde a equação da mutação diferencial no domínio das variáveis contínuas é aplicada diretamente em problemas combinatórios, caso das abordagens IRP e FBT, e o outro em que a equação da mutação diferencial é redefinida para a classe de problemas combinatórios, como ocorre para a AMP e AMA. No primeiro grupo há necessidade de reparos nas soluções geradas, ora devido a soluções repetidas, ora devido a soluções fora do espaço de busca do problema. Isto se deve ao fato de que os operadores aritméticos da equação da

[4]Como os elementos de uma matriz de adjacências são zeros e uns, todas as operações são realizadas usando aritmética módulo 2 (MOD2 ou operação lógica XOR).

mutação diferencial, operadores estes definidos para o domínio das variáveis contínuas, não são definidos no espaço das variáveis discretas. Já no segundo grupo as abordagens não são abrangentes e estão restritas a problemas combinatórios com permutação e, mesmo assim, em alguns casos necessitam de algum mecanismo de reparo para as soluções geradas.

Embora as técnicas apresentadas sejam importantes no contexto da otimização discreta, as limitações apresentadas comprometem o seu uso em um grande número de problemas de natureza combinatória.

Com o objetivo de contornar tais problemas, esta seção apresenta uma nova abordagem para a evolução diferencial, denominada *Lista de Movimentos*, que busca ser genérica no âmbito dos problemas de natureza combinatória. A flexibilidade existente nesta abordagem deve-se ao fato de que os movimentos definidos na lista de movimentos são específicos para cada tipo de problema combinatório. Assim, os movimentos da lista ora são representados por vetores de *arrays*, ora por matrizes ou por uma outra estrutura de dados mais apropriada a cada tipo de problema tratado. Além disso, nenhum mecanismo de reparo ou rotina auxiliar se faz necessária para reparar soluções encontradas e a autoadaptação da DE, observada no domínio das variáveis contínuas, é preservada no espaço de variáveis discretas.

De modo a conceber uma meta-heurística para o método de otimização do algoritmo DE aplicado a problemas de natureza combinatória, a diferença entre dois vetores no espaço das variáveis contínuas deve ser redefinida para o espaço das variáveis discretas, uma vez que o operador diferença não é definido neste domínio. Assim, a diferença vetorial entre as duas soluções candidatas da equação da mutação diferencial é representada por uma *Lista de Movimentos* e definida como:

> *Definição* 2.3. **Operador Subtração:** A operação de subtração entre duas soluções candidatas origina uma lista de movimentos. A lista de movimentos resultante, $M_{ij}$, é a lista contendo uma sequência de movimentos válidos $m_k$, tal que, a aplicação destes movimentos à solução $x_j \in \mathcal{X}$ leva à solução $x_i \in \mathcal{X}$.

Sejam $x_{g,r_1}$ e $x_{g,r_2}$ duas soluções candidatas de um problema combinatório qualquer, escolhidas aleatoriamente da população corrente, para realizar a "diferença vetorial" da equação da mutação diferencial. Com base na Definição 2.3, a lista de movimentos obtida é dada por:

$$M_{r_1 r_2} = x_{g,r_1} \ominus x_{g,r_2} \tag{2.6}$$

onde $\ominus$ é um operador de subtração binário especial que retorna a lista de movimentos $M_{r_1 r_2}$ que representa o "caminho" ou a "distância" entre a solução $x_{g,r_2}$ e a solução $x_{g,r_1}$. Esta lista, de algum modo, captura a diferença entre estas duas soluções.

A multiplicação da lista de movimentos por um escalar deve ser também definida:

*Definição* 2.4. **Operador Multiplicação por Escalar:** A multiplicação da lista de movimentos, $M_{ij}$, por um escalar $\lambda \in [0, 1]$, retorna a lista $M_{ij}^{'}$ com os primeiros $\lceil \lambda \times |M_{ij}| \rceil$ movimentos de $M_{ij}$ , onde $|M_{ij}|$ é o tamanho da lista.

Como o escalar $\lambda \in [0, 1]$, esta multiplicação prioriza, ou considera, tão somente os $100\lambda\%$ primeiros movimentos da lista.

Aplicando a Definição 2.4 à Equação 2.6, com o escalar $\lambda$ substituído pelo fator de multiplicação escalar **F** da evolução diferencial, a multiplicação por escalar no domínio discreto pode ser escrita como:

$$M'_{r_1 r_2} = F \otimes M_{r_1 r_2} \tag{2.7}$$

onde $\otimes$ é um operador de multiplicação binário especial no domínio das variáveis discretas.

Finalmente, a aplicação de uma lista de movimentos a uma dada solução é definida pela operação de adição:

*Definição* 2.5. **Operador Adição:** A operação de adição é a aplicação da sequência de movimentos da lista $M'_{ij}$ à solução $x_k$ e a soma resultante é a nova solução $x'_k$.

Seja o vetor solução $x_{g,r_0}$, escolhido aleatoriamente na população corrente, o vetor base da equação da mutação diferencial. O vetor mutante, resultante da adição do vetor base à lista de movimentos dada pela Equação 2.7, é dado por:

$$v_{g,i} \; = \; x_{g,r_0} \oplus M'_{r_1 r_2} \tag{2.8}$$

onde $\oplus$ é um operador binário especial de adição no domínio das variáveis discretas.

Com as definições acima, pode-se escrever a equação da mutação diferencial que determina o vetor mutante como:

$$\begin{aligned} v_{g,i} &= x_{g,r0} \oplus F \otimes (x_{g,r_1} \ominus x_{g,r_2}) \\ v_{g,i} &= x_{g,r0} \oplus F \otimes M_{r_1 r_2} \\ v_{g,i} &= x_{g,r0} \oplus M'_{r_1 r_2} \end{aligned} \tag{2.9}$$

que é a abordagem discreta proposta para a equação da mutação diferencial (Eq. 1.4).

A definição para a lista de movimentos é genérica e a estrutura de dados utilizada para representá-la depende da estrutura de dados utilizada para representar as soluções do problema. Desta forma, as definições acima podem ser aplicadas a qualquer tipo de problema combinatório. Para tanto, a estrutura de dados da Lista de Movimentos deve ser bem definida e específica para o tipo de problema combinatório em questão.

A abordagem por Lista de Movimentos proposta para evolução diferencial discreta tem como pilar a Definição 2.3 que define como uma lista de movimentos é construída. Segundo esta definição, a lista de movimentos deve conter uma sequência de movimentos válidos, tal que, a aplicação destes movimentos a uma solução $x_j$ leva a uma outra solução $x_i$ pertencente ao espaço de soluções $\mathcal{X}$. Portanto, para cada tipo de problema combinatório, baseado na definição da estrutura de dados utilizada na representação das soluções do problema, é definida a estrutura de dados que melhor represente a lista de movimentos. As seções 2.6 a 2.8 apresentam algumas estruturas de dados utilizadas na representação das soluções de alguns problemas de otimização combinatória e as representações possíveis para as listas de movimentos.

## 2.6 Lista de Movimentos *Swap*

Problemas combinatórios com permutação podem ter suas soluções representadas por uma estrutura de dados do tipo *array*. Quando representados desta forma, uma possível representação da estrutura de dados da lista de movimentos seria uma lista de pares de índices *(i,j)* dos elementos do *array* indicando quais elementos da solução sofreram um movimento de troca de posição (*movimento de swap*). Sejam, por exemplo, duas soluções candidatas de um problema combinatório com permutação representadas pelos seguintes *arrays*:

$$X_{g,r_1} = \left[ \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \right]$$
$$X_{g,r_2} = \left[ \begin{array}{cccccccc} E & D & A & F & H & B & C & G \end{array} \right] \tag{2.10}$$

A lista de movimentos $M_{ij} = M_{X_{r_2},X_{r_1}}$ contendo os movimentos necessários para levar a solução $X_{g,r_2}$ à solução $X_{g,r_1}$ é escrita anotando-se os pares de índices dos elementos da solução $X_{g,r_2}$ que devem ser trocados de modo a se obter a solução $X_{g,r_1}$. As Figuras 2.1 à 2.6 ilustram tal procedimento.

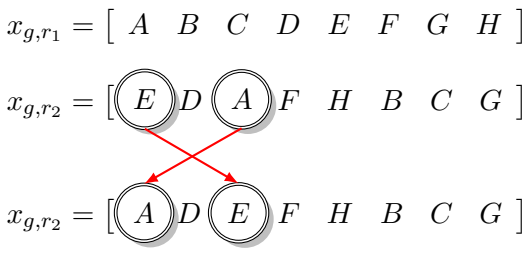



FIGURA 2.1: **Passo1:** Troca dos elementos de índice 0 e 2 para obter o primeiro movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)]$.

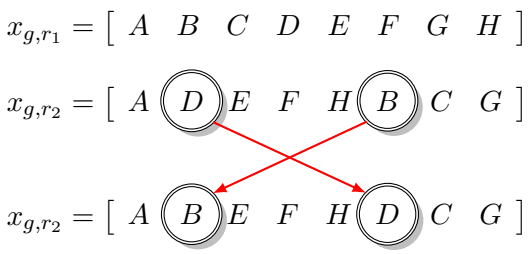



FIGURA 2.2: **Passo2:** Troca dos elementos de índice 1 e 5 para obter o segundo movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)]$.

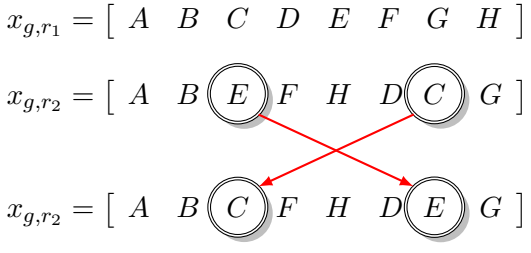



FIGURA 2.3: **Passo3:** Troca dos elementos de índice 2 e 6 para obter o terceiro movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)]$.

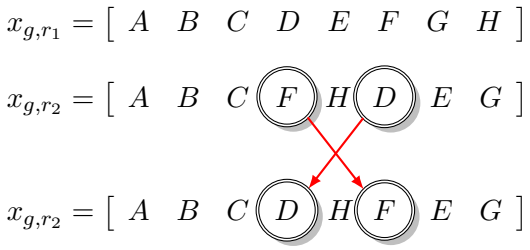



FIGURA 2.4: **Passo4:** Troca dos elementos de índice 3 e 5 para obter o quarto movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(3,5)]$.

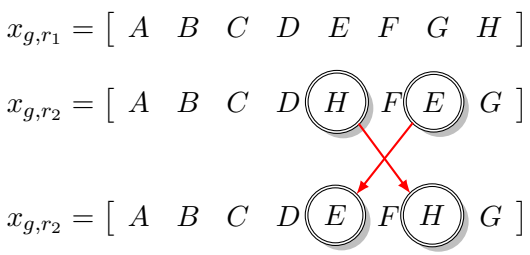



FIGURA 2.5: **Passo5:** Troca dos elementos de índice 4 e 6 para obter o quinto movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(3,5)\,(4,6)]$.

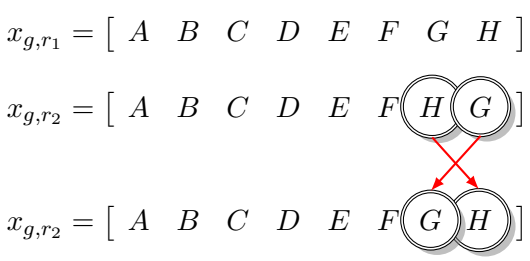



FIGURA 2.6: **Passo6:** Troca dos elementos de índice 6 e 7 para obter o último movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(3,5)\,(4,6)\,(6,7)]$.

A Lista de Movimentos final obtida com estes movimentos é dada por:

$$M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(3,5)\,(4,6)\,(6,7)]\,. \tag{2.11}$$

Os movimentos de $M_{X_{r_2},X_{r_1}}$ aplicados à solução $X_{g,r_2}$ geram a solução $X_{g,r_1}$, assim, essa lista pode ser interpretada como sendo o "caminho" ou a "distância" entre estas duas soluções.

## 2.7 Lista de Movimentos 2-opt

Em alguns problemas combinatórios com permutação, como por exemplo o caso do TSP, trocar elementos (os vértices representando as cidades) de posição dentro do *array* (movimentos de *swap*) não geram soluções muito úteis. Soluções melhores são geradas quando trocas de arestas (percursos) são realizadas. Este tipo de troca é utilizado em métodos de busca local e é conhecido como heurística de refinamento *k-opt*. Nesta operação, $k$ arcos (ou percursos) em uma configuração são removidos e substituídos por outros novos $k$ arcos resultando em uma nova configuração. A Figura 2.7 ilustra o caso de um refinamento *2-opt* em que 2 arcos de uma configuração são removidos e substituídos por 2 outros gerando uma nova configuração.

A nova configuração obtida com o refinamento 2-opt mostrada na Figura 2.7 é equivalente a inverter o percurso entre as duas inserções realizadas. Na forma de um *array*, as duas configurações são escritas conforme a Eq. 2.12.

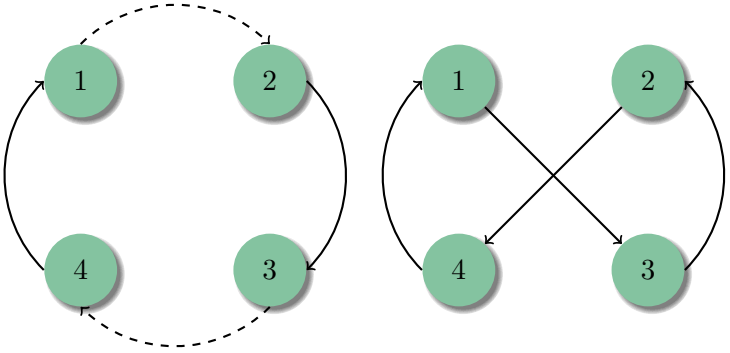



FIGURA 2.7: Heurística de refinamento 2-opt: Os arcos tracejados são substituídos por retas gerando uma nova configuração.

$$X_{Antes} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$
$$X_{Depois} = \begin{bmatrix} 1 & 3 & 2 & 4 \end{bmatrix} \tag{2.12}$$

Uma representação em lista de movimentos baseada neste princípio pode ser obtida através de uma lista de duplas de índices indicando as posições dos elementos do *array* onde as inserções foram realizadas e as inversões de percursos que devem ser feitas. As Figuras 2.8 à 2.11 ilustram os procedimentos necessários para se obter uma representação para lista de movimentos com inserções 2-opt realizadas na solução $X_{g,r_2}$ de modo a obter a solução $X_{g,r_1}$ (Eq. 2.10).

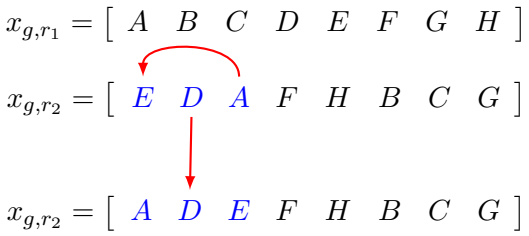



FIGURA 2.8: **Passo1:** inverter os elementos entre as inserções feitas nos índices 0 e 2 para obter o primeiro movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)]$.

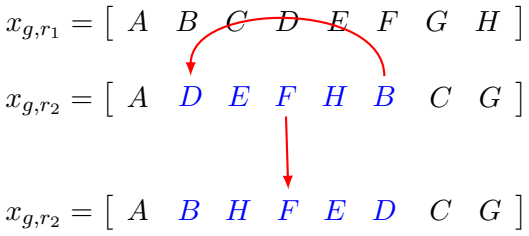



FIGURA 2.9: **Passo2:** inverter os elementos entre as inserções feitas nos índices 1 e 5 para obter o segundo movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)]$.

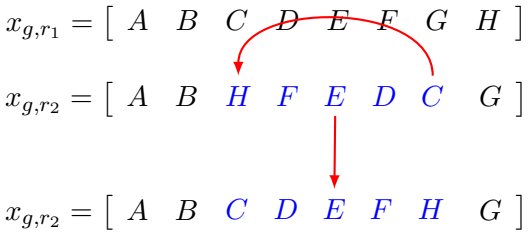



FIGURA 2.10: **Passo3:** inverter os elementos entre as inserções feitas nos índices 2 e 6 para obter o terceiro movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)]$.

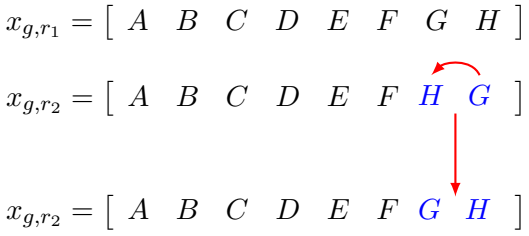



FIGURA 2.11: **Passo4:** inverter os elementos entre as inserções feitas nos índices 6 e 7 para obter o último movimento da lista: $M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(6,7)]$.

A lista de movimentos final obtida com estes movimentos é dada pela Equação 2.13 abaixo:

$$M_{X_{r_2},X_{r_1}} = [(0,2)\,(1,5)\,(2,6)\,(6,7)]\,. \tag{2.13}$$

## 2.8 Lista de Movimentos Nó-Profundidade

Uma estrutura de dados eficiente para representar problemas combinatórios cujas soluções são grafos do tipo árvore é a Representação Nó-Profundidade (RNP) apresentada em [Delbem et al. 2004, dos Santos 2009, Santos et al. 2010, Mansour et al. 2010]. Nesta representação, a codificação é realizada como uma lista contendo os nós da árvore e suas respectivas profundidades em relação à raiz. A lista é formada por *tuplas* $(n, p)$, onde $n$ é o nó e $p$ sua profundidade.

Seja, por exemplo, um problema combinatório qualquer cujos indivíduos são representados por um grafo $G_1$ como o da Figura 2.12. No grafo é identificado uma floresta com três árvores T1, T2 e T3 cujas raízes são respectivamente os nós 1, 2 e 3.

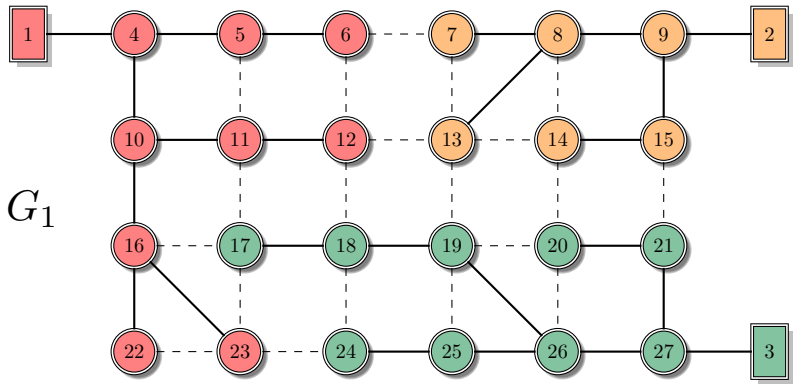



FIGURA 2.12: Representação de um indivíduo por um grafo (figura reproduzida de [dos Santos 2009] pág. 56).

As três árvores são, então, representadas, usando a RNP, por meio de uma matriz indicando os nós e suas profundidades a partir da raiz conforme mostrado na Equação 2.14. A ordem dos nós é obtida através de uma busca em profundidade realizada em cada árvore [dos Santos 2009].

$$
\begin{aligned}
T1 &= \begin{bmatrix} prof. \\ nó \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \end{bmatrix} \\
T2 &= \begin{bmatrix} prof. \\ nó \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 3 & 2 & 3 \\ 2 & 9 & 8 & 7 & 13 & 15 & 14 \end{bmatrix} \\
T3 &= \begin{bmatrix} prof. \\ nó \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix}
\end{aligned}
\tag{2.14}
$$

Dois operadores são apresentados para geração de novas florestas a partir de uma floresta de um dado grafo $G$: *i)* o *Preserve Ancestor Operator (PAO)*, que realiza uma

poda de uma subárvore a partir de um nó $p$ de uma árvore e enxerta a subárvore em um nó $e$ de outra árvore da floresta; *ii)* e o *Change Ancestor Operator (CAO)* que, além dos nós de poda e enxerto, determina um novo nó raiz $r$ para a subárvore, o qual passa a ser o nó de enxerto. O nó de enxerto deve ser um nó válido pertencente à lista de adjacência de nós do grafo $G$. A Tabela 2.1 apresenta a lista de adjacência de nós para o grafo da Figura 2.12. Uma vez obtida as RNPs das árvores de uma floresta de um grafo, novas florestas podem ser obtidas diretamente da RNP com a aplicação dos operadores PAO e CAO. A seguir, é apresentada a utilização destes dois operadores para geração de novos indivíduos.

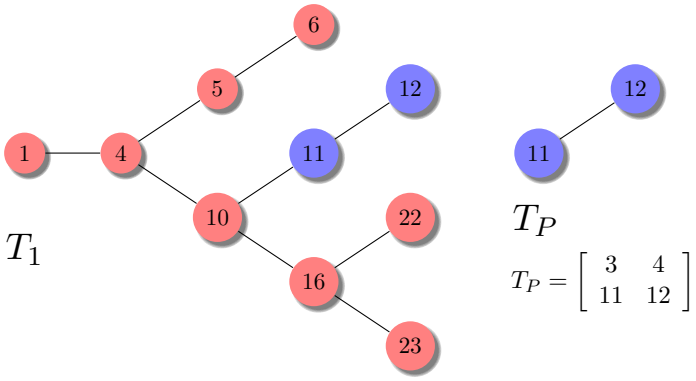
| Nó | Nós Adjacentes | | | | |
|---|---|---|---|---|---|
| 1 | 4 | | | | |
| 2 | 9 | | | | |
| 3 | 27 | | | | |
| 4 | 1 | 5 | 10 | | |
| 5 | 4 | 6 | 11 | | |
| 6 | 5 | 7 | 12 | | |
| 7 | 6 | 8 | 13 | | |
| 8 | 7 | 9 | 13 | 14 | |
| 9 | 2 | 8 | 15 | | |
| 10 | 4 | 11 | 16 | | |
| 11 | 5 | 10 | 12 | 17 | |
| 12 | 6 | 11 | 13 | 18 | |
| 13 | 7 | 8 | 12 | 14 | 19 |
| 14 | 8 | 13 | 15 | 20 | |
| 15 | 9 | 14 | 21 | | |
| 16 | 10 | 17 | 22 | 23 | |
| 17 | 11 | 16 | 18 | 23 | |
| 18 | 12 | 17 | 19 | 24 | |
| 19 | 13 | 18 | 20 | 25 | 26 |
| 20 | 14 | 19 | 21 | 26 | |
| 21 | 15 | 20 | 27 | | |
| 22 | 16 | 23 | | | |
| 23 | 16 | 17 | 22 | 24 | |
| 24 | 18 | 23 | 25 | | |
| 25 | 19 | 24 | 26 | | |
| 26 | 19 | 20 | 25 | 27 | |
| 27 | 3 | 21 | 26 | | |

TABELA 2.1: Lista de nós adjacentes do grafo G1

### 2.8.1 *Preserve Ancestor Operator*

O PAO realiza a remoção de um trecho de uma RNP e a inserção deste trecho em outra RNP. O efeito da aplicação do PAO é a transferência de uma subárvore podada, $T_p$, de

uma árvore $T_i$ para uma árvore $T_j$ da floresta. A Figura 2.13 ilustra a aplicação do PAO na árvore $T_1$ da floresta apresentada na Figura 2.12. Nela é mostrada o trecho de RNP removido e a correspondente subárvore podada com base na escolha do nó 11 da árvore $T_1$ como nó de poda. O trecho da RNP a ser removido a partir do no $p$ é determinado percorrendo a RNP para a direita a partir de $p$ até encontrar um nó com profundidade igual ou inferior a profundidade de $p$.




$$T_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \end{bmatrix}$$

FIGURA 2.13: Poda de uma subárvore de $T_1$ e respectivas RNP

A Figura 2.14 mostra a inserção do trecho de RNP na RNP de $T_3$. O nó de enxerto $e$ deve ser um dos possíveis nós adjacentes ao nó de poda (nós $5, 10, 12$ *ou* $17$) conforme sua lista de adjacência (Tabela 2.1). Como foi escolhido o nó 17 para enxerto, o trecho de RNP é inserido à direita deste nó. As profundidades dos nós do trecho inserido devem ser atualizados de acordo com a Equação 2.15:

$$p_x = p_x - p_p + p_e + 1 \tag{2.15}$$

onde: $p_x$ é a profundidade do nó a ser atualizado, $p_p$ é a profundidade do nó de poda e $p_e$ é a profundidade do nó de enxerto. Assim, as profundidades para os nós da subárvore gerada são:

$$p_{nó_{11}} = 3 - 3 + 5 + 1 = 6$$
$$p_{nó_{12}} = 4 - 3 + 5 + 1 = 7.$$

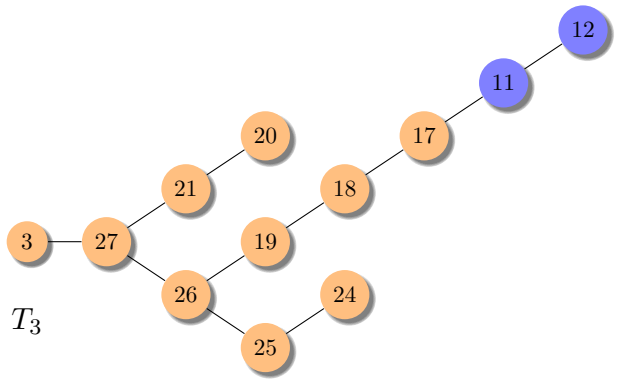



FIGURA 2.14: RNP de $T_3$ após inserção do trecho removido da RNP de $T_1$ e a nova árvore $T_3$ correspondente.

### 2.8.2 *Change Ancestor Operator*

O CAO também transfere uma subárvore podada $T_p$ gerada de uma árvore $T_i$ para uma árvore $T_j$. No entanto, um novo nó, designado como nó $r$, é escolhido entre os nós da subárvore podada para ser o nó raiz. A nova raiz pode ser qualquer nó da subárvore diferente da raiz original. A Figura 2.15 mostra a aplicação do CAO na árvore $T_1$, com o nó de poda escolhido $p = 10$ e o novo nó raiz é $r = 16$. A subárvore gerada tem, então, suas profundidades atualizadas de acordo com o novo nó raiz.

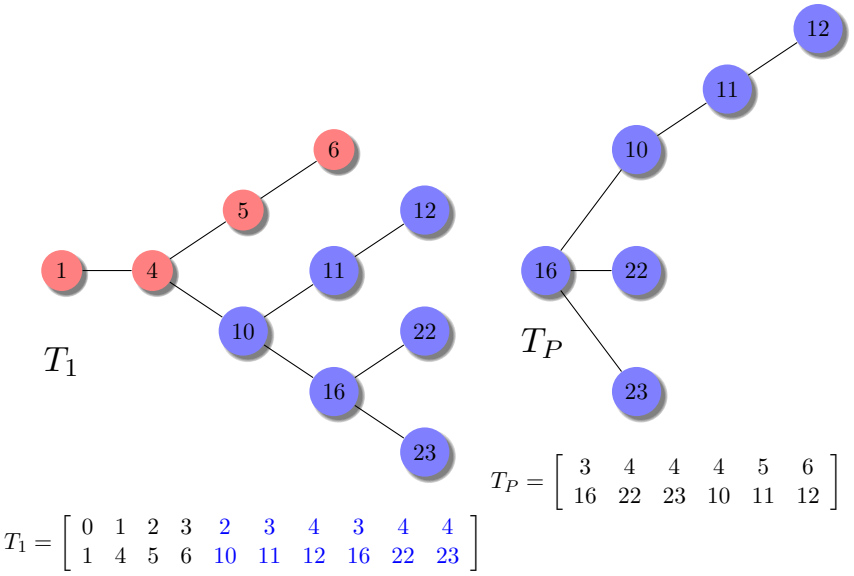



FIGURA 2.15: RNPs e correspondente subárvore gerada para $r = 16$.

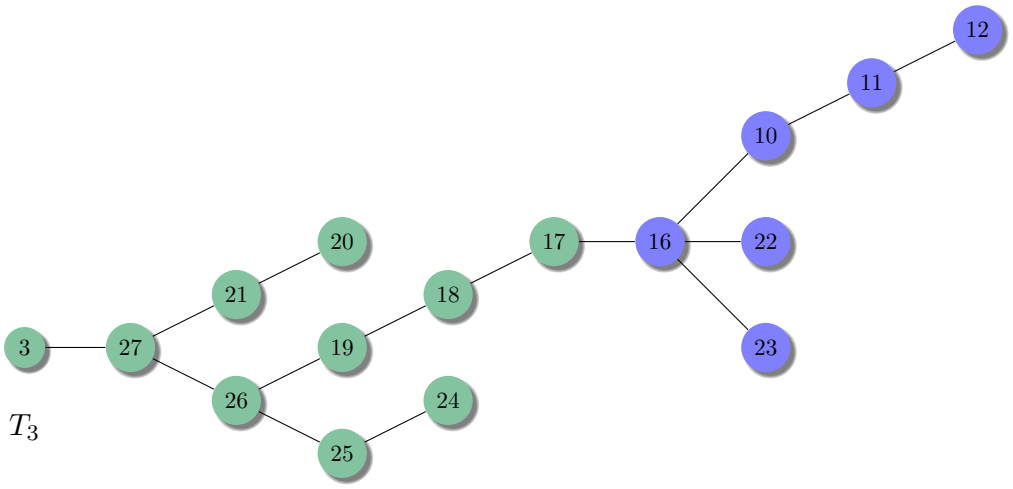
O nó de enxerto é escolhido da mesma forma como a realizada no PAO. A Figura 2.16 mostra o caso em que o nó 17 da RNP da árvore $T_3$ é escolhido para o enxerto e a sua RNP atualizada. O cálculo da nova profundidade da subárvore que contém o nó raiz é

realizado segundo a Equação 2.16.

$$p_x = p_x \; - \; p_r \; + \; p_e \; + \; 1 \tag{2.16}$$

Assim, as novas profundidades da subárvore enxertada em $T_3$ são atualizadas com as seguintes profundidades:

$$\begin{aligned} p_{nó16} &= 3-3+5+1 = 6 \\ p_{nó22} &= 4-3+5+1 = 7 \\ p_{nó23} &= 4-3+5+1 = 7 \\ p_{nó10} &= 2-1+5+1 = 7 \\ p_{nó11} &= 3-1+5+1 = 8 \\ p_{nó12} &= 4-1+5+1 = 9. \end{aligned}$$




$$T_3 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 8 & 9 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 16 & 22 & 23 & 10 & 11 & 12 & 25 & 24 \end{bmatrix}$$

FIGURA 2.16: Árvore $T_3$ após o enxerto no nó 17

A RNP juntamente com a lista de adjacências e as definições de PAO e CAO mostram-se úteis para criar uma população inicial de um problema de otimização cuja estrutura de dados é um grafo representando uma floresta (um indivíduo da população). Todas as configurações de grafo geradas são factíveis (isto é, correspondem a uma floresta com o mesmo número de árvores da floresta inicial), além disso podem ser obtidas de forma eficiente uma vez que requerem apenas "cópias e colas" de trechos de arrays e de rápida determinação. Para tanto, basta escolher aleatoriamente na população um

indivíduo, deste indivíduo um dos nós de poda dentro da floresta e, a partir dele, sortear um nó dentro da lista de adjacências para ser o nó de enxerto.

### 2.8.3 Estrutura de dados para a lista de movimentos RNP

Uma proposta de estrutura de dados para a lista de movimentos, cujo problema de otimização tenha uma estrutura de dados do tipo RNP representando os indivíduos da população, é apresentada a seguir.

Sejam $G_0$, $G_1$ e $G_2$ grafos representando três indivíduos aleatoriamente escolhidos dentro de uma população para fazerem parte da equação da mutação diferencial (Eq. 1.4), conforme mostrado na Figura 2.17 e suas respectivas RNP serem dadas pela Equação 2.17:

$$
G_0 = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 \\
1 & 4 & 5 & 11 & 12 & 6 & 10 & 16 & 23 & 22 \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & & & \\
2 & 9 & 8 & 7 & 13 & 14 & 15 & & & \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
3 & 27 & 21 & 20 & 19 & 26 & 25 & 24 & 18 & 17
\end{bmatrix}
$$

$$
G_1 = \begin{bmatrix}
0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\
1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\
2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\
3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24
\end{bmatrix}
\tag{2.17}
$$

$$
G_2 = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 2 & 3 & 4 & 3 & 4 & 4 & 4 \\
1 & 4 & 5 & 6 & 7 & 8 & 13 & 10 & 11 & 12 & 16 & 17 & 22 & 23 \\
 & & & & & & & & & & & & & \\
0 & 1 & 2 & 3 & 4 & & & & & & & & & \\
2 & 9 & 15 & 14 & 20 & & & & & & & & & \\
 & & & & & & & & & & & & & \\
0 & 1 & 2 & 2 & 3 & 3 & 4 & 5 & & & & & & \\
3 & 27 & 21 & 26 & 19 & 25 & 24 & 18 & & & & & &
\end{bmatrix}
$$

A equação da mutação diferencial é, então, escrita como:

$$
G_{mutante} = G_0 + \mathbf{F}.(G_1 - G_2) \tag{2.18}
$$

Uma representação possível para a lista de movimentos poderia ser, por exemplo, uma lista contendo triplas *(poda,raiz,enxerto)* obtidas de sucessivas aplicações de PAO e

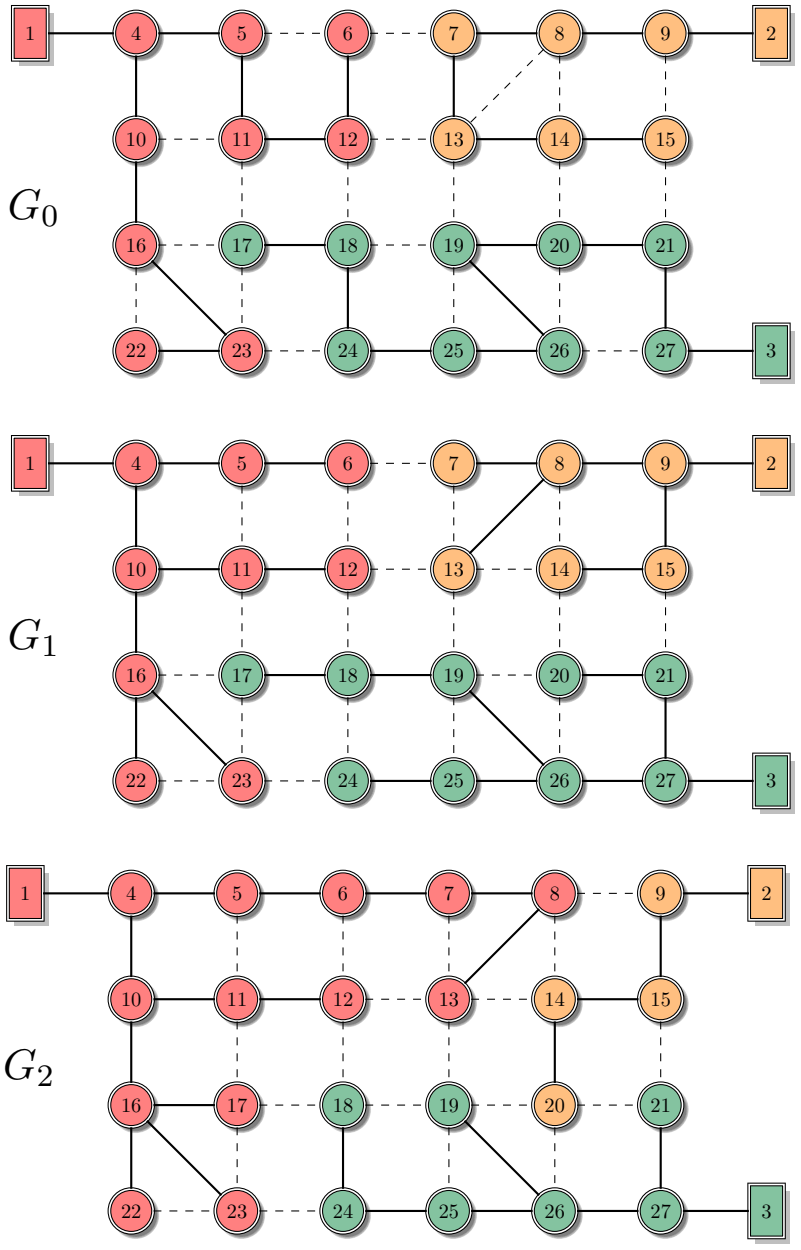



FIGURA 2.17: Indivíduos $G_0$, $G_1$ e $G_2$ aplicados a equação da mutação diferencial.

CAO na solução $G_2$ até que a solução $G_1$ seja obtida. Desta forma, a lista de movimentos conteria armazenada nessas triplas os passos necessários para, a partir da solução $G_2$, se chegar à solução $G_1$.

Comparando-se as duas soluções $G_1$ e $G_2$ na Equação 2.17, índice a índice, e supondo que os '*arrays* relativos as RNPs iniciam com índice 0, $G_2$ deve ter o seu nó 7 removido do índice 4 como o primeiro passo para se chegar à solução $G_1$. A decisão de qual operador deverá ser utilizado para a realização da poda é feita conforme descrito a seguir. Se a poda gerar uma subárvore, o CAO é utilizado, caso contrário utiliza-se o PAO. Como

a poda do nó 7 gera uma subárvore, conforme mostrado na Figura 2.18, o CAO é o escolhido. O nó raiz será escolhido entre um dos nós da subárvore gerada de modo que o nó equivalente em $G_1$ não esteja presente nesta subárvore, evitando-se assim de a geração de ciclos. A subárvore gerada possui os nós 8 e 13. O nó 13, em $G_1$, está conectado ao nó 8 que é nó da subárvore gerada, portanto, não pode ser o escolhido para ser raiz. O nó 8, em $G_1$, está conectado ao nó 9 que não está presente na subárvore gerada. Portanto ele é o escolhido para ser o nó raiz. Caso nenhum dos nós da subárvore gerada não atenda aos pré-requisitos acima, o PAO é utilizado. O nó de enxerto será aquele em que, na solução $G_1$, o nó escolhido para ser a nova raiz estiver conectado. O nó 8, em $G_1$, está conectado no nó 9. Portanto, este é nó escolhido para o enxerto da subárvore gerada. As profundidades dos nós da subárvore gerada são, então, atualizadas de acordo com a profundidade do nó de enxerto, nó 9, no indivíduo $G_2$ (Figura 2.18).

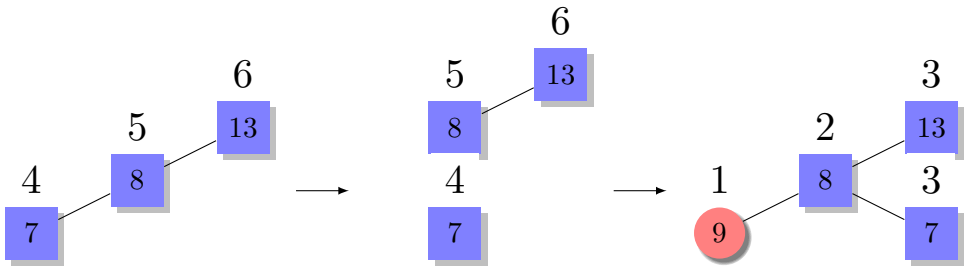



FIGURA 2.18: Primeiro movimento da lista: aplicação do CAO 2 ao nó 7 em $G_2$.

Após esta operação, a RNP de $G_2$ torna-se:

$$G_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 17 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 4 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 20 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 2 & 3 & 3 & 4 & 5 & & & \\ 3 & 27 & 21 & 26 & 19 & 25 & 24 & 18 & & & \end{bmatrix}$$

O primeiro movimento armazenado na Lista de Movimentos é a tripla:

$$M_{G_1G_2}(p, r, e) = [(7, 8, 9)]\,. \tag{2.19}$$

A próxima poda deve ser feita no nó 17, índice 8 em $G_2$. Como o nó 17 é um nó-folha, o PAO é o utilizado. O nó de enxerto será aquele em que o nó 17, em $G_1$, estiver conectado, neste caso, o nó 18. As profundidades dos nós da subárvore gerada são atualizadas de acordo com a profundidade do nó de enxerto, conforme mostrado na Figura 2.19, e à lista de movimentos é acrescentado o segundo movimento com o valor da raiz sinalizado como *null* indicando que o PAO foi o utilizado (Eq. 2.20).

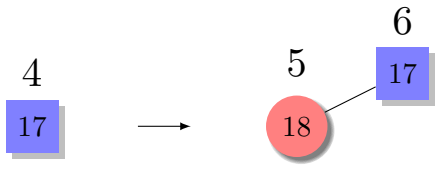



FIGURA 2.19: Segundo movimento da lista: aplicação do PAO ao nó 17.

$$M_{G_1G_2}(p, r, e) = [(7, 8, 9)\ (17, null, 18)] \tag{2.20}$$

A RNP de $G_2$ passa a ser então:

$$G_1 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{array}\right]$$

$$G_2 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 2 & 3 & 3 & & \\ 2 & 9 & 15 & 14 & 20 & 8 & 7 & 13 & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 2 & 3 & 3 & 4 & 5 & 6 & \\ 3 & 27 & 21 & 26 & 19 & 25 & 24 & 18 & 17 & \end{array}\right]$$

O próximo movimento será a remoção do nó 20 da posição de índice 14 em $G_2$. O nó 20 é um nó-folha, portanto é aplicado o PAO. Em $G_1$, o nó 20 está conectado no nó 21. Assim, o nó de enxerto será o nó 21. Após a atualização das profundidades dos nós da subárvore gerada, de acordo com a profundidade do nó 21 em $G_2$ (Fig. 2.20), a lista de movimentos passa a ser a mostrada na Equação 2.21.

$$M_{G_1G_2}(p, r, e) = [(7, 8, 9)\ (17, null, 18)\ (20, null, 21)] \tag{2.21}$$

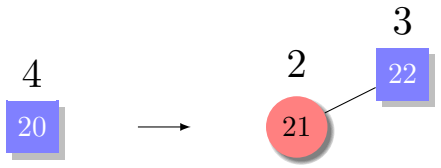



FIGURA 2.20: Aplicação do PAO ao nó 20.

E a RNP de $G_2$ é atualizada para:

$$G_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & 4 & 5 & 6 \\ 3 & 27 & 21 & 20 & 26 & 19 & 25 & 24 & 18 & 17 \end{bmatrix}$$

Com esses movimentos, a primeira e segunda árvores em $G_2$ já são idênticas às de $G_1$. Os próximos movimentos serão os necessários para modificar a terceira árvore de $G_2$. Para tanto, o nó 25 deve ser podado da posição de índice 23 em $G_2$. Esta poda gera uma subárvore e, portanto, é utilizado o CAO. O nó raiz deve, então, ser escolhido entre os nós subadjacentes ao nó de poda da subárvore gerada (Figura 2.21). O nó 24, em $G_1$, está conectado ao nó 25. Como este nó é um nó da subárvore gerada ele é descartado. O mesmo ocorre com o nó 17 que, em $G_1$, está conectado ao nó 18 que também faz parte da subárvore gerada. O nó 18, por sua vez, em $G_1$, está conectado ao nó 19 e é o escolhido para ser o nó raiz e o nó 19 em $G_1$ o nó de enxerto.

Adicionando-se o quarto movimento à Lista de Movimentos, tem-se:

$$M_{G_1G_2}(p, r, e) = [(7, 8, 9)\ (17, null, 18)\ (20, null, 21)\ (25, 18, 19)]\,. \tag{2.22}$$

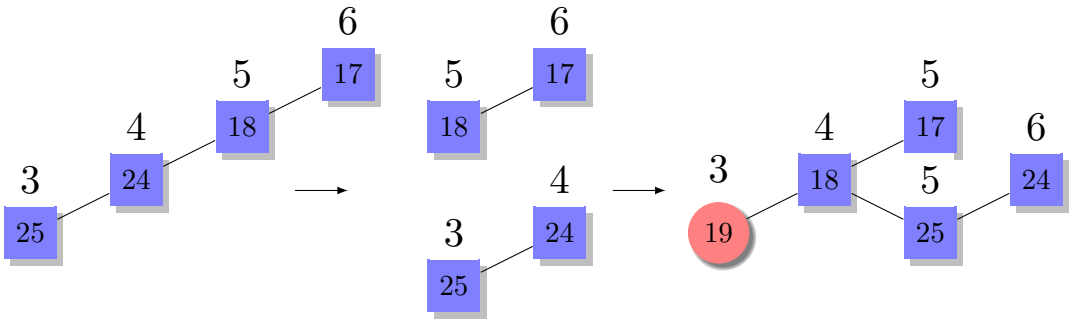



FIGURA 2.21: Aplicação do CAO ao nó 25.

E a RNP de $G_2$ é atualizada para:

$$G_1 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{array}\right]$$

$$G_2 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 5 & 6 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{array}\right]$$

O último movimento da lista de movimentos será o de podar o nó 25 da posição de índice 26 em $G_2$[5]. Como esta poda gera uma subárvore formada pelos nós 25 e 24, o CAO deveria ser o escolhido para realizar esta operação. Entretanto, como o nó subadjacente ao nó de poda, nó 24, não pode ser o nó raiz pois, em $G_1$, ele está conectado ao nó 25 que é nó da subárvore gerada, o PAO deve ser utilizado. Em $G_1$, o nó 25 está conectado ao nó 26 que passa a ser o nó de enxerto (Figura 2.22).

Adicionando-se este movimento à lista obtém-se Lista de Movimentos final:

$$M_{G_1G_2}(p,r,e) = [(7,8,9)\ (17,null,18)\ (20,null,21)\ (25,18,19)\ (25,null,26)]\,. \quad (2.23)$$

[5]Observe que em $G_1$ o nó 25, com profundidade 3, está conectado ao nó 26 (profundidade 2) e que em $G_2$ o nó 25, com profundidade 5, está conectado ao nó 18 (profundidade 4).

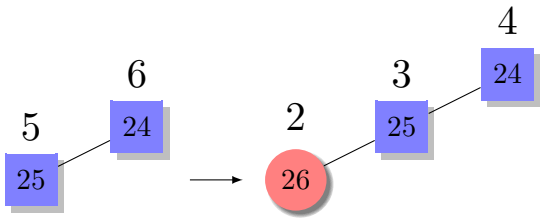



FIGURA 2.22: Aplicação do PAO ao nó 25.

Assim, a solução $G_2$ "chega" à solução $G_1$:

$$G_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 3 & & & \\ 2 & 9 & 15 & 14 & 8 & 7 & 13 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix}.$$

A Lista de Movimentos, após ser ponderada pelo fator escalar **F**, deve ser, então, aplicada ao vetor base ($G_0$), reescrito na Equação 2.24, para se determinar o vetor mutante ($G_{mutante}$).

Em geral na literatura especializada, o fator escalar **F** é escolhido entre o intervalo $[0,4;\ 0,6]$. Entretanto, como um primeiro exemplo de aplicação do fator escalar **F** à lista de movimentos, ele é escolhido ser igual a 1 de modo que todos os movimentos da lista possam ser aplicados à solução $G_0$. Se, por exemplo, fosse escolhido um fator escalar $\mathbf{F} = 0,6$ somente os 60% primeiros elementos da lista seriam aplicados a solução $G_0$.

$$G_0 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 \\ 1 & 4 & 5 & 11 & 12 & 6 & 10 & 16 & 23 & 22 \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & & & \\ 2 & 9 & 8 & 7 & 13 & 14 & 15 & & & \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 27 & 21 & 20 & 19 & 26 & 25 & 24 & 18 & 17 \end{bmatrix} \tag{2.24}$$

O primeiro movimento da lista (7,8,9) determina que uma poda deve ser feita no nó 7 de $G_0$ (gerando a subárvore 7-13-14-15) e que esta subárvore deve ter como raiz o nó 8 e este, por sua vez, enxertado no nó 9. Se este movimento fosse aplicado, a solução $G_0$ permaneceria inalterada uma vez que a estrutura gerada por este movimento já existe. Este movimento, portanto, é descartado.

O segundo movimento da lista (17,*null*,*18*) determina que uma poda deve ser realizada no nó 17 (gerando um nó-folha) e esse nó deve ser enxertado ao nó 18. Este movimento também não apresenta nenhuma alteração na solução $G_0$ e ele, então, é descartado. O mesmo ocorre com o terceiro movimento (20,*null*,21).

O quarto movimento da lista (25,18,19) determina uma poda no nó 25, escolher o nó 18 para ser o nó raiz da subárvore gerada e este ser enxertado ao nó 19 (Figura 2.23).

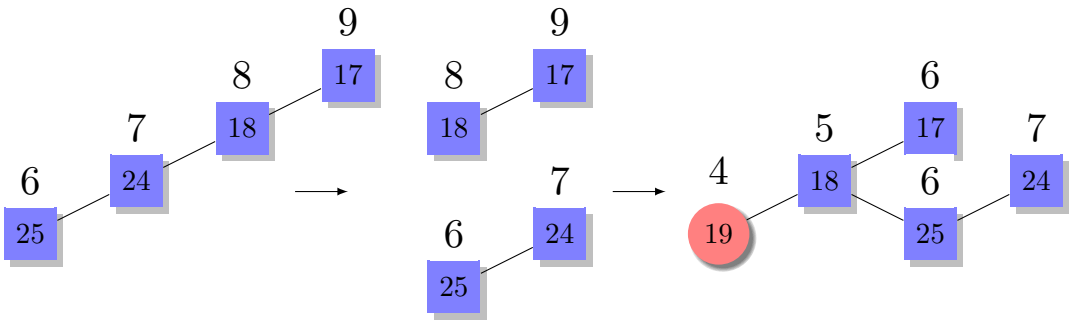



FIGURA 2.23: Aplicação do quarto movimento da lista ao vetor base.

Com este movimento aplicado, o vetor base passa a ser:

$$G_0 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 \\ 1 & 4 & 5 & 11 & 12 & 6 & 10 & 16 & 23 & 22 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & & & \\ 2 & 9 & 8 & 7 & 13 & 14 & 15 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 5 & 6 & 6 & 7 \\ 3 & 27 & 21 & 20 & 19 & 26 & 18 & 17 & 25 & 24 \end{bmatrix}.$$

O quinto e último movimento da lista (25,*null*,26) determina a poda do nó 25 e seu enxerto no nó 26 como mostrado na Figura 2.24.

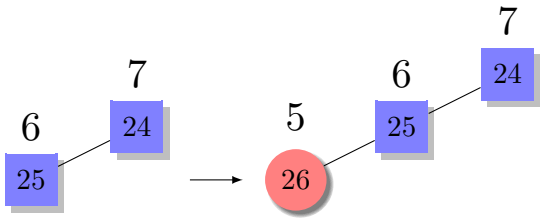



FIGURA 2.24: Aplicação do quinto movimento da lista ao vetor base.

O vetor mutante resultante da aplicação dos movimentos da lista de movimentos ao vetor base é mostrado na equação abaixo e sua representação em grafo na Figura 2.25.

$$G_{mutante} = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 \\ 1 & 4 & 5 & 11 & 12 & 6 & 10 & 16 & 23 & 22 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & & & \\ 2 & 9 & 8 & 7 & 13 & 14 & 15 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 5 & 6 \\ 3 & 27 & 21 & 20 & 19 & 26 & 25 & 24 & 18 & 17 \end{array}\right]$$

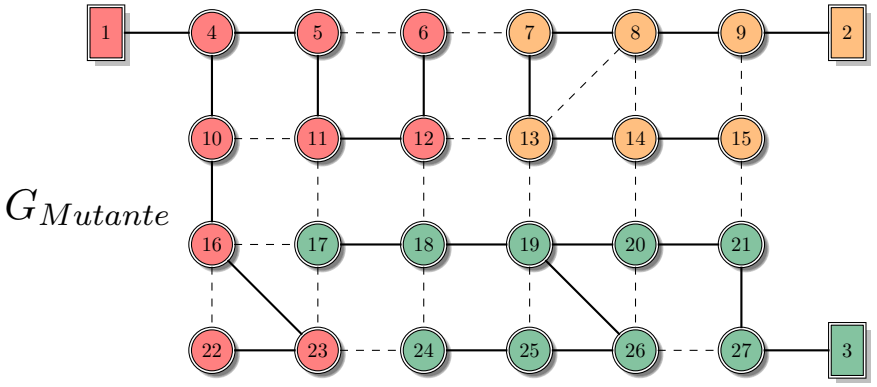



FIGURA 2.25: Vetor mutante resultante da mutação diferencial.

## 2.9 Resumo

Neste capítulo foram apresentadas as principais abordagens para a DE encontradas na literatura, e uma nova técnica foi proposta: a abordagem por Lista de Movimentos. Estas abordagens apresentadas para a DE discreta foram classificadas em dois principais grupos: Um grupo, em que a equação da mutação diferencial no domínio das variáveis contínuas é aplicada diretamente em problemas combinatórios, caso da IRP e da FTB, e o outro em que a mutação diferencial é redefinida para a classe de problemas combinatórios,

como ocorre com a Abordagens AMP, AMA e para a abordagem proposta *Lista de Movimentos*.

No primeiro grupo, observa-se a necessidade de algum tipo de reparo nas soluções geradas, ora devido a valores repetidos encontrados para as soluções, ora devido a soluções encontradas fora do espaço de busca do problema. Isto se deve ao fato de que os operadores aritméticos da equação da mutação diferencial não são definidos no espaço das variáveis discretas e, portanto, não podem ser aplicados em problemas de natureza combinatória. No segundo grupo, as AMP e AMA não são abrangentes e são, tão somente, aplicáveis em problemas combinatórios com permutação. Além disso, necessitam de mecanismos de reparos em alguns casos.

A abordagem por lista de movimentos, devido as definições adotadas para os operadores aritméticos da equação da mutação diferencial, é genérica e aplicável a outros problemas de otimização combinatória. Essa flexibilidade deve-se ao fato de que os movimentos da lista de movimentos são definidos especificamente para cada tipo de problema combinatório. De acordo com o problema proposto os movimentos da lista podem ser representados por *arrays*, matrizes, grafos, etc; dependendo da estrutura de dados utilizada para representar as soluções do problema.

Na abordagem proposta nenhum mecanismo de reparo das soluções geradas se faz necessário e a diferença vetorial do DE, observada no domínio das variáveis contínuas, é, de alguma forma, preservada no domínio das variáveis discretas devido a lista de movimentos armazenar a "diferença" (ou a "distância") entre as duas soluções candidatas na equação da mutação diferencial.

# Capítulo 3

# Restauração de Sistemas de Distribuição de Energia Elétrica

Este capítulo discorre acerca do problema de ocorrência de contingências em um sistema elétrico de potência na fase de distribuição de energia, no caso específico da restauração de serviço após a ocorrência de uma falta ou manutenção preventiva em um dado setor do sistema de distribuição primária. Primeiramente é apresentado o problema de restauração de sistemas de distribuição de energia, sua formulação matemática e, em seguida, é apresentada uma versão do algoritmo DE proposto neste trabalho para a solução deste tipo de problema.

## 3.1 Sistema de Distribuição de Energia Primária

Os sistemas elétricos de potência são sistemas que englobam três partes principais: a geração, a transmissão e a distribuição da energia elétrica. A geração, formada pelas usinas, tem a função de converter alguma forma de energia em energia elétrica. A transmissão, composta basicamente por linhas de transmissão, é responsável pelo transporte da energia elétrica vinda dos pontos de geração até os consumidores. A distribuição é composta por subestações abaixadoras e linhas de distribuição de energia e é responsável pela distribuição de energia elétrica, recebida do sistema de transmissão, até os centros consumidores. A Figura 3.1 ilustra um sistema elétrico de potência enfatizando estas três principais fases.

O sistema de distribuição pode ser dividido em Sistema de Distribuição Primária, parte do sistema em que o algoritmo DE será aplicado, e Sistema de Distribuição Secundária. O sistema de distribuição primária opera geralmente em redes radiais aéreas

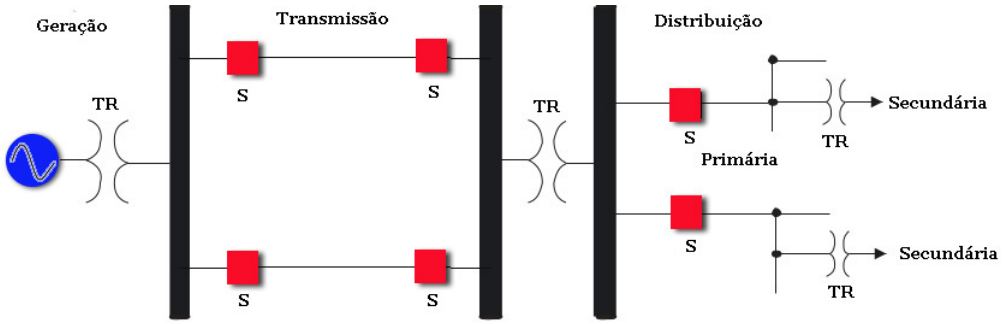



FIGURA 3.1: Sistema elétrico de potência, onde: TR - transformadores, S - chaves seccionadoras.

na tensão de $13,8kV$ e atendem consumidores tais como as indústrias de médio porte, iluminação pública, grandes comércios, hospitais, etc; e aos transformadores de distribuição que suprem o Sistema de Distribuição Secundária ou de Baixa Tensão. O sistema de distribuição secundária opera com as tensões de $220/127V$ ou de $380/220V$ e supre consumidores tais como pequenos comércios, industrias e consumidores residenciais.

Os sistemas de distribuição de energia primária são muito complexos e apresentam um grande desafio no que diz respeito ao seu controle e sua operação. Por questões de custos operacionais e de investimentos eles, em geral, operam em configurações radiais possibilitando alterações da topologia através de abertura e fechamento de chaves seccionadoras localizadas em pontos estratégicos. Se por um lado esta configuração radial tem por objetivo simplificar a operação e proteção da rede, por outro lado diminui a confiabilidade do sistema em relação à continuidade do fornecimento de energia aos consumidores, seja em consequência de manutenções corretivas ou seja por execuções de serviços de manutenção preventiva que normalmente se fazem necessários ao bom funcionamento do sistema. Com o intuito de melhorar a confiabilidade destes sistemas radiais, sem incorrer em gastos excessivos, vários pontos de cargas são agrupados em blocos (denominados setores) e separados for chaves que operam no estado Normalmente Aberto (NA) e Normalmente Fechado (NF). Desta forma, em caso de interrupção de energia em algum ponto da rede, a operação de chaves permite a troca de cargas entre os circuitos do sistema de modo a atender o maior número possível de consumidores.

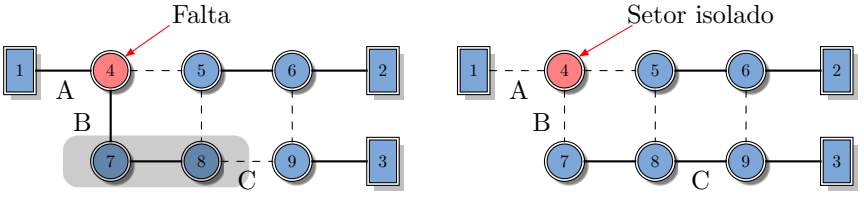
O agrupamento em setores permite, através da utilização de algoritmos de restauração de redes, restringir as interrupções a uma menor parte possível da rede de distribuição. Em condições normais de operação, estes algoritmos são utilizados para reduzir as perdas totais por efeito joule, balanceamento de carga entre alimentadores, redução de queda de tensão e aliviar trechos da rede com sobrecarga. Eles também são utilizados no planejamento e expansão dos Sistemas de Distribuição de Energia (SDE) e na ocorrência de contingências como, por exemplo, a restauração do sistema em caso de ocorrência de uma falta ou manutenção preventiva do sistema. Neste caso é necessário

determinar quais chaves NF e/ou NA deverão ser abertas e/ou fechadas a fim de isolar trechos da rede com defeito e alimentar os setores a jusante da falta.

## 3.2 O Problema de Restauração

A Restauração de Serviço (RS) em um SDE consiste em, após a localização e o isolamento do setor em falta, reconfigurar o sistema através de aberturas e fechamentos de chaves NF e NA de tal forma a atender o maior número possível de consumidores. A Figura 3.2 exemplifica a RS através da restauração de um SDE constituído por três alimentadores. Na figura, os retângulos representam as fontes de energia dos alimentadores, as linhas sólidas chaves NF, as linhas tracejadas chaves NA e os círculos indicam os setores[1]. A Figura 3.2(a) indica uma falta no setor 4 que deve ser, portanto, isolado do resto do sistema através da abertura das chaves **A** e **B** até que o problema no setor seja sanado. Entretanto, ao isolar o setor 4, os setores 7 e 8 (região sombreada na figura) ficam desconectados do sistema pois os mesmos são alimentados através deste setor em falta. Uma possibilidade de restaurar a energia para estes setores seria o fechamento da chave **C** ligando estes setores ao alimentador 3, como mostrado na Figura 3.2(b). Assim, o problema de restauração de redes pode ser resumido, em poucas palavaras [dos Santos 2009], como:

- Reduzir o número de consumidores sem o fornecimento de energia através do fechamento e abertura de chaves seccionadoras dos setores.
- Reduzir o número de manobras, ou seja, o ligamento e desligamento destas chaves.
- Garantir que nenhum componente da rede esteja sobrecarregado, reduzir as perdas de potência total e a queda de tensão.
- Manter a forma radial do sistema de distribuição.




(a) Setor em falta (b) Nova configuração

FIGURA 3.2: Restauração de serviço em um SDE com 3 alimentadores. Figura reproduzida de [dos Santos 2009]

[1]Os setores em um SDE são conjuntos de barras conectadas por linhas sem a presença de chaves.

## 3.3 Formulação Matemática

Os SDE são sistemas normalmente representados por grafos pois os mesmos permitem uma melhor visualização da sua complexidade além de ser uma ferramenta muito útil na simulação e/ou resolução de problemas de restauração ou expansão do sistema. Assim, a seguir são apresentados alguns conceitos básicos da teoria de grafos que serão importantes para o entendimento das seções seguintes deste capítulo.

*Definição* 3.1. : Um grafo G é um par $(V, E)$ onde $V$ é um conjunto finito de elementos denominados vértices (ou nós) e $E$ é um conjunto não ordenado de pares $v = (x, y)$, $\forall\, x, y \in V$, denominados arestas de $V$.

Seja, por exemplo, $V$ o conjunto de setores do SDE mostrado na Figura 3.2(a)

$$V = \{s \mid s\,\acute{e}\,um\,setor\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

e $E$ um conjunto de pares de setores que estão conectados:

$$E = \{(x, y) \mid x\,est\acute{a}\,ligado\,em\,y\} = \{(1,4)(4,7)(7,8)(2,6)(6,5)(3,9)\}$$

Desta forma, a configuração corrente do SDE da Figura 3.2(a) pode ser representada por um grafo representando uma floresta constituída por três árvores, como mostrado na Figura 3.3.

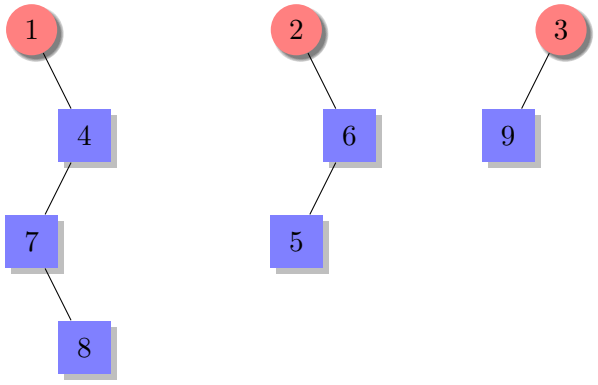



FIGURA 3.3: Representação em grafo do SDE da Fig. 3.2(a)

Da teoria dos grafos são apresentadas as seguintes definições importantes para a representação de um SDE:

*Definição* 3.2. Se $x$ e $y$ são dois nós de um grafo e o par $(x, y)$ é uma aresta, denotada por $\boldsymbol{e}$, diz-se que $\boldsymbol{e}$ conecta $(x, y)$.

*Definição* 3.3. O ***grau*** de um nó $x$ em um grafo G é dado pelo número de arestas que incindem em $x$: O grau do nó 7 é 2. O de 2 é 1.

*Definição* 3.4. Uma sequência de arestas distintas em um grafo é chamada de ***caminho***: A sequência $(1,4)(4,7)(7,8)$ no grafo da Figura 3.3 é um caminho.

*Definição* 3.5. Um caminho entre os nós de um grafo é chamado de ***ciclo*** se ao partimos de um nó retornamos a ele. No grafo da Figura 3.3 não há ciclos.

*Definição* 3.6. Um grafo G é um ***grafo conexo*** se todo par de nós em G é um par conexo. O grafo da Figura 3.3 não é um grafo conexo pois não existem caminhos entre todos os nós dos grafos.

*Definição* 3.7. Uma **árvore** é um grafo sem ciclos. A Figura 3.3 é um grafo que contém três árvores formando uma floresta, ou seja, uma floresta é um grafo formado por um conjunto de árvores.

*Definição* 3.8. Duas arestas são adjacentes se elas incidem sobre o mesmo nó: as arestas $(1,4)\, e\, (4,7)$ são adjacentes.

*Definição* 3.9. Dois nós, $x$ e $y$, em um grafo são adjacentes se existe uma aresta $e = (x,y)$ em G: $6\, e\, 5$ são nós adjacentes

Grafos também podem ser representados por uma Lista de Adjacências de nós onde, para cada nó do grafo, são listados seus nós adjacentes. A Tabela 3.1 apresenta a Lista de Adjacências para o grafo da Fig 3.3.

| Nó | Nós Adjacentes |
|---|---|
| 1 | 4 |
| 2 | 6 |
| 3 | 9 |
| 4 | 1 7 |
| 5 | 6 |
| 6 | 2 5 |
| 7 | 4 8 |
| 8 | 7 |
| 9 | 3 |

Tabela 3.1: Lista de nós adjacentes do grafo da Fig. 3.3.

Com as definições acima, com os setores representados por nós, as chaves seccionadoras NA e NF representadas por arestas e as subestações alimentadoras por nós raízes, um grafo do tipo árvore na RNP é utilizado para modelar um SDE e seus operadores para evoluir a população corrente.

A formulação para o problema de restauração de redes de distribuição de energia pode ser obtida considerando todos os objetivos e restrições envolvidas. Os objetivos são os de minimizar o número de áreas fora de serviço, o número de chaveamentos necessários para a restauração do sistema e o total de perdas de potência sem violar as restrições

de carga e tensão. As restrições a serem respeitadas, após o isolamento dos setores afetados e a área fora de serviço ter sido reconectada ao sistema, são o de manter a estrutura radial do sistema, respeitar a capacidade limite de cada alimentador, respeitar a capacidade da corrente elétrica das linhas e chaves e o limite permissível para a queda de tensão nas barras do SDE. A minimização do número de chaveamento é importante porque o tempo necessário para a restauração do SDE depende basicamente do número de operações de desligamento e ligamento de chaves. Assim, o problema de restauração do SDE pode ser formulado como [dos Santos 2009, Santos et al. 2010]:

$$\begin{aligned} Minimizar: \quad & \phi(G), \psi(G, G^0)\, e\, \gamma(G) \\ sujeito\, a: \quad & \\ & \mathbf{Ax} = \mathbf{b} \\ & X(G) \leq 1 \\ & B(G) \leq 1 \\ & V(G) \leq 1 \end{aligned} \tag{3.1}$$

em que:

- $G$ é o grafo representado a configuração do SDE, onde cada árvore da floresta corresponde ao alimentador conectado a subestação.
- $G^0$ é a configuração do SDE antes da ocorrência da falta.
- $\phi(G)$ é o número de consumidores fora da área de serviço devido a falta no SDE.
- $\psi(G, G^0)$ é o número de operações de chaveamento necessário para, a partir da configuração $G^0$, atingir a configuração $G$.
- $\gamma(G)$ é a perda resistiva, em p.u., da configuração $G$.
- $\mathbf{A}$ é a matriz incidente de $G$.
- $\mathbf{x}$ é o vetor corrente de linha (constante) nos barramentos.
- $\mathbf{b}$ é o vetor contendo as correntes de carga nas barras (se $b_i \leq 0$) ou as correntes injetadas nas subestações (se $b_i \geq 0$).
- $X(G)$ é o carregamento da rede da configuração $G$, isto é, $X(G)$ é a maior taxa $x_j/\overline{x}_j$, onde $\overline{x}_j$ é o limite superior para cada corrente de linha $x_j$ na linha $j$.
- $B(G)$ é o carregamento das subestações da configuração $G$, isto é, $B(G)$ é a maior taxa $b_s/\overline{b}_s$ onde $\overline{b}_s$ é o limite superior da corrente injetada na subestação ($s$ é a subestação no barramento).
- $V(G)$ é a máxima queda de tensão na configuração $G$, isto é, $V(G)$ é o maior valor de $\mid v_s - v_k \mid /\delta$, onde $v_s$ é a tensão de nó no barramento da subestação $s$ em p.u. e $v_k$ é a tensão de nó no barramento $k$ em p.u. e $\delta$ é a máxima queda de tensão aceitável (neste trabalho $\delta = 0,1$, isto é, a queda de tensão é limitada em 10%).

A formulação da Equação 3.1 pode ser sintetizada considerando-se:

1. Penalidades para as violações das restrições de carregamento da rede e das subestações, $X(G)\,e\,B(G)$, e da queda de tensão $V(G)$;
2. O uso da RNP [Delbem et al. 2004] para garantir que as modificações realizadas na configuração do SDE sempre produzam uma nova configuração $G$ que também é uma floresta (uma configuração factível). Por meio desta codificação garante-se as restrições da topologia da rede;
3. Os nós são arranjados na Ordem Terminal-Subestação (TSO)[2] [Srinivas 2000, Shirmohammadi et al. 1988, Das et al. 1994] para cada configuração $G$ produzida de maneira a resolver **Ax=b** usando o algoritmo SLFA[3] para distribuição de sistemas [Santos et al. 2008];
4. $\phi(G) = 0$: A RNP sempre gera florestas que correspondem a redes sem consumidores fora-de-serviço.

Desta maneira, neste trabalho é adotada a seguinte função mono-objetiva sem restrições:

$$Minimizar: \quad \psi(G, G^0) + \gamma(G) + \hat{\omega}_x X(G) + \hat{\omega}_b B(G) + \hat{\omega}_v V(G) \tag{3.2}$$

em que $G$ é a floresta gerada pela RNP e o fluxo de carga calculado usando a RNP.

Os multiplicadores $\hat{\omega}_x$, $\hat{\omega}_b\,e\,\hat{\omega}_v$ são pesos penalizando as restrições operacionais do SDE. Neste trabalho, essas penalidades são dadas por:

$$\hat{\omega}_x = \begin{cases} \omega_x, \text{ se } X(G) > 1 \\ 0, \text{ caso contrário;} \end{cases}$$

$$\hat{\omega}_b = \begin{cases} \omega_b, \text{ se } B(G) > 1 \\ 0, \text{ caso contrário;} \end{cases}$$

$$\hat{\omega}_v = \begin{cases} \omega_v, \text{ se } V(G) > 1 \\ 0, \text{ caso contrário;} \end{cases}$$

em que $\omega_x$, $\omega_b\,e\,\omega_v$ são valores positivos.

A função objetivo dada pela Equação 3.2 é não-linear, descontínua e contendo vários ótimos locais, sendo portanto, necessário a utilização de métodos heurísticos tais como os algoritmos evolucionários.

[2] Do inglês: Terminal-Substation Order
[3] Do inglês: Sweep Load Flow Algorithm

## 3.4 DE Aplicada ao Problema de Restauração de SDE

Diversos métodos têm sido propostos na literatura especializada para a otimização de problemas de restauração de SDE. Estes métodos podem ser agrupados em duas grandes linhas: a da Programação Matemática e a baseada em heurísticas. Na programação matemática destacam-se os métodos de Programação Inteira Mista [Sun et al. 1982, Paiva et al. 2005], Programação Não-linear [El-Khattam et al. 2005], Programação Dinâmica [Boulaxis and Papadopoulos 2002, Díaz-Dorado and Pidre 2004] e a Programação Linear [Farrag et al. 1999]. Entretanto, devido ao grande número de variáveis envolvidas no problema de restauração de redes de distribuição, seja de médio ou grande porte, a aplicação da programação matemática neste tipo de problema torna-se inviável uma vez que o número de soluções possíveis aumenta exponencialmente com o número de chaves seccionadoras: o número de soluções possíveis para este tipo de problema é dado pelo número de chaves NA vezes o número de chaves NF para 1 par de manobras. Para $n$ pares de manobras, $(NA \times NF)^n$ soluções possíveis devem ser investigadas em um algoritmo de busca exaustiva [dos Santos 2009]. Em contrapartida, os algoritmos baseados em heurísticas têm sido uma solução alternativa para resolver este tipo de problema uma vez que, mesmo que a solução encontrada não seja a ótima, eles garantem soluções adequadas ou quase ótimas em um tempo computacional relativamente pequeno. Estes algoritmos são utilizados geralmente em problemas onde não são conhecidos métodos exatos eficientes que garantem uma solução ótima.

Dos algoritmos baseados em heurística os AEs têm sido aplicados, nas últimas décadas, na resolução de um grande número de problemas de restauração de SDE com grande relevância em problemas multiobjetivos [Mendoza et al. 2006, Deb et al. 2006]. Ainda assim, na sua grande maioria, eles ainda demandam um tempo computacional muito grande quando aplicados em sistemas de larga escala [Delbem et al. 2005]. Além disso, o desempenho obtido com estes algoritmos quando aplicado em problemas de restauração de SDE é afetado pela estrutura de dados utilizada na representação topológica do sistema bem como os operadores genéticos utilizados na evolução da população: geralmente os operadores utilizados geram soluções cuja configuração não são radiais, necessitando de operadores ou rotinas de reparo adicionais [Carreno et al. 2008].

Em contrapartida, o desempenho dos AEs aplicados em problemas de restauração de SDE tem sido aprimorado utilizando a RNP e seus operadores genéticos (vide Capítulo 2.8). Os operadores genéticos propostos nesta representação produzem exclusivamente configurações factíveis, ou seja, redes radiais capazes de suprir energia a todo o sistema reconfigurado. As configurações geradas são mais significativas em relação a outras representações uma vez que, em um mesmo tempo de execução, o tempo médio de execução é $O(\sqrt{n})$, onde $n$ é o número de nós do grafo. As soluções geradas garantem

que todos os nós são ordenados segundo a relação denominada Modelo Pai-Filho (MPF) que, por sua vez, garante a execução de um fluxo de carga de varredura direta/inversa mais eficiente [dos Santos 2009].

Nesta conjuntura, as vantagens proporcionadas pela RNP são utilizadas no algoritmo DE, proposto neste trabalho, para representar a estrutura de dados do problema de restauração e seus operadores para evoluir a população de indivíduos. Isto é possível devido as definições para a lista de movimentos do algoritmo terem a flexibilidade de serem formuladas de acordo com o problema combinatório em questão e da estrutura de dados utilizada para representá-lo. Entretanto, para se utilizar a RNP juntamente com o algoritmo DE, aqui denominado DE-Tree[4], é necessário definir uma *Árvore de Ancestralidade* utilizada para construir a lista de movimentos e armazenar as manobras[5] que serão necessárias para a nova configuração do SDE. As seções seguintes deste Capítulo apresentam um exemplo ilustrativo da aplicação do algoritmo DE-Tree na resolução de problemas de restauração de SDE utilizando esta representação de dados.

## 3.5 DE-Tree: DE com Árvore de Ancestralidade Exemplo Ilustrativo

O algoritmo DE-Tree, versão discreta do DE com árvore de ancestralidade aplicada ao problema de restauração de SDE, assim como na sua versão original contínua, é composto pelas etapas de inicialização da população, a mutação diferencial, a fase de cruzamento e a de seleção. Cada uma destas etapas é descrita através de um exemplo ilustrativo para o problema de restauração de um SDE.

### 3.5.1 Inicialização da População

Como muitas meta-heurísticas baseadas em populações, o algoritmo DE-Tree é inicializado por uma população gerada aleatoriamente. Para o problema de restauração de SDE a população inicial é gerada a partir da configuração corrente através de abertura e fechamento de chaves NA e NF aleatoriamente escolhidas gerando, assim, novos indivíduos. O grafo mostrado na Figura 3.4 representa a configuração corrente de um SDE simplificado utilizado para ilustrar o exemplo de aplicação do algoritmo DE-Tree na restauração de SDE. Nele são identificadas três árvores cujas raízes, os alimentadores do SDE, são os retângulos 1, 2 e 3; as chaves NF, indicadas por linhas contínuas, e

[4]Do inglês *Differential Evolution with Ancestor Tree*
[5]Sequência de abertura e fechamento de chaves NA e NF necessária para reconfigurar o SDE

as chaves NA, indicadas por linhas pontilhadas, conectam os vários setores do sistema indicados por círculos.

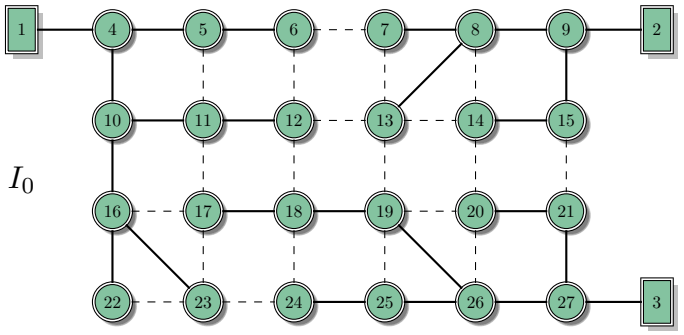



FIGURA 3.4: Indivíduo $I_0$: Grafo de um SDE simplificado

A lista de adjacências indicando as possíveis ligações entre os setores deste SDE é aquela dada pela tabela 2.1, apresentada no Capítulo 2, e reproduzida abaixo para maior facilidade de leitura.

| Nó | Nós Adjacentes | | | | |
|---|---|---|---|---|---|
| 1 | 4 | | | | |
| 2 | 9 | | | | |
| 3 | 27 | | | | |
| 4 | 1 | 5 | 10 | | |
| 5 | 4 | 6 | 11 | | |
| 6 | 5 | 7 | 12 | | |
| 7 | 6 | 8 | 13 | | |
| 8 | 7 | 9 | 13 | 14 | |
| 9 | 2 | 8 | 15 | | |
| 10 | 4 | 11 | 16 | | |
| 11 | 5 | 10 | 12 | 17 | |
| 12 | 6 | 11 | 13 | 18 | |
| 13 | 7 | 8 | 12 | 14 | 19 |
| 14 | 8 | 13 | 15 | 20 | |
| 15 | 9 | 14 | 21 | | |
| 16 | 10 | 17 | 22 | 23 | |
| 17 | 11 | 16 | 18 | 23 | |
| 18 | 12 | 17 | 19 | 24 | |
| 19 | 13 | 18 | 20 | 25 | 26 |
| 20 | 14 | 19 | 21 | 26 | |
| 21 | 15 | 20 | 27 | | |
| 22 | 16 | 23 | | | |
| 23 | 16 | 17 | 22 | 24 | |
| 24 | 18 | 23 | 25 | | |
| 25 | 19 | 24 | 26 | | |
| 26 | 19 | 20 | 25 | 27 | |
| 27 | 3 | 21 | 26 | | |

TABELA 3.2: Lista de nós adjacentes do SDE da Figura 3.4

A RNP para a configuração inicial, $I_0$, é mostrada na Equação 3.3. As duas primeiras linhas na equação indicam, respectivamente, as profundidades e os nós da árvore 1, as duas segundas linhas as da árvore 2 e as duas últimas as da árvore 3.

$$I_0 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 3 & 2 & 3 & & & \\ 2 & 9 & 8 & 7 & 13 & 15 & 14 & & & \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{array}\right] \tag{3.3}$$

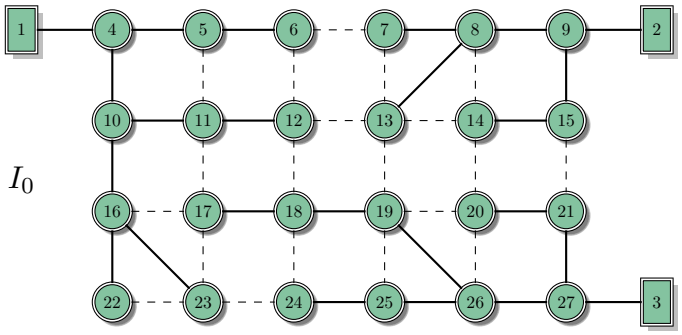
No caso do problema de restauração de SDE, a população inicial é gerada a partir da aplicação dos operadores PAO e/ou CAO da RNP em $I_0$ gerando, assim, novos indivíduos. Por simplicidade, somente o PAO será utilizado neste exemplo ilustrativo.

O primeiro indivíduo da população, $I_1$, é gerado escolhendo-se aleatoriamente uma das três árvores de $I_0$ para a aplicação do PAO. A partir desta, escolhe-se aleatoriamente um nó para *poda*. Escolhido o nó de *poda*, e com a ajuda da Lista de Adjacências de Nós, um nó adjacente ao nó de *poda* é escolhido nesta lista, também aleatoriamente, para ser o nó de *enxerto*. Desta forma, o primeiro indivíduo da população ($I_1$) é facilmente determinado pela tripla dada por:

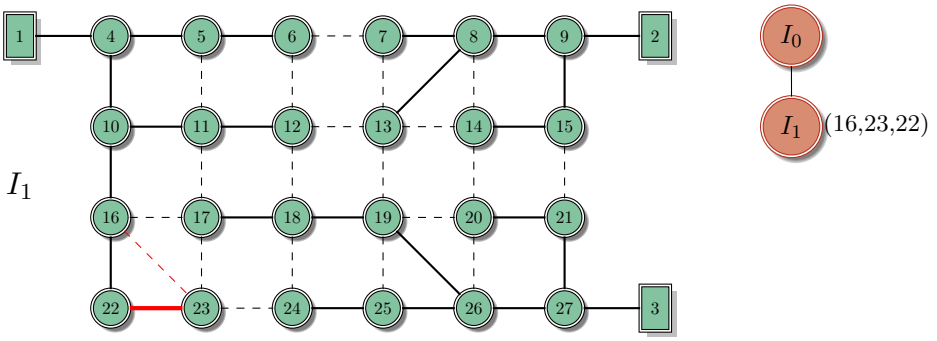
$$(nó\_de\_origem,\ nó\_de\_poda,\ nó\_de\_enxerto) \tag{3.4}$$

Esta tripla indica que o indivíduo $I_1$ veio do indivíduo $I_0$ através da poda do ***nó_de_poda*** do ***nó_de_origem*** e enxertando a subárvore gerada no ***nó_de_enxerto***. A poda de um nó e seu enxerto em outro nó, para o SDE, significa desconectar um dado setor do sistema e conectá-lo em outro setor gerando uma nova configuração (novo indivíduo). A Figura 3.5(b) mostra o caso em que o indivíduo $I_1$ foi hipoteticamente determinado através da tripla $(16, 23, 22)$ e a correspondente Árvore de Ancestralidade gerada. O grafo nesta figura mostra que a árvore de $I_0$ escolhida para a aplicação do PAO foi a árvore 1, uma vez que o ***nó_de_poda*** é o nó 23, pertencente a esta árvore. O nó 23 é, portando, podado do ***nó_de_origem***: o nó 16. Com o auxílio da lista de adjacências de nós, observa-se que o nó 23 só pode ser conectado aos nós 16, 17, 22 ou 24. Um destes nós é aleatoriamente escolhido para ser o ***nó_de_enxerto***, no exemplo o nó de enxerto escolhido foi o nó 22.

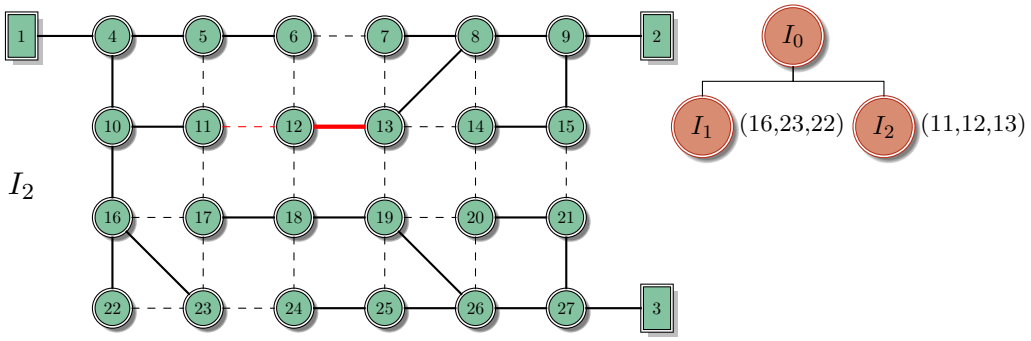
A seguir, um ancestral para a geração do segundo indivíduo da população deve ser escolhido aleatoriamente entre os indivíduos $I_0$ e $I_1$. Uma vez escolhido seu ancestral, o segundo indivíduo da população é criado aplicando o PAO da mesma maneira como aplicado na criação do indivíduo $I_1$. A Figura 3.5(c) ilustra o caso em que o indivíduo $I_2$ foi gerado novamente a partir do indivíduo $I_0$ e a Árvore de Ancestralidade atualizada com a tripla $(11, 12, 13)$ utilizada para a sua criação.




(a) Indivíduo inicial $I_0$: Configuração corrente




(b) Indivíduo $I_1$ obtido a partir da aplicação do PAO em $I_0$ e a Árvore de Ancestralidade resultante.




(c) Indivíduo $I_2$ obtido a partir da aplicação do PAO em $I_0$ e a Árvore de Ancestralidade resultante.

FIGURA 3.5: Geração dos indivíduos $I_1$ e $I_2$ da população inicial.

As equações 3.5 e 3.6 mostram, respectivamente, as RNPs dos indivíduos $I_1$ e $I_2$ gerados destacando em texto negrito-vermelho o nó podado e sua nova profundidade.

$$I_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & \mathbf{5} \\ 1 & 4 & 5 & 6 & 10 & 11 & 12 & 16 & 22 & \mathbf{23} \\ \\ 0 & 1 & 2 & 3 & 3 & 2 & 3 & & & \\ 2 & 9 & 8 & 7 & 13 & 15 & 14 & & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix} \tag{3.5}$$

$$I_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 3 & 4 & 4 & \\ 1 & 4 & 5 & 6 & 10 & 11 & 16 & 22 & 23 & \\ \\ 0 & 1 & 2 & 3 & 3 & \mathbf{4} & 2 & 3 & & \\ 2 & 9 & 8 & 7 & 13 & \mathbf{12} & 15 & 14 & & \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 5 & 3 & 4 \\ 3 & 27 & 21 & 20 & 26 & 19 & 18 & 17 & 25 & 24 \end{bmatrix} \tag{3.6}$$

Da mesma forma, o terceiro indivíduo da população é gerado através da escolha randômica do seu ancestral entre os três indivíduos existentes $I_0$, $I_1$ e $I_2$. Escolhido o seu ancestral, é aplicado o PAO, anotado a sua tripla e, então, a Árvore de Ancestralidade é atualizada. Este procedimento é aplicado até que todos os indivíduos da população inicial sejam determinados. A Figura 3.6 mostra a Árvore de Ancestralidade gerada para uma população inicial de 20 indivíduos que será utilizada neste exemplo ilustrativo.

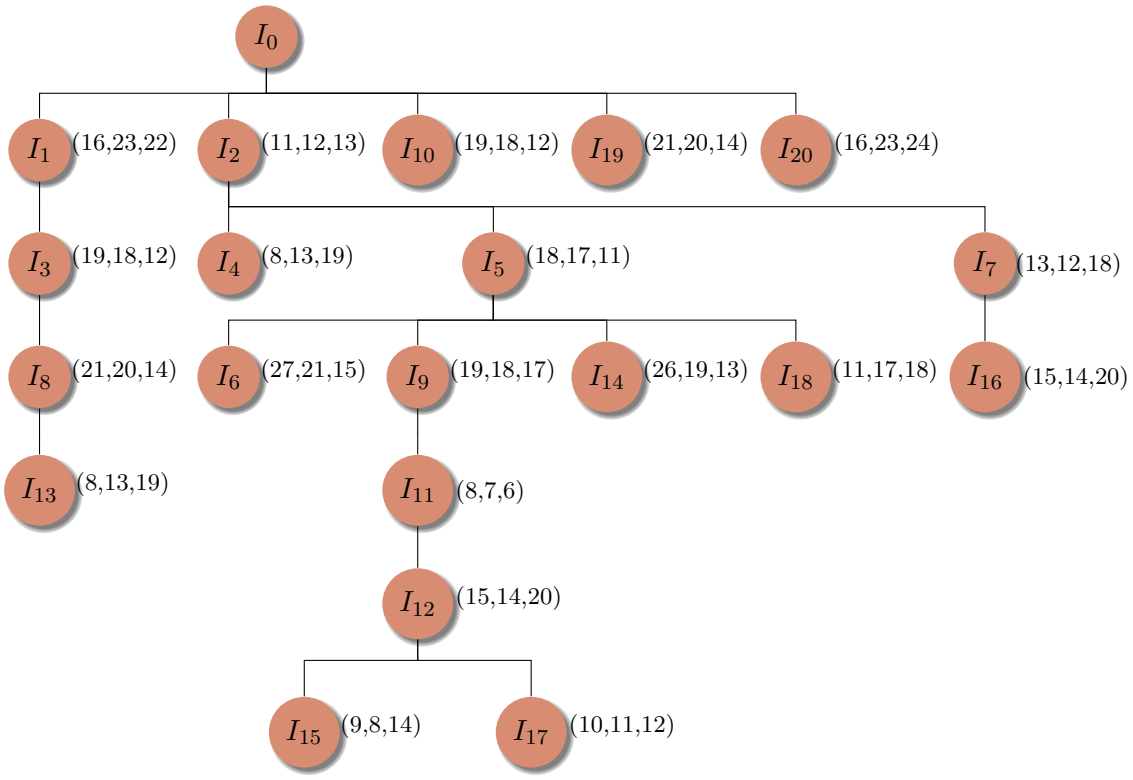



FIGURA 3.6: Árvore de Ancestralidade da População Inicial

### 3.5.2 Mutação Diferencial

Após a criação da população inicial, a equação da mutação diferencial discreta (Eq. 2.9) é utilizada para evoluir os indivíduos da população. Para cada indivíduo $I_i$ um vetor mutante $V_i$ é gerado.

Suponha que $I_6$, $I_{15}$ e $I_7$, aleatoriamente escolhidos dentro da população inicial anteriormente gerada, façam parte da equação da mutação diferencial, respectivamente os vetores $X_{base}$, $X_{g,r_1}$ e $X_{g,r_2}$. O vetor mutante correspondente é reescrito como:

$$V_i = I_6 \oplus \mathbf{F} \otimes (I_{15} \ominus I_7) \tag{3.7}$$

Agora, através deste exemplo, os operadores subtração, soma e multiplicação por escalar da equação da mutação diferencial são definidos para o problema de restauração do SDE.

### Diferença Vetorial

A diferença vetorial entre os dois indivíduos na equação da mutação diferencial é definida para o problema de restauração do SDE como um conjunto de movimentos elementares obtidos da árvore de ancestralidade de tal modo que, partindo-se do indivíduo $I_{15}$, chega-se ao indivíduo $I_7$. Este conjunto de movimentos gera uma Lista de Movimentos $\mathcal{M}_{1,2}$ contendo dois tipos de "caminhos" entre estes dois indivíduos: Um denominado de *Backtracking Path* [6] formado por todos os indivíduos entre o indivíduo $I_{15}$ e o ancestral comum entre ele e o indivíduo $I_7$ (no exemplo o indivíduo $I_2$), e o outro denominado de *Forward Path* [7] formado por todos os indivíduos entre o ancestral comum e o indivíduo $I_7$. Desta forma, a diferença vetorial na equação da mutação diferencial é escrita para estes dois indivíduos como:

$$\begin{aligned}\mathcal{M}_{1,2} &= (I_{15} \ominus I_7) \\ &= \big[I_{15} \leftarrow F_{12} \leftarrow I_{11} \leftarrow I_9 \leftarrow I_5 \leftarrow \boldsymbol{I_2} \rightarrow I_7\big],\end{aligned} \tag{3.8}$$

onde:

$\boldsymbol{I_2}$ é o ancestral comum entre os indivíduos $I_{15}$ e $I_7$ (ver Fig. 3.6).

$I_{15} \leftarrow I_{12} \leftarrow I_{11} \leftarrow I_9 \leftarrow I_5 \leftarrow I_2$: são os movimentos de *Backtracking Path*

$I_2 \rightarrow I_7$: é o movimento de *Forward Path*.

Observe que esta estrutura de dados definida para a lista de movimentos é diferente daquela estrutura genérica apresentada na Seção 2.8.3, sendo que aqui ela foi definida especificamente para ser utilizada juntamente com a árvore de ancestralidade e aplicada ao problema de restauração de SDE.

### Multiplicação por escalar

A multiplicação da lista de movimentos pelo escalar $\mathbf{F} \in [0, 1]$, segundo a Definição 2.4, deve retornar os primeiros $\mathbf{F\%}$ movimentos da lista de movimentos. As equações 3.9 à 3.11 mostram as listas de movimentos ponderadas, $\mathcal{M}'_{1,2}$, para os casos em que o fator

[6] em português: caminho de retorno, é o caminho de retrocesso na árvore de ancestralidade que deve ser percorrido de modo que, partindo do indivíduo $I_{15}$ chega-se a um ancestral comum entre os indivíduos $I_{15}$ e $I_7$.

[7] em português: caminho para frente, é o caminho para adiante na árvore de ancestralidade que, partindo do ancestral comum, chega-se ao indivíduo $I_7$

de multiplicação são respectivamente $\mathbf{F} = 0,6$; $\mathbf{F} = 0,8$ e $\mathbf{F} = 1,0$.

$$\mathcal{M}'_{1,2} = \mathbf{F} \otimes \mathcal{M}_{1,2} = 0,6 \times 7 = \left[I_{15},\, I_{12},\, I_{11},\, I_9\right] \tag{3.9}$$

(apenas os 4 primeiros movimentos da lista são considerados.)

$$\mathcal{M}'_{1,2} = \mathbf{F} \otimes \mathcal{M}_{1,2} = 0,8 \times 7 = \left[I_{15},\, I_{12},\, I_{11},\, I_9,\, I_5,\, \boldsymbol{I}_2\right] \tag{3.10}$$

(apenas os 6 primeiros movimentos da lista são considerados.)

$$\mathcal{M}'_{1,2} = \mathbf{F} \otimes \mathcal{M}_{1,2} = 1,0 \times 7 = \left[I_{15},\, I_{12},\, I_{11},\, I_9,\, I_5,\, \boldsymbol{I}_2,\, I_7\right] \tag{3.11}$$

(todos os movimentos da lista são considerados.)

### Operador Soma: Vetor Mutante

Com a operação de adição, isto é, a aplicação da lista de movimentos ponderada $\mathcal{M}'_{1,2}$ ao vetor base conforme a Definição 2.5, determina-se finalmente o vetor mutante. As estratégias possíveis de como aplicar estes movimentos ao vetor base são definidas de acordo com as características do SDE. Duas estratégias possíveis são apresentadas a seguir. Para tanto, a Equação 3.12 apresenta a RNP do vetor $X_{base}$ (indivíduo $I_6$) ao qual serão aplicados os movimentos da lista de movimentos.

$$I_6 = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 17 & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 3 & 4 & 2 & 3 & 4 & 3 \\ 2 & 9 & 8 & 7 & 13 & 12 & 15 & 21 & 20 & 14 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 3 & 4 & & & \\ 3 & 27 & 26 & 19 & 18 & 25 & 24 & & & \end{array}\right] \tag{3.12}$$

**Estratégia Nº.1:** Uma definição possível de como aplicar os movimentos da lista ao vetor base seria, primeiro, substituir cada par de indivíduos da lista de movimentos pela tripla da árvore de ancestralidade que relaciona estes dois indivíduos e, em seguida, as informações contidas nas triplas seriam aplicadas ao vetor base de acordo com o tipo de percurso indicado na lista de movimentos: nos percursos de *Backtracking Path* voltar o nó de poda ao nó de origem e nos percursos de *forward Path* enxertar o nó de poda ao nó de enxerto.

Para exemplificar esta estratégia, o fator escalar **F** é considerado igual a 1 de modo que todos os movimentos da lista possam ser aplicados ao vetor base. O primeiro movimento da lista a ser aplicado ao vetor base é o movimento de *backtracking path* ($I_{15} \leftarrow I_{12}$). Na árvore de ancestralidade, a tripla que relaciona estes dois indivíduos é a tripla $(9, 8, 14)$ que indica que o indivíduo $I_{15}$ originou do indivíduo $I_{12}$ através da poda do nó 8 do nó 9 e do seu enxerto ao nó 14. Como este percurso é um percurso de *Backtrtacking Path* ele é aplicado ao vetor base como um caminho de retrocesso, ou seja, o PAO deve ser aplicado ao vetor base de maneira a retornar o nó 8 ao nó 9. Entretanto, o nó 8 no indivíduo $I_6$ (árvore 2) já está conectado ao nó 9. Neste caso, este movimento da lista é negligenciado e passa-se para o próximo movimento. O mesmo ocorre com os três movimentos seguintes ($I_{12} \leftarrow I_{11}$), ($I_{11} \leftarrow I_9$) e ($I_9 \leftarrow I_5$). Todos eles já existem em $I_6$ e são descartados.

O próximo movimento da lista é o movimento de *Backtracking Path* ($I_5 \leftarrow I_2$) cuja tripla correspondente é $(18, 17, 11)$. Esta tripla indica que o indivíduo 5 originou-se do indivíduo 2 através da poda do nó 17 do nó 18 e do seu enxerto no nó 11. A aplicação deste movimento de *backtracking* ao vetor base é interpretado como o de desconectar o nó 17 de onde ele estiver conectado em $I_6$ (no caso o nó 11) e conectá-lo ao nó 18 (árvore 3 em $I_6$). Este movimento é um movimento válido e, portanto, as alterações são realizadas no indivíduo $I_6$, através do PAO, podando o nó 17 do nó 11 e enxertando-o ao nó 18 como destacado na Equação 3.13.

$$I_6 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & \_ & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & \_ & 16 & 22 & 23 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 3 & 4 & 2 & 3 & 4 & 3 \\ 2 & 9 & 8 & 7 & 13 & 12 & 15 & 21 & 20 & 14 \\ & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & \mathbf{5} & 3 & 4 & & \\ 3 & 27 & 26 & 19 & 18 & \mathbf{17} & 25 & 24 & & \end{bmatrix} \tag{3.13}$$

Finalmente, o último movimento da lista a ser aplicado ao vetor base é o movimento de *Forward Path* ($I_2 \rightarrow I_7$). A tripla na árvore de ancestralidade correspondente a este movimento é a tripla $(13, 12, 18)$. Ela indica que, no vetor base, o nó 12 deve ser podado de onde ele estiver conectado, neste exemplo coincidentemente o nó 13, e ser enxertado ao nó 18. Este é um movimento válido em $I_6$ e, portanto, é aplicado. O vetor multante resultante da aplicação destes movimentos da lista ao vetor base é mostrado

na Equação 3.14.

$$V_{mutante} = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 16 & 22 & 23 \\ & & & & & & & & \\ 0 & 1 & 2 & 3 & 3 & 2 & 3 & 4 & 3 \\ 2 & 9 & 8 & 7 & 13 & 15 & 21 & 20 & 14 \\ & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 5 & 3 & 4 \\ 3 & 27 & 26 & 19 & 18 & 12 & 17 & 25 & 24 \end{bmatrix} \tag{3.14}$$

O vetor mutante determinado desta maneira pode ser interpretado, assim como aquele determinado no domínio contínuo, como a "distância vetorial" entre os indivíduos $I_{15}$ e $I_7$ aplicada ao indivíduo $I_6$.

Após a fase de cruzamento, o vetor mutante é acrescentado temporariamente à árvore de ancestralidade para, posteriormente na fase de seleção, competir pela sobrevivência na próxima geração com o seu vetor alvo correspondente. Neste caso, supondo que o indivíduo $I_3$ seja o vetor alvo desta mutação e que, na disputa pela sobrevivência, ele perca para o vetor mutante, ele é retirado da árvore de ancestralidade e seus movimentos são adicionados aos do seu genitor, indivíduo $I_8$, e o vetor mutante passa a ser o indivíduo $I_3$. A Figura 3.7 ilustra a árvore de ancestralidade neste caso através da adição do vetor mutante ao seu ancestral (o vetor base $I_6$) e todas as triplas em sequência utilizadas para determiná-lo.

Nesta estratégia observa-se uma quantidade muito grande de movimentos de *Backtracking Path* cancelados. Isto se deve ao fato de que, em profundidade na árvore de ancestralidade, um indivíduo difere do seu antecessor apenas dos movimentos indicados naquele ramo da árvore. Estes movimentos de *backtracking* aplicados em indivíduos imediatamente ao lado de um indivíduo com a mesma profundidade não provocam mudanças nos mesmos. Assim, com poucos movimentos sendo aplicados ao vetor base, o vetor mutante resultante sofre pouca mutação. Com isto a diversidade da população torna-se menor quanto maior for a profundidade da árvore de ancestralidade. Associado ao fato de que no DE o passo da mutação varia de acordo com a distribuição da população[8], faz com que a população convirja rapidamente.

**Estratégia Nº.2:** Uma maneira de aumentar a diversidade de indivíduos, sem ter que aumentar demasiadamente a população, seria aplicar o PAO ao vetor base como quando

[8]Indivíduos muito distintos geram uma mutação maior enquanto que indivíduos muito parecidos geram uma mutação menor

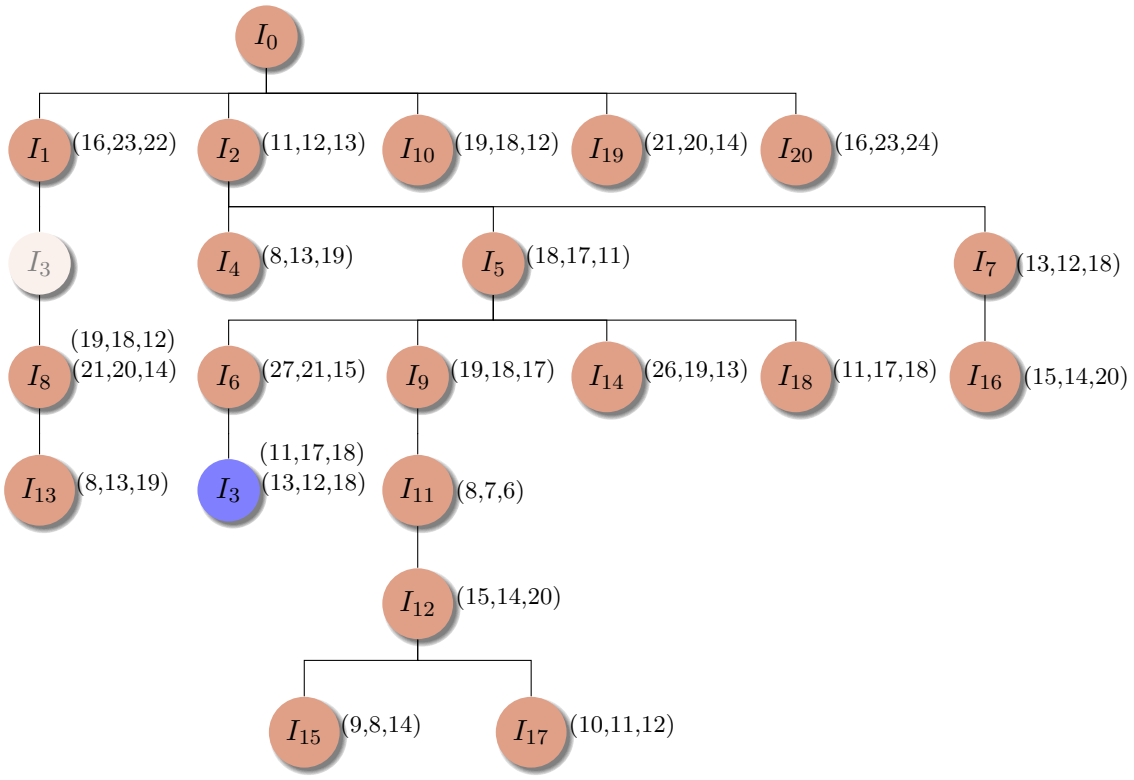



FIGURA 3.7: Adição do vetor mutante à árvore de ancestralidade.

da criação da população inicial, exceto que, neste caso, o indivíduo a ser escolhido na população para a mutação, a árvore para poda e o nó de poda são determinados através da lista de movimentos. Quando da criação da população inicial todos estes passos eram determinados aleatoriamente. A estratégia utilizada para a mutação era definida pelos os seguintes passos:

- Escolher aleatoriamente um indivíduo da população para sofrer a mutação.
- Escolher aleatoriamente uma árvore deste indivíduo.
- Desta árvore, escolher um nó para ser o nó de poda.
- Com o auxílio da lista de adjacências, escolher um nó adjacente ao nó de poda para ser o nó de enxerto.
- Determinados os nós de poda e de enxerto, aplicar um dos operadores da RNP ao indivíduo escolhido.

Para a mutação diferencial, para cada indivíduo da lista de movimentos, a sequência a ser seguida nesta estratégia é:

- Identificar na tripla do indivíduo o nó de poda.
- Identificar no vetor base a árvore em que este nó está conectado.
- Desta árvore, escolher um novo nó para ser o nó de poda.

- Com o auxílio da lista de adjacências, escolher um nó adjacente ao novo nó de poda para ser o nó de enxerto.
- Determinado os nós de poda e de enxerto aplicar um dos operadores da RNP ao vetor base.

Para exemplificar esta estratégia, suponha que a lista de movimentos ponderada a ser aplicada ao vetor base seja a lista dada pela Equação 3.9. O primeiro movimento desta lista é o movimento $I_{15}$. Este indivíduo foi determinado pela tripla $(9, 8, 14)$. O nó de poda desta tripla é o nó 8. Este nó no vetor base está na árvore 2. Portanto, da árvore 2 do vetor base é escolhido aleatoriamente um novo nó de poda, por exemplo o nó 13. Esta poda gera uma subárvore formada pelos nós 13 e 12, respectivamente a raiz e um nó folha. Através da lista de adjacências é escolhido um nó adjacente ao nó 13 para ser o nó de enxerto desta subárvore, por exemplo o nó 19. Assim, escolhidos os nós de poda e de enxerto, o PAO é aplicado ao vetor base para a realização da mutação. A RNP do vetor mutante resultante da aplicação deste primeiro movimento da lista ao vetor base é mostrada na Equação 3.15.

$$
\begin{aligned}
I_6 &= \begin{bmatrix}
0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\
1 & 4 & 5 & 6 & 10 & 11 & 17 & 16 & 22 & 23 \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 3 & 4 & 2 & 3 & 4 & 3 \\
2 & 9 & 8 & 7 & 13 & 12 & 15 & 21 & 20 & 14 \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 3 & 4 & & & \\
3 & 27 & 26 & 19 & 18 & 25 & 24 & & &
\end{bmatrix} \\
&= \begin{bmatrix}
0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\
1 & 4 & 5 & 6 & 10 & 11 & 17 & 16 & 22 & 23 \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & & \\
2 & 9 & 8 & 7 & 15 & 21 & 20 & 14 & & \\
 & & & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & \\
3 & 27 & 26 & 19 & 13 & 12 & 18 & 25 & 24 &
\end{bmatrix}
\end{aligned}
\tag{3.15}
$$

O próximo movimento da lista a ser aplicado é o movimento $I_{12}$. A tripla correspondente a este indivíduo, indicada na árvore de ancestralidade, é a tripla $(15, 14, 20)$ que indica como nó de poda o nó 14. Este nó encontra-se na árvore 2 do vetor base. Da árvore 2, um nó é escolhido para ser o novo nó de poda, por exemplo o nó 7. Os nós adjacentes ao nó 7 na lista de adjacências são os nós 6, 8 e 13. Escolhendo-se o nó 6 como o nó de enxerto e aplicando o PAO, o vetor base é atualizado para o indicado na Equação 3.16.

$$I_6 = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 10 & 11 & 17 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & & \\ 2 & 9 & 8 & 7 & 15 & 21 & 20 & 14 & & \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & \\ 3 & 27 & 26 & 19 & 13 & 12 & 18 & 25 & 24 & \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 7 & 10 & 11 & 17 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 2 & 3 & 4 & 3 & & & & \\ 2 & 9 & 8 & 15 & 21 & 20 & 14 & & & & \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & & \\ 3 & 27 & 26 & 19 & 13 & 12 & 18 & 25 & 24 & & \end{bmatrix} \tag{3.16}$$

O terceiro movimento da lista a ser aplicado ao vetor base é o indivíduo $I_{11}$ cuja tripla na árvore de ancestralidade é a tripla $(8, 7, 6)$. O nó de poda desta tripla, o nó 7, no vetor base encontra-se, agora, na árvore 1. Portanto, da árvore 1 é escolhido o novo nó de poda, por exemplo o nó 6 que gera uma subárvore formada pelos nós 6 e 7. Os nós adjacentes ao nó 6 na lista de adjacências são os nós 5, 7 e 12. Destes, escolhe-se um novo nó de enxerto, por exemplo o nó 12. Observe que se o nó 7 fosse o escolhido, ele deveria ser descartado pois isso geraria um ciclo desconectado das três árvores do vetor base. A RNP do vetor base após a aplicação do PAO é atualizada como mostrada na Equação 3.17 abaixo.

$$I_6 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 3 & 4 & 4 \\ 1 & 4 & 5 & 6 & 7 & 10 & 11 & 17 & 16 & 22 & 23 \\ \\ 0 & 1 & 2 & 2 & 3 & 4 & 3 & & & & \\ 2 & 9 & 8 & 15 & 21 & 20 & 14 & & & & \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & & \\ 3 & 27 & 26 & 19 & 13 & 12 & 18 & 25 & 24 & & \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & & \\ 1 & 4 & 5 & 10 & 11 & 17 & 16 & 22 & 23 & & \\ \\ 0 & 1 & 2 & 2 & 3 & 4 & 3 & & & & \\ 2 & 9 & 8 & 15 & 21 & 20 & 14 & & & & \\ \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 4 & 3 & 4 \\ 3 & 27 & 26 & 19 & 13 & 12 & 6 & 7 & 18 & 25 & 24 \end{bmatrix} \tag{3.17}$$

O último movimento da lista a ser aplicado é o movimento $I_9$ cuja tripla é $(19, 18, 17)$ e o nó de poda é o nó 18 que, no vetor base, encontra-se em $T_3$. De $T_3$, portanto, deve ser escolhido o novo nó de poda, por exemplo o nó 13. Os nós adjacentes ao nó 13 são os nós 7, 8, 12, 14 e 19. Escolhendo-se o nó 14 como nó de enxerto e aplicando o PAO, finalmente, obtém-se o vetor mutante dado pela RNP mostrada na Equação 3.18.

$$I_6 = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & & \\ 1 & 4 & 5 & 10 & 11 & 17 & 16 & 22 & 23 & & \\ & & & & & & & & & & \\ 0 & 1 & 2 & 2 & 3 & 4 & 3 & & & & \\ 2 & 9 & 8 & 15 & 21 & 20 & 14 & & & & \\ & & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 4 & 3 & 4 \\ 3 & 27 & 26 & 19 & 13 & 12 & 6 & 7 & 18 & 25 & 24 \end{bmatrix}$$

$$I_{mutante} = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & & \\ 1 & 4 & 5 & 10 & 11 & 17 & 16 & 22 & 23 & & \\ & & & & & & & & & & \\ 0 & 1 & 2 & 2 & 3 & 4 & 3 & 4 & 5 & 6 & 7 \\ 2 & 9 & 8 & 15 & 21 & 20 & 14 & 13 & 12 & 6 & 7 \\ & & & & & & & & & & \\ 0 & 1 & 2 & 3 & 4 & 3 & 4 & & & & \\ 3 & 27 & 26 & 19 & 18 & 25 & 24 & & & & \end{bmatrix} \tag{3.18}$$

A árvore de ancestralidade, atualizada com todas as triplas utilizadas para determinar o vetor mutante a partir do vetor base, obtida com esta estratégia é mostrada na Figura 3.8.

Nesta estratégia observa-se que todos os movimentos da lista de movimentos ponderada foram utilizados e que o grau de liberdade obtido para se ter uma diversidade de indivíduos é muito maior devido a aleatoriedade em que os nós de poda e enxerto são determinados no vetor base. Isto evita a necessidade de se aumentar demasiadamente a população inicial de modo a ter uma diversidade maior de indivíduos.

Na estratégia anterior, os nós de poda e enxerto utilizados na mutação eram somente aqueles que estivessem presentes nas triplas da árvore de ancestralidade, o que restringia a diversidade de indivíduos e, consequentemente, promovia a convergência prematura da população.

### 3.5.3 Cruzamento

Após a etapa de mutação diferencial, o algoritmo DE realiza a etapa de cruzamento ou recombinação. Diferentemente dos demais AG, em que o cruzamento é realizado entre dois indivíduos pais escolhidos na população para gerar um indivíduo filho, a DE

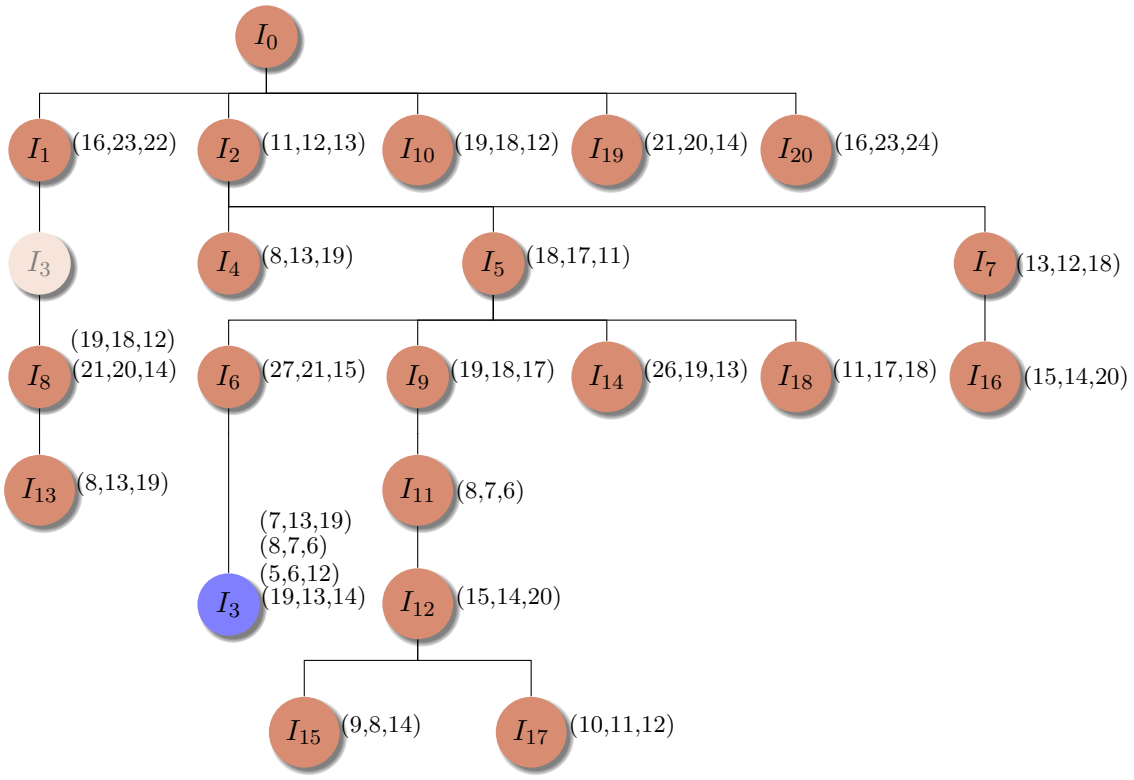



FIGURA 3.8: Adição do vetor mutante à árvore de ancestralidade - Estratégia 2.

realiza uma recombinação entre o vetor alvo (o vetor corrente) e o seu vetor mutante correspondente para gerar um único filho: o vetor experimental.

Porém, em problemas combinatórios onde a estrutura de dados utilizada para representar os indivíduos são estruturas em árvore, realizar um cruzamento entre dois indivíduos, sem gerar indivíduos infactíveis, não é nada trivial. Nestes casos, rotinas de reparos são necessárias de modo a garantir que o indivíduo filho, resultado do cruzamento, seja também uma estrutura em árvore, ou seja, um grafo conexo sem ciclos. Uma alternativa para cruzar dois indivíduos cuja estrutura de dados é um grafo do tipo árvore é utilizar a equação da evolução diferencial novamente para realizar o cruzamento entre o vetor alvo e o vetor mutante. Com este procedimento, o vetor experimental pode ser, então, determinado de duas maneiras: aplicando os movimentos da lista de movimentos obtida entre estes dois vetores ao vetor alvo ou aplicando-os ao vetor mutante, conforme mostrado na Equação 3.19.

$$\begin{aligned} I_{experimental} &= I_{alvo} \oplus \mathcal{C}_r \otimes (I_{mutante} \ominus I_{alvo}) \\ &ou \\ I_{experimental} &= I_{mutante} \oplus \mathcal{C}_r \otimes (I_{mutante} \ominus I_{alvo}), \end{aligned} \tag{3.19}$$

em que o escalar **F** da equação da mutação diferencial é substituído pelo escalar probabilidade de cruzamento $\mathcal{C}_r$.

Para exemplificar, e supondo que o vetor alvo seja o indivíduo $I_{13}$ e o vetor mutante seja aquele determinado na seção anterior, a lista de movimentos entre o vetor alvo e o vetor mutante é escrita como:

$$\begin{aligned}\mathcal{M}_{1,2} &= (I_M \ominus I_{13}) \\ &= \left[I_M \leftarrow I_6 \leftarrow I_5 \leftarrow I_2 \leftarrow \boldsymbol{I_0} \rightarrow I_1 \rightarrow I_3 \rightarrow I_8 \rightarrow I_{13}\right].\end{aligned} \tag{3.20}$$

Para um fator de cruzamento $\mathcal{C}_r = 0,5$, o vetor experimental é determinado aplicando-se 50% dos movimentos da lista de movimentos em $I_{13}$, como mostrado na Equação 3.21.

$$\begin{aligned}I_{experimental} &= I_{alvo} \oplus \mathcal{C}_r \otimes (I_M \ominus I_{13}) \\ &= I_{13} \oplus 0,5 \otimes \left[I_M \leftarrow I_6 \leftarrow I_5 \leftarrow I_2 \leftarrow \boldsymbol{I_0} \rightarrow I_1 \rightarrow I_3 \rightarrow I_8 \rightarrow I_{13}\right] \\ &= I_{13} \oplus \left[I_M \leftarrow I_6 \leftarrow I_5 \leftarrow I_2 \leftarrow \boldsymbol{I_0}\right].\end{aligned} \tag{3.21}$$

Após ser determinado por uma das duas estratégias mostradas acima, o vetor experimental vai para a fase de seleção juntamente com o seu vetor alvo correspondente disputar um lugar na próxima geração. Caso vença a disputa, na próxima geração ele passa a ser o indivíduo $I_{13}$, a árvore de ancestralidade é atualizada e a tripla do vetor alvo é acrescentada às do vetor experimental como mostrado na Figura 3.9.

### 3.5.4 Busca Local

Algoritmos de busca local são algoritmos de refinamento que realizam uma busca a partir de uma solução, obtida de uma heurística construtiva, e uma dada estrutura de vizinhança previamente definida com o intuito de melhorar a solução encontrada. A estrutura de vizinhança definida para o problema deve ser composta por um conjunto de soluções com características "bem próximas" às da solução corrente e que sejam capazes de acrescentar alguma melhora na solução. Assim, a cada iteração, o algoritmo percorre a estrutura de vizinhança em busca de um vizinho com uma solução que seja melhor que a da solução corrente. Se uma solução melhor for encontrada esta substitui a solução corrente. O algoritmo efetua trocas entre a solução corrente e outra pertencente a estrutura de vizinhança, em uma busca exaustiva até que nenhum melhoramento seja mais possível, caindo-se, assim, em um ótimo local.

No algoritmo DE-Tree (a heurística construtiva aqui utilizada) para realizar uma busca local na solução corrente (o vetor experimental oriundo da mutação diferencial) é

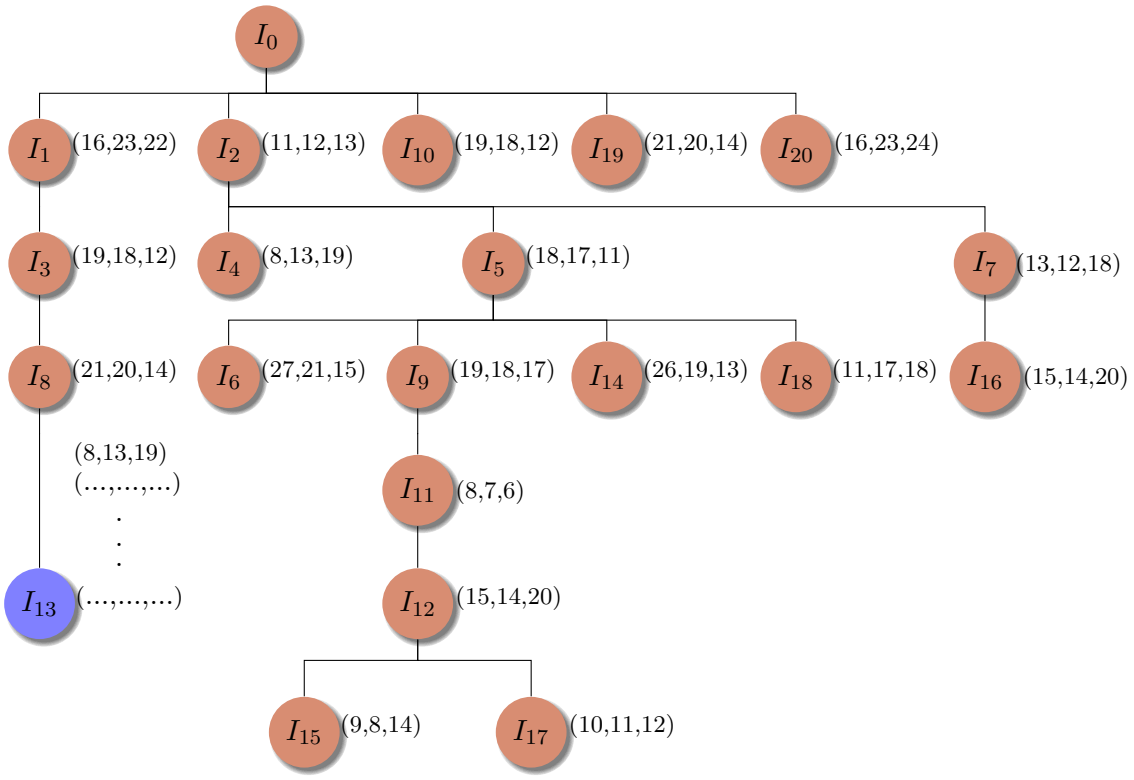



FIGURA 3.9: Atualização da árvore de ancestralidade com o vetor experimental.

necessário primeiramente definir uma estrutura de vizinhança que se adeque à estrutura de dados utilizada para representar as soluções do problema de restauração do SDE.

Sejam, por exemplo, os movimentos que originaram o vetor experimental ($I_{13}$) aqueles em vermelho mostrados na árvore de ancestralidade da Figura 3.10. Uma estrutura de vizinhança que poderia ser definida para esta solução seria obtida primeiro, com o auxílio da Lista de Adjacências de chaves dada pela Tabela 3.2, listar todos os nós adjacentes aos *nós* presentes na lista de movimentos do vetor experimental (nós 13, 7, 6 *e* 23) e, em seguida, testar cada um deles como um possível novo nó de enxerto. Assim, o primeiro movimento da lista tem como *nó* o setor 13. Este nó, segundo a Lista de Adjacências, só pode ser conectado aos nós:

$$13 \rightarrow 7 - 8 - 12 - 14 - 19.$$

Com exceção do nó 19, ao qual o nó 13 já está conectado, todos os outros nós são testados sequencialmente segundo a lista de adjacências. Assim, utilizando os Operadores definidos na RNP, o nó 13 é enxertado em cada um dos nós desta lista e, em havendo uma melhora, o vetor experimental passa a ser este vizinho e a árvore de ancestralidade é atualizada com este movimento. Este procedimento é realizado para todos os nós da lista de movimentos do vetor experimental.

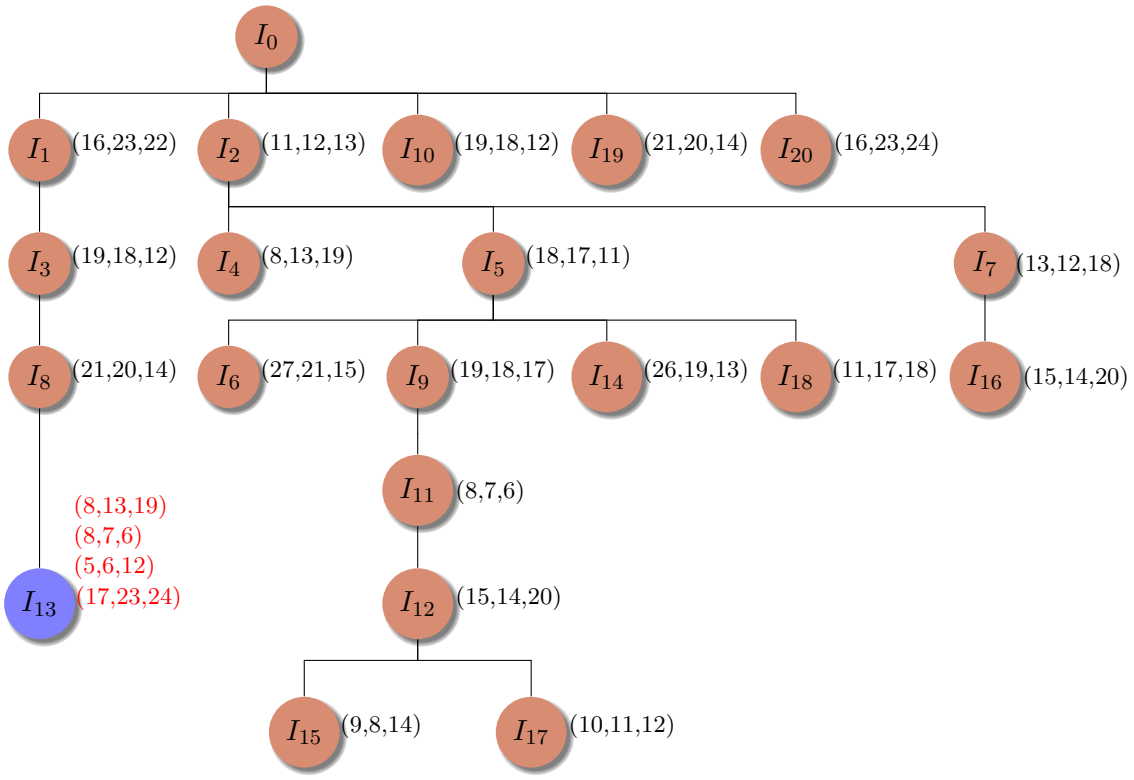



FIGURA 3.10: Atualização da árvore de ancestralidade com o vetor experimental.

### 3.5.5 Árvore de Ancestralidade

A árvore de ancestralidade utilizada pelo algoritmo DE-Tree proposto neste trabalho tem duas importantes finalidades: A primeira é a de armazenar as informações relativas entre os diversos indivíduos, tais como, o seu ancestral e as triplas *(poda, nó , enxerto)* contidas nos diversos movimentos utilizados na criação da população. Estas informações são importantes para a montagem da lista de movimentos utilizada na equação da mutação diferencial, pois elas indicam o "caminho" que deve ser percorrido entre cada dois indivíduos da população de modo a se obter a "diferença vetorial" entre eles. A segunda, importante principalmente para o problema de contingências no SDE devido a faltas, é a de ter armazenado nas triplas dos movimentos de cada indivíduo as chaves que devem ser abertas/fechadas para a nova configuração do sistema. Assim, após obter o indivíduo com o menor custo, a aplicação de todos os movimentos relativos entre este indivíduo e o indivíduo inicial é possível obter a nova configuração para o sistema.

Seja, por exemplo, a árvore mostrada na Figura 3.11 a árvore de ancestralidade obtida para a população final após a execução do algoritmo. Nela são mostrados os custos da função objetivo que cada indivíduo obteve e, em vermelho, é destacado o custo do melhor indivíduo, o indivíduo 16 com custo igual à $491,44$. Para se reconfigurar o

SDE com a configuração ótima obtida por este indivíduo basta, partindo-se da configuração corrente (indivíduo $I_0$), percorrer a árvore de ancestralidade até o indivíduo $I_{16}$ ser alcançado, isto é, percorrer o caminho $I_0 \rightarrow I_{15} \rightarrow I_{16}$ aplicando-se todos os movimentos contidos neste caminho: os 14 movimentos do indivíduo $I_{15}$ e os 2 movimentos do indivíduo $I_{16}$.

É possível observar nesta figura que quando a população converge para um determinado valor, a maioria dos movimentos contidos em cada indivíduo são os mesmos diferindo apenas nos últimos movimentos. Além disso, neste exemplo, observa-se também que os indivíduos $I_8$ e $I_{16}$ são os mesmos pois eles têm o custo e todos os movimentos idênticos. Deste ponto em diante uma busca local pode ser aplicada para o refinamento do resultado.

O número de manobras necessárias para reconfigurar o sistema pode ser também obtido da árvore de ancestralidade bastando para tanto somar todos os movimentos contidos entre o indivíduo corrente e o indivíduo ótimo e multiplicar por 2. Ao resultado deve ser acrescentado as manobras utilizadas no isolamento da falta.

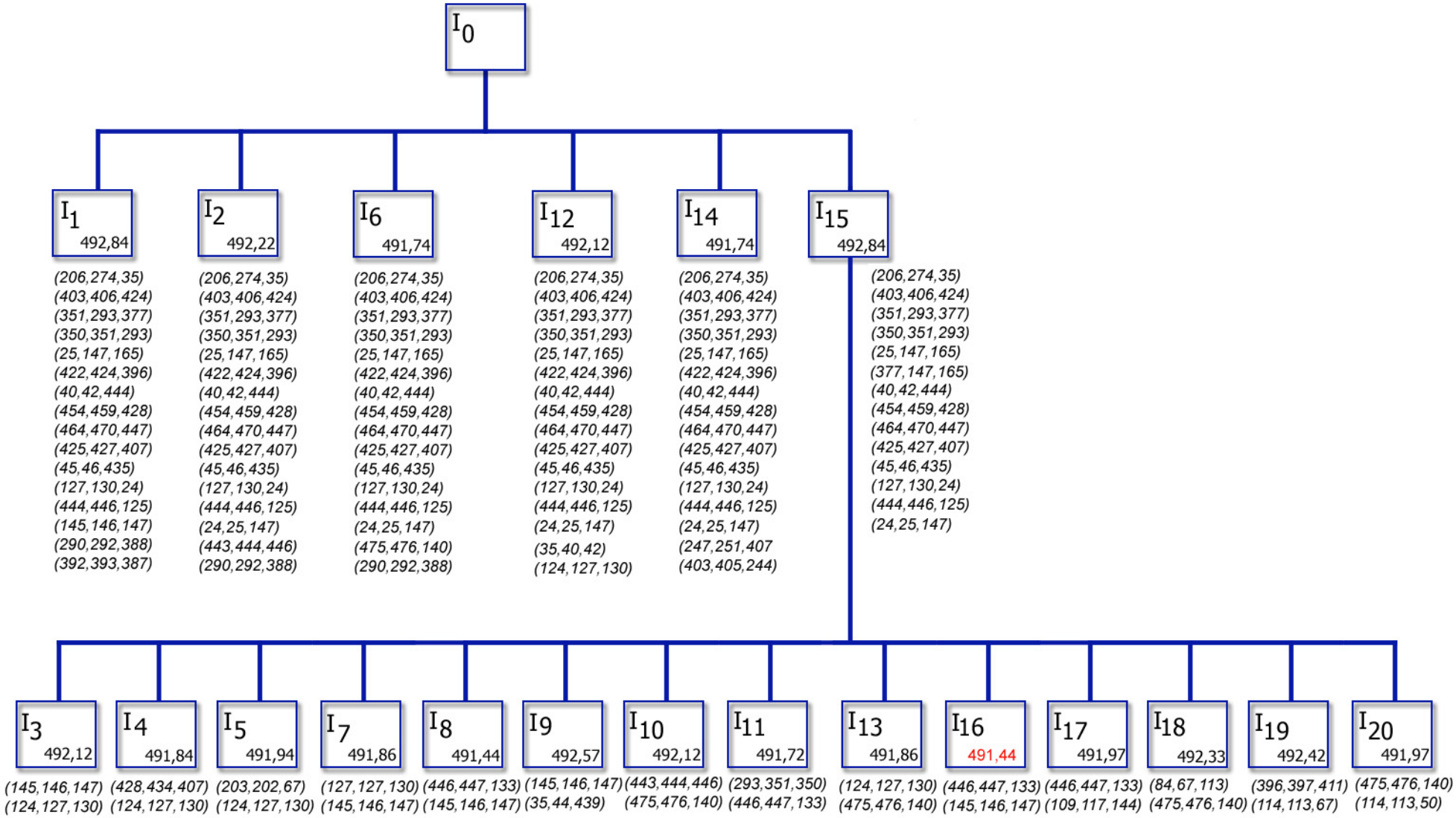



FIGURA 3.11: Árvore de ancestralidade da população final.

## 3.6 Resumo

Neste capítulo foi apresentado o problema de ocorrência de contingências em um sistema elétrico de potência na etapa de distribuição de energia primária, sejam elas devido a falhas ou manutenções preventivas no sistema. Primeiramente, uma formulação matemática para uma função objetivo que agregasse todos os objetivos considerados foi determinada de modo que técnicas de otimização mono-objetivo pudessem ser empregadas pelo algoritmo. Em seguida, o algoritmo DE-Tree proposto para ser utilizado em problemas de restauração foi apresentado destacando as fases da mutação diferencial, de cruzamento e a de busca local. O algoritmo apresentado é de propósito geral e pode ser empregado tanto em problemas de restauração de energia após uma contingência, caso aqui tratado, como também em problemas de redução de perdas de potência, queda de tensão, carregamento na rede e nas subestações, expansão da rede, etc; bastando para isto uma reformulação da função objetivo empregada de forma a atender estes tipos de problemas. Finalmente, foi apresentado a finalidade e a necessidade do uso de uma árvore de ancestralidade para o armazenamento de informações necessárias utilizadas pelo algoritmo desenvolvido.

# Capítulo 4

# Resultados Experimentais

Neste capítulo são apresentados os resultados de simulação computacional realizados em um SDE de grande porte com o intuito de observar o desempenho do algoritmo DE-Tree na resolução de problemas de contingências devido a faltas no sistema de distribuição de energia primária. Para tanto, o SDE da cidade de São Carlos-SP é utilizado como referência. A partir dele dois sistemas são investigados sendo que o primeiro é o sistema da cidade de São Carlos propriamente dito e o segundo é o Sistema da cidade de São Carlos duplicado. O Capítulo está dividido da seguinte maneira: a Seção 4.1 discorre sobre as principais características dos dois sistemas destacando as faltas simuladas nos setores durante os testes. A Seção 4.2 apresenta a formulação matemática da função objetivo implementada e, finalmente, na Seção 4.3 são apresentados os testes realizados e os resultados são comentados.

## 4.1 Testes Realizados

O desempenho do algoritmo DE-Tree foi testado na resolução de problemas de restauração de redes radiais de distribuição de energia primária no caso específico de contingências devido a faltas no sistema utilizando dois SDE de grande porte. O primeiro, doravante denominado **Sistema 1**, é a rede da cidade de São Carlos referente ao ano de 1994 [dos Santos 2009] composto por 3.860 barramentos, 533 setores, 632 chaves sendo 509 NF e 123 NA, 3 subestações e 23 alimentadores. O segundo, denominado **Sistema 2**, é composto por dois Sistemas 1 interligados por 13 novas chaves NA e composto, portanto, por 7.720 barramentos, 1.066 setores, 1.277 chaves sendo 1.018 NF e 259 NA, 6 subestações e 46 alimentadores. Tanto no Sistema 1 como no Sistema 2, dois testes para o restabelecimento de energia foram realizados: um considerando uma única falta e outro considerando três faltas ocorridas simultaneamente.

Os testes realizados para o restabelecimento de energia após uma única falta considera uma falta ocorrida no setor 504 interrompendo a distribuição de energia vinda do maior alimentador do sistema, o alimentador 23. A Figura 4.1 ilustra este caso destacando em detalhe as conexões do setor 504.

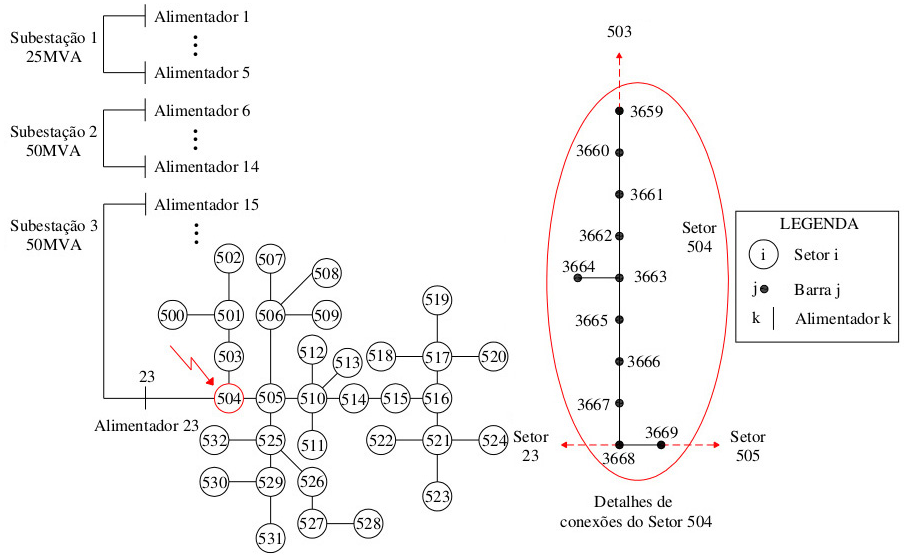



FIGURA 4.1: Alimentador 23 do SDE da cidade de São Carlos destacando uma falta ocorrida no setor 504 (Figura reproduzida de [dos Santos 2009] pág 115).

Ao se isolar o setor 504 do sistema, duas subárvores são geradas: uma com o setor 503 e outra com o setor 505 como raízes. Com o auxílio do CAO da RNP, estas duas subárvores são realocadas ao sistema. Assim, na primeira subárvore, o CAO escolhe um novo setor para ser a nova raiz, por exemplo o setor 501, e com o auxílio da Lista de Adjacências de chaves deste setor (dada por: setor 501 $\rightarrow$ 449 $-$ 500 $-$ 502 $-$ 503) escolhe-se um novo nó de enxerto, por exemplo o nó 449. O mesmo procedimento é realizado para a segunda subárvore gerada.

O teste para múltiplas faltas considera o caso em que três faltas simultâneas ocorrem nos sistemas, isto é, além do setor 504 uma falta no setor 182 do alimentador 6, como mostrado na Figura 4.2, e a outra no setor 486 do alimentador 22, Figura 4.3. Como no caso de uma única falta, após os setores faltosos serem isolados do sistema, os setores a jusante da falta são realocados utilizando o CAO da RNP de modo a restabelecer o fornecimento de energia a estes setores e o algoritmo DE-Tree é executado para otimizar os sistemas.

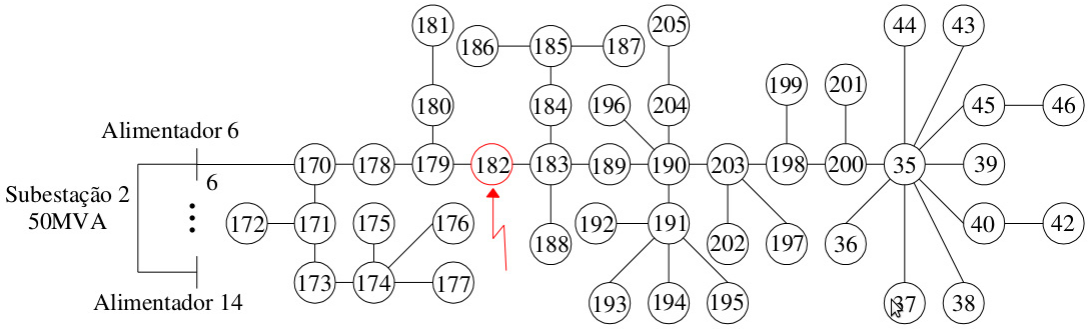



FIGURA 4.2: Alimentador 6 do SDE da cidade de São Carlos destacando uma falta ocorrida no setor 182 (Figura reproduzida de [dos Santos 2009] pág 117).

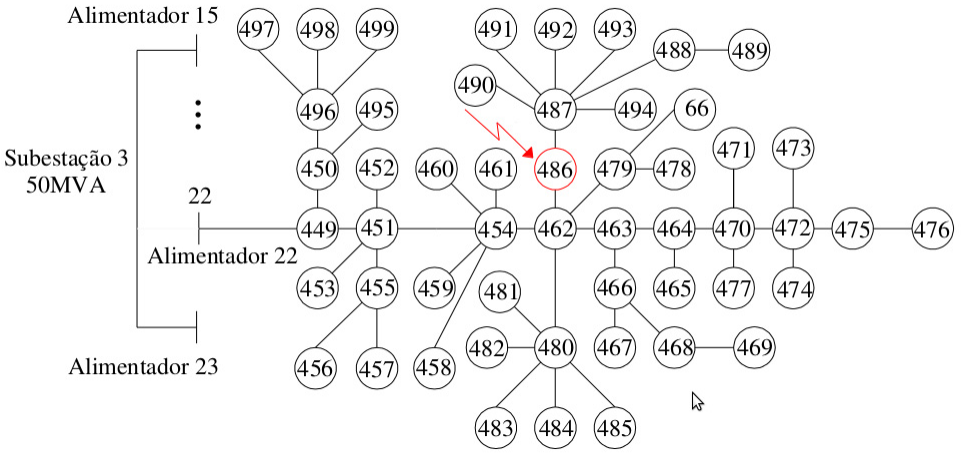



FIGURA 4.3: Alimentador 22 do SDE da cidade de São Carlos destacando uma falta ocorrida no setor 486 (Figura reproduzida de [dos Santos 2009] pág 117).

## 4.2 Função Objetivo

O problema de restauração de redes de SDE compreende vários propósitos ou objetivos a serem otimizados, como, por exemplo, o restabelecimento de energia após uma contingência, a redução de perdas de potência, a queda de tensão nas linhas de transmissão, o sobrecarregamento na rede e nas subestações, o balanceamento de cargas nos transformadores, estudos de previsão da expansão e alocação ótima de componentes na rede tais como capacitores, chaves, etc. [dos Santos 2009]. Dentre estes, no presente trabalho, foi considerado otimizar o sistema de modo a obter o rápido restabelecimento de energia após as contingências devido a ocorrência de faltas no setor primário da rede de distribuição. Assim, os objetivos a serem minimizados foram as perdas resistivas e o número de manobras de chaves necessárias para restabelecer o sistema. As restrições consideradas foram a maior queda de tensão nas linhas de transmissão, o maior carregamento da rede e das subestações. De maneira a utilizar técnicas de otimização mono-objetivo, os

objetivos e restrições foram agrupados em uma única Função Agregada ($Fg$) dada por:

$$Fg = (Pr + Nm) + (w_1.V + w_2.Cr + w_3.Cs) \tag{4.1}$$

onde:

- $Pr$ - são as Perdas Resistivas , dadas em p.u..
- $Nm$ - é o Número de Manobras de Chaves necessárias para reconfigurar o sistema.
- $V$ - é a Maior Queda de Tensão [%] permitida para as linhas de transmissão.
- $Cr$ - é o Maior Carregamento da Rede [%].
- $Cs$ - é o Maior Carregamento das Subestações [%].
- Os escalares $w_1, w_2$ *e* $w_3$ são penalizações aplicadas a função objetivo agregada caso alguma restrição seja violada e podem ser ajustados de acordo com a necessidade de cada rede de distribuição. Nos testes realizados estes parâmetros foram ajustados conforme os indicados abaixo:

$$w_1 = w_2 = w_3 = \begin{cases} 100, & \text{Se ocorrer alguma violação das restrições} \\ 0, & \text{caso contrário.} \end{cases}$$

No caso específico do Sistema 2, para o escalar $w_3$ que penaliza as violações da restrição Carregamento das Subestações foi necessário um valor maior de maneira a restringir o número de indivíduos que violam esta restrição dentro da população. Assim, neste sistema, ele é dado por:

$$w_3 = \begin{cases} 400, & \text{se ocorrer alguma violação da restrição} \\ 0, & \text{caso contrário.} \end{cases}$$

Os valores limites adotados para as restrições do sistema são:

- A máxima taxa da queda de tensão permissível limitada em 10%.
- A capacidade do carregamento da rede e das subestações, limitadas em 100%.

## 4.3 Resultados

Os parâmetros iniciais do algoritmo foram ajustados após uma bateria de testes preliminares com o intuito de melhor sintonizar o algoritmo para ambos os sistemas. Verificou-se que populações acima de 100 indivíduos não produzem melhoras significativas para as faltas simuladas e que 2.000 gerações eram o suficiente para a convergência dos sistemas e melhor observar o desempenho do algoritmo. Assim, adotou-se os seguintes parâmetros iniciais utilizados em todos os testes:

- Número de testes realizados: $t = 50$.
- Critério de parada: número de gerações $g = 2.000$.
- População inicial: $P_t = 60$.
- Escalar da equação da mutação diferencial: $\mathbf{F} = 0,6$.

A população inicial obtida a partir da configuração corrente, com os setores em falta devidamente isolados do SDE, foi gerada escolhendo-se aleatoriamente os Operadores PAO ou CAO para realizar a mutação dos indivíduos. Porém, durante a execução do algoritmo, na etapa de mutação diferencial os operadores são selecionados pelo *Método da Roleta* baseado em uma *Taxa de Adaptação* definida inicialmente em 50% para ambos os operadores. Se o vetor mutante, resultante da mutação diferencial, vencer o vetor corrente na disputa pela sobrevivência a taxa de adaptação do operador utilizado nesta mutação é incrementada e a do outro decrementada proporcionalmente. Desta forma, nas mutações seguintes, o operador que tiver maior taxa de adaptação é privilegiado. O Algoritmo 2 descreve o pseudocódigo do DE-Tree empregado durante os testes simulados.

**Algorithm 2** *DE-Tree - Pseudocódigo*

```
begin
    g ← 0                    (contador de geração)
    P_t ← 60                 (População Inicial)
    F ← 0,6                  (Escalar da mutação diferencial)
    g_max ← 2.000            (critério de parada)
    P_t                      (população inicial)
    f(P_t)                   (Avaliação da população inicial)
    while not g_max do
        for each x_i ∈ P_t do
            Operador = Seleção_por_Roleta(CAO, PAO)
            x_base ≠ x_1 ≠ x_2 ∈ [1, P_t]
            v_i = x_base ⊕ F ⊗ (x_1 ⊖ x_2)  (vetor mutante)
            if f(v_i) < f(x_i) then
                x_i(g+1) ← v_i
            else
                x_i(g+1) ← x_i
        g ← g + 1
end
```

No algoritmo DE-Tree utilizado na restauração dos dois sistemas, as seguintes observações devem ser destacadas:

- A estratégia utilizada na aplicação dos movimentos da lista de movimentos ao indivíduo base, quando da aplicação da equação da mutação diferencial, foi a Estratégia 2 apresentada na Seção 3.5.2 por apresentar maior diversidade de indivíduos

para o problema de restauração.

- Uma característica importante da DE é a variação do passo da mutação conforme a distribuição da população no espaço de busca, isto é, à medida em que a população converge menores são as mutações provocadas nos indivíduos e, como uma consequência direta disto, há uma convergência prematura da população. Para evitar, ou minimizar, esta convergência prematura, o DE, na sua versão clássica, realiza a fase de cruzamento logo após a fase de mutação, cruzando o vetor mutante com o indivíduo corrente com o intuito de gerar pertubações no indivíduo resultante de modo a aumentar a diversidade dos indivíduos da população. Uma solução encontrada para cruzar duas soluções com estrutura de dados do tipo árvore sem gerar soluções infactíveis foi utilizar a DE, como apresentada na Seção 3.5.3, novamente para realizar o cruzamento. Entretanto, melhoras significativas só foram verificadas com um aumento significativo do número de indivíduos da população o que onerava o tempo de resposta do algoritmo. Como o problema de restauração de SDE é atender o maior número possível de consumidores e em um tempo relativamente curto em detrimento da solução ótima, optou-se por omitir a fase de cruzamento do algoritmo. Com isso, o vetor experimental passa a ser o vetor obtido quando da mutação diferencial.
- De maneira habitual em sistemas discreto, após a fase de seleção, é de praxe realizar uma busca local no indivíduo selecionado para a próxima geração utilizando uma estrutura de vizinhança previamente definida de modo a apurar a função objetivo encontrada. Entretanto, da mesma forma como observado na etapa de cruzamento, a busca local em problemas de restauração tornou muito alto o tempo de processamento. Portando, esta fase também foi suprimida do algoritmo.

As configurações correntes dos dois sistemas antes das ocorrências das faltas são apresentadas na Tabela 4.1, na qual se observa que, enquanto no Sistema 1 as três restrições impostas ao sistema estão dentro dos limites estipulados, no Sistema 2 as restrições de carregamento da rede e de carregamento das subestações estão quase em seu limite máximo.

Tabela 4.1: Configurão corrente dos sistemas antes da ocorrência das faltas.

| | Pr [kW] | V [%] | Cr [%] | Cs [%] |
|---|---|---|---|---|
| Sistema 1 | 281.27 | 3.25 | 67.12 | 53.34 |
| Sistema 2 | 859.93 | 3.31 | 98.93 | 99.99 |

A Tabela 4.2 mostra, a título de exemplo, configurações típicas encontradas para os sistemas após os setores faltosos serem devidamente isolados e os setores a jusante serem realocados com a aplicação do PAO. Observa-se na tabela que há uma tendência

de os sistemas se sobrecarregarem violando as restrições de carregamento da rede e das subestações.

TABELA 4.2: Exemplo de Configurações para os sistemas após da ocorrência das faltas.

| | Sistema 1 | | Sistema 2 | |
|---|---|---|---|---|
| | 1 Falta | 3 Faltas | 1 Falta | 3 Faltas |
| $P_r[kW]$ | 382,61 | 409,04 | 955,08 | 971,59 |
| $V$ [%] | 4,35 | 4,47 | 4,58 | 3,31 |
| $C_r$[%] | 139,60 | 139,60 | 98,93 | 98,93 |
| $C_s$[%] | 52,78 | 52,77 | 105,28 | 99,66 |

### 4.3.1 Testes Para Falta Única

A Tabela 4.3 mostra os valores máximo, mínimo, médio e o desvio padrão dos resultados obtidos para uma única falta para ambos os sistemas, quando da minimização da Função da Agregada. Observa-se nesta tabela que, apesar do setor em falta, os carregamentos da rede e das subestações não ultrapassam os valores máximos permitidos respeitando as restrições dos sistemas. Neste caso, como as restrições não foram violadas, a função agregada passa a ser tão somente as duas funções objetivos: perdas resistivas e o número de manobras de chaves.

TABELA 4.3: Resultados obtidos para uma única falta no setor 504.

| | Sistema 1 | | | | Sistema 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Máx | Mín | Média | DP($\sigma$) | Máx | Mín | Média | DP($\sigma$) |
| Geração | 1999 | 363 | 1579.20 | 430.13 | 1999 | 697 | 1569.12 | 379.69 |
| Pr [kW] | 318.77 | 290.67 | 302.07 | 6.05 | 953.70 | 884.87 | 902.13 | 16.34 |
| V [%] | 3.30 | 3.13 | 3.24 | 0.03 | 4.68 | 3.24 | 3.47 | 0.42 |
| Cr [%] | 93.31 | 69.68 | 78.19 | 5.51 | 98.93 | 86.37 | 98.52 | 1.85 |
| Cs [%] | 56.71 | 51.24 | 53.47 | 1.00 | 100.00 | 99.62 | 99.90 | 0.08 |
| Nm | 84 | 40 | 61.16 | 11.08 | 94 | 6 | 39.28 | 17.97 |
| t [s] | 33.16 | 29.48 | 31.09 | 1.03 | 118.81 | 105.47 | 112.84 | 3.01 |

Se comparado os valores obtidos antes da ocorrência da falta e os valores médios obtidos após a falta apresentados nas tabelas acima, observa-se que para o Sistema 1 houve um aumento de $7,4\%$ nas perdas resistivas e $12,5\%$ no carregamento da rede, enquanto que, o carregamento das subestações e a queda de tensão permaneceram praticamente os mesmos. Para o Sistema 2 as perdas resistivas tiveram um aumento de $4,9\%$ e os demais objetivos são praticamente os mesmos. Estes são valores aceitáveis uma vez que o maior alimentador está isolado do sistema e que o algoritmo, ao minimizar uma função agregada, privilegiou indivíduos que não infringissem as restrições impostas

pelo sistema em detrimento daqueles que apresentavam melhores perdas resistivas. Vale ressaltar também que, para o Sistema 2, o valor mínimo encontrado para o número de manobras de chaves (6 manobras) é devido a uma convergência prematura da população e não um valor ótimo para o objetivo Número de Manobras de chaves. Provavelmente esta estagnação da população deve-se aos estados iniciais dos carregamentos da rede e das subestações que impediram uma convergência para valores melhores.

As Tabelas 4.4 e 4.5 mostram, em negrito, os melhores valores encontrados para cada objetivo e restrição respectivamente para o Sistema 1 e o Sistema 2 . Assim, no Sistema 1 por exemplo, dos 50 testes realizados a melhor perda resistiva encontrada foi de $290,67kW$ na geração 1.838 com 84 operações de manobras de chaves. As restrições neste teste não ultrapassaram seus valores limites, sendo que a queda de tensão ficou em $3,25\%$, o carregamento da rede em $73,12\%$ e o carregamento das subestações em $56,71\%$ em um tempo de execução do algoritmo de $30,30s$. Fica evidente nestas tabelas, para um decisor, qual dos objetivos/restrições dos sistemas deve ser preservado em detrimento dos demais.

TABELA 4.4: Melhor resultado encontrado para cada objetivo/restrição para uma falta no setor 504 do Sistema 1.

| Melhor | Ger. | Pr[kW] | Nm | V [%] | Cr [%] | Cs [%] | t [s] |
|---|---|---|---|---|---|---|---|
| Pr [kW] | 1838 | **290.67** | 84 | 3.25 | 73.12 | 56.71 | 30.30 |
| Nm | 1636 | 307.53 | **40** | 3.25 | 80.23 | 55.67 | 29.48 |
| V [%] | 830 | 306.29 | 64 | **3.13** | 81.17 | 53.56 | 30.53 |
| Cr [%] | 1962 | 310.49 | 52 | 3.27 | **69.68** | 52.34 | 31.21 |
| Cs [%] | 1109 | 301.64 | 52 | 3.24 | 79.48 | **51.24** | 32.51 |

TABELA 4.5: Melhor resultado encontrado para cada objetivo/restrição para uma falta no setor 504 do Sistema 2.

| Melhor | Ger. | Pr [Kw] | Nm | V [%] | Cr [%] | Cs[%] | t [s] |
|---|---|---|---|---|---|---|---|
| Pr [kW] | 1944 | **884.87** | 28 | 3.31 | 98.93 | 99.91 | 114.94 |
| Nm | 1996 | 887.31 | **6** | 3.91 | 98.93 | 99.93 | 113.78 |
| V [%] | 1797 | 895.02 | 30 | **3.24** | 98.93 | 99.86 | 112.76 |
| Cr [%] | 1509 | 889.87 | 32 | 3.30 | **86.37** | 99.92 | 112.51 |
| Cs [%] | 1239 | 892.36 | 54 | 3.31 | 98.93 | **99.62** | 115.83 |

As Figuras 4.4 e 4.5 mostram, respectivamente para o Sistema 1 e o Sistema 2, os valores obtidos para os objetivos e restrições do sistema para o melhor teste realizado quando da minimização da Função Agregada. Observa-se nestas figuras a rápida convergência da Função Agregada bem antes de 50 gerações. A partir daí, como as restrições dos sistemas não são mais violadas, a minimização da função agregada passa a ser a minimização tão somente dos objetivos Perdas Resistivas ($P_r$) e o Número de Manobras

de Chaves ($N_m$). Esta rápida convergência reflete uma característica importante do algoritmo DE que é a adaptação do passo da mutação de acordo com a distribuição da população no espaço de busca, como mencionado anteriormente.

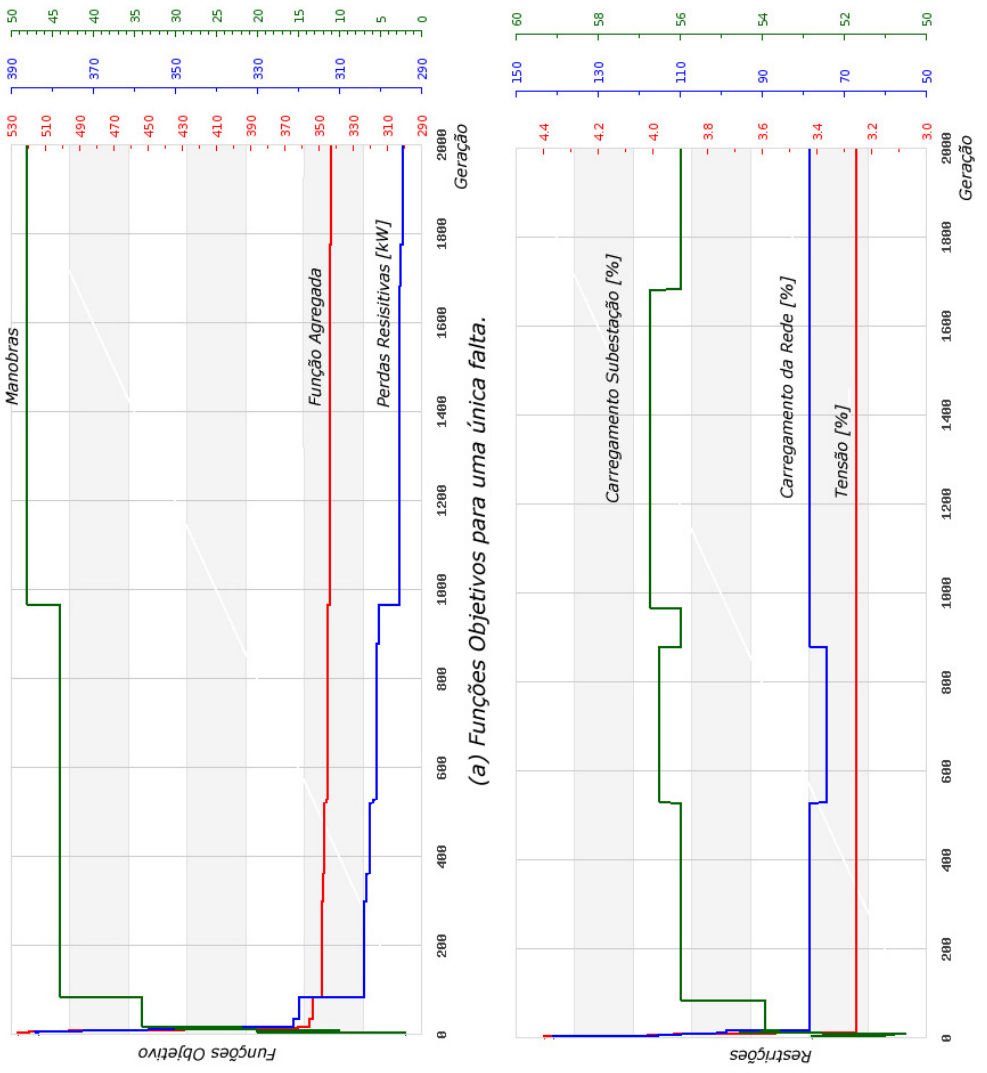



FIGURA 4.4: Melhor solução encontrada para a Função Agregada para uma única falta no Sistema 1.

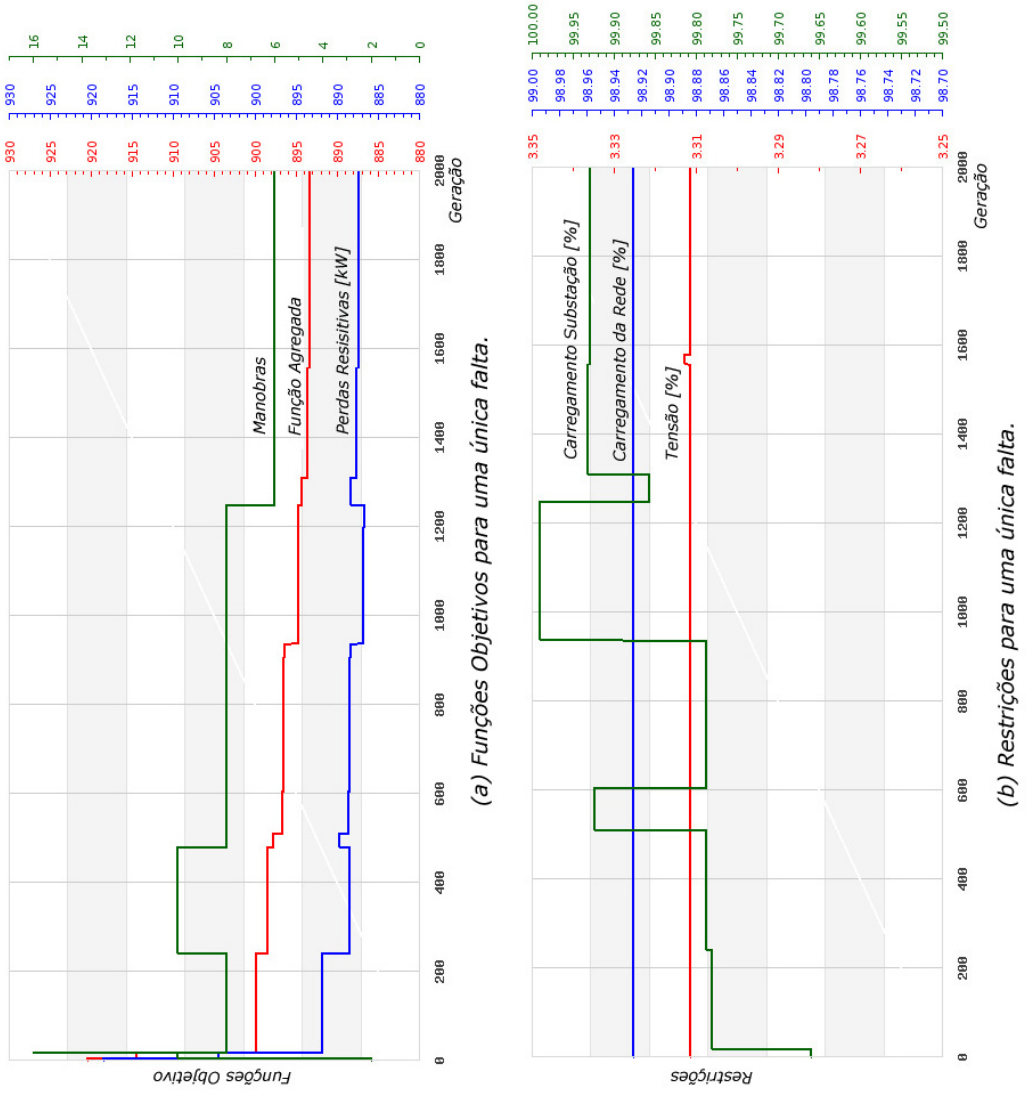



FIGURA 4.5: Melhor solução encontrada para a Função Agregada para uma única falta no Sistema 2.

### 4.3.2 Testes para Múltiplas Faltas

Assim como os testes realizados para uma única falta, a Tabela 4.6 mostra os valores máximo, mínimo, médio e o desvio padrão obtidos de 50 testes realizados nos dois sistemas para o caso de três faltas simultâneas. Observa-se que, neste caso, para o Sistema 1 as perdas resistivas tiveram um aumento de $9,68\%$, o carregamento da rede um aumento de $18,1\%$ e o carregamento das subestações um aumento de 10%. Para o Sistema 2, as perdas resistivas aumentaram em $8,86\%$ enquanto que os carregamentos da rede e das subestações tiveram uma pequena redução.

TABELA 4.6: Resultdos obtidos para 3 faltas simultâneas ocorridas nos setores 504, 182 e 486.

| | Sistema 1 | | | | Sistema 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Máx | Mín | Média | DP($\sigma$) | Máx | Mín | Média | DP($\sigma$) |
| Geração | 1999 | 551 | 1628.44 | 345.96 | 1996 | 327 | 1566.98 | 435.53 |
| Pr [kW] | 326.32 | 292.99 | 308.50 | 7.42 | 991.99 | 910.24 | 936.15 | 20.13 |
| V [%] | 3.33 | 3.18 | 3.25 | 0.02 | 4.58 | 3.25 | 3.41 | 0.34 |
| Cr [%] | 92.40 | 68.20 | 79.25 | 5.54 | 99.48 | 87.53 | 97.82 | 2.39 |
| Cs [%] | 63.80 | 52.94 | 58.70 | 2.38 | 99.99 | 99.62 | 99.91 | 0.08 |
| Nm | 84 | 28 | 52.64 | 11.83 | 184 | 22 | 79.24 | 37.40 |
| t [s] | 33.61 | 29.58 | 31.16 | 0.92 | 107.05 | 98.56 | 102.43 | 1.98 |

As Tabelas 4.7 e 4.8 mostram, em negrito, os melhores valores encontrados para cada objetivo e restrição respectivamente para o Sistema 1 e o Sistema 2.

TABELA 4.7: Melhor reusltado encontrado para cada objetivo/restrição para 3 Faltas simultâneas ocorridas no Sistema 1.

| Melhor | Ger. | Pr [kW] | Nm | V [%] | Cr [%] | Cs [%] | t [s] |
|---|---|---|---|---|---|---|---|
| Pr [kW] | 1744 | **292.99** | 56 | 3.25 | 74.23 | 58.58 | 30.43 |
| Nm | 1463 | 313.59 | **28** | 3.25 | 85.16 | 57.65 | 33.61 |
| V [%] | 1667 | 308.73 | 66 | **3.18** | 85.16 | 57.46 | 31.61 |
| Cr [%] | 1955 | 326.32 | 50 | 3.33 | **68.20** | 56.43 | 30.26 |
| Cs [%] | 1989 | 311.22 | 36 | 3.25 | 74.23 | **52.94** | 30.56 |

TABELA 4.8: Melhor reusltado encontrado para cada objetivo/restrição para 3 Faltas simultâneas ocorridas no Sistema 2.

| Melhor | Ger. | Pr [Kw] | Nm | V [%] | Cr [%] | Cs [%] | t [s] |
|---|---|---|---|---|---|---|---|
| Pr [Kw] | 1885 | **910.24** | 34 | 3.28 | 98.05 | 99.89 | 104.03 |
| Nm | 1643 | 928.78 | **22** | 3.31 | 98.93 | 99.93 | 102.39 |
| V [%] | 1722 | 933.66 | 90 | **3.25** | 98.93 | 99.96 | 101.17 |
| Cr [%] | 1232 | 919.09 | 84 | 3.30 | **87.53** | 99.96 | 103.40 |
| Cs [%] | 1699 | 954.81 | 138 | 3.31 | 92.66 | **99.62** | 105.04 |

As Figuras 4.6 e 4.7 mostram, respectivamente para o Sistema 1 e o Sistema 2, os valores obtidos para os objetivos e restrições do sistema para o melhor teste realizado quando da minimização da Função Agregada no caso de três faltas simultâneas. Pode-se observar nas figuras o problema dos conflitos entre os objetivos e as restrições do sistema. No Sistema 1 é mais evidente o aumento do carregamento da rede quando da minimização das perdas resistivas e, no caso do sistema 2 o carregamento das subestações. Observa-se que, apesar disto, novamente após a convergência da Função Agregada, as restrições do sistema não são mais violadas.

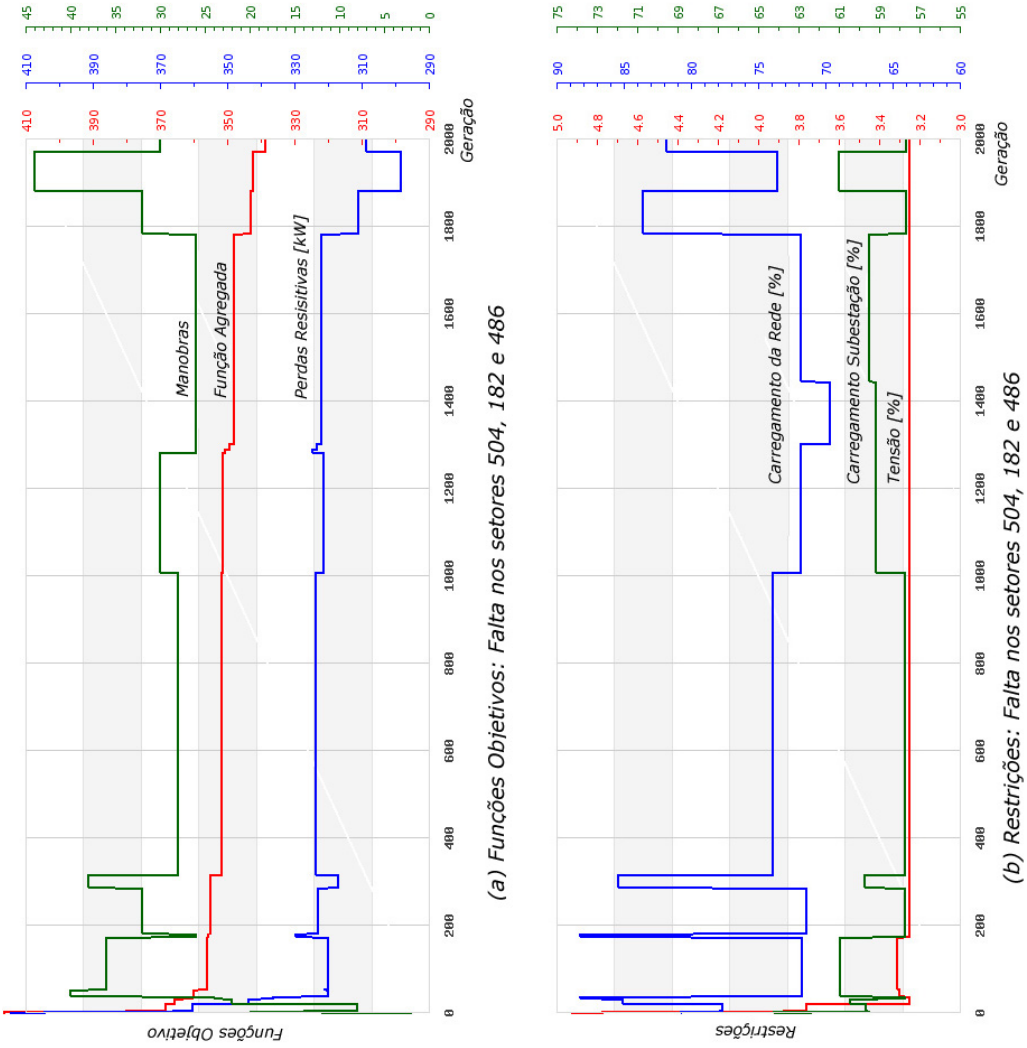



FIGURA 4.6: Melhor solução encontrada para a Função Agregada para três faltas simultâneas no Sistema 1.

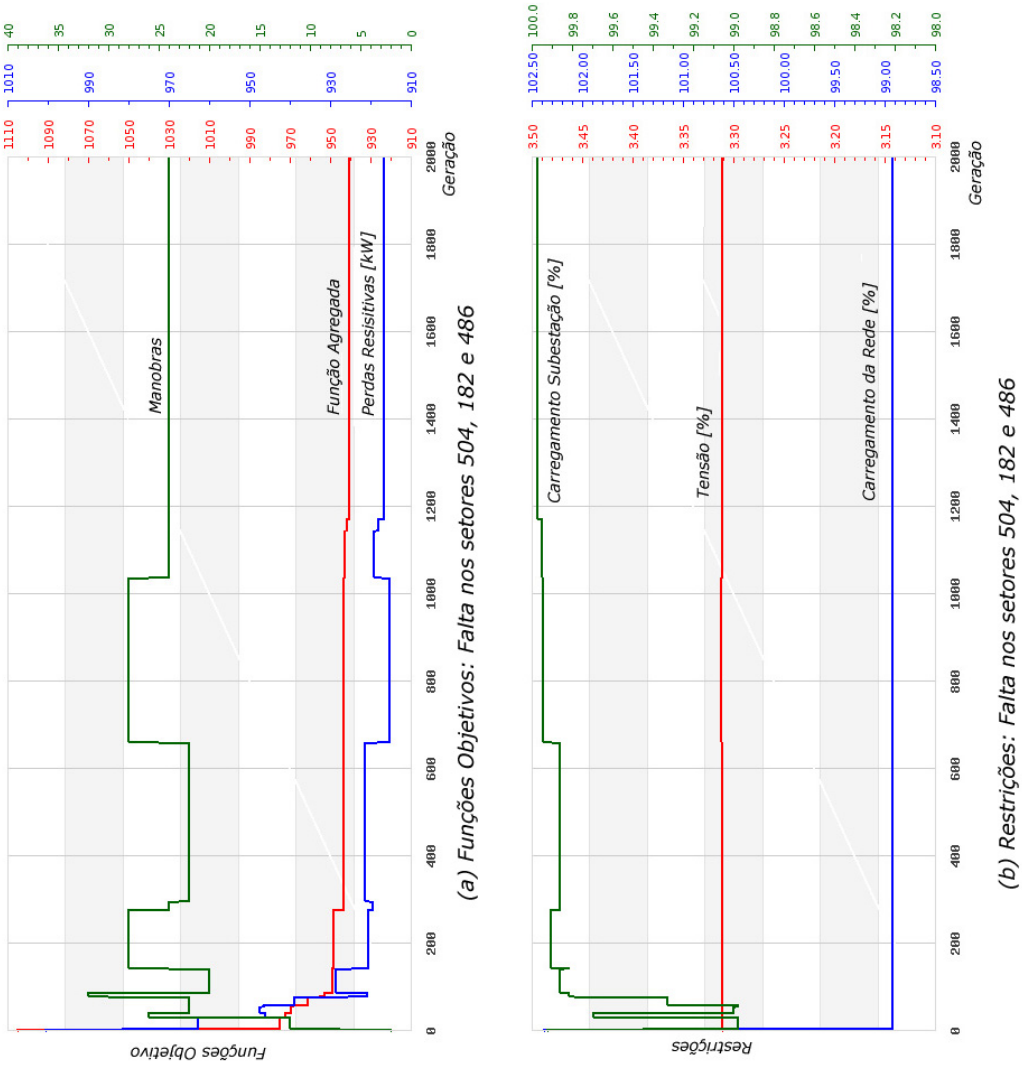



FIGURA 4.7: Melhor solução encontrada para a Função Agregada para três faltas simultâneas no Sistema 2.

### 4.3.3 Comparação do DE-Tree com outras Abordagens

Nesta seção, os resultados obtidos são comparados com outras duas abordagens apresentadas na literatura, para os mesmos sistemas, com as mesmas faltas e configurações iniciais mostradas na Tabela 4.1, aplicadas em problema de restauração de redes.

O primeiro algoritmo, denominado AEMT[1] apresentado em [dos Santos 2009], é um AE que trabalha com várias subpopulações em paralelo, armazenadas em tabelas, onde os melhores indivíduos para cada objetivo são armazenados em sua respectiva subpopulação. Além destas subpopulações, é acrescentada uma subpopulação para a função agregada, a qual é utilizada na comparação com o DE-Tree. O segundo algoritmo utilizado para comparação, foi o algoritmo denominado MEAN-DE [2] proposto em [Sanches et al. 2013] também um algoritmo baseado em tabelas de subpopulações mas, na fase de mutação do algoritmo, é utilizada uma mutação diferencial com lista de movimentos como proposto neste trabalho. Estes dois algoritmos não são baseados em populações e sim em um indivíduo inicial que, ao sofrer mutações, vai populando as subpopulações. Todos os três algoritmos utilizam a RNP na evolução dos indivíduos.

A Tabela 4.9 compara os valores médios obtidos pelos três algoritmos para os testes com uma única falta e três faltas simultâneas no Sistema 1. Observa-se nesta tabela que os três algoritmos apresentam desempenhos bem semelhantes, exceto pelo objetivo Número de Manobras de chaves no qual o MEAN-DE teve um desempenho bem superior.

Tabela 4.9: Valores Médios obtidos com os algoritmos DE-Tree, AEMT e MEAN-DE

| Falta Única no Sistema 1 | | | | | |
|---|---|---|---|---|---|
| | Pr [kW] | V [%] | Cr [%] | Cs [%] | Nm |
| DE-Tree | 302,07 | 3,24 | 78,19 | 53,47 | 61,16 |
| AEMT | 294,08 | 3,24 | 76,69 | 54,31 | 24,87 |
| MEAN-DE | 377,20 | 4,10 | 77,70 | 53,90 | 9,00 |

| 3 Faltas Simultâneas no Sistema 1 | | | | | |
|---|---|---|---|---|---|
| DE-Tree | 308,50 | 3,25 | 79,25 | 58,70 | 52,64 |
| AEMT | 302,46 | 3,25 | 77,71 | 59,33 | 28,20 |

A Tabela 4.10 apresenta os valores médios obtidos com os algoritmos DE-Tree e MEAN-DE para o caso de uma única falta no Sistema 2.

[1] Algoritmo Evolutivo Multiobjetivo em Tabelas
[2] *Multiobjective Evolutionary Algorithm with Differential Mutation Operator*

TABELA 4.10: Valores Médios obtidos com os algoritmos DE-Tree e MEAN-DE

| Única falta no Sistema 2 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Pr [kW] | | V [%] | | Cr [%] | | Cs [%] | | Nm | |
| | Média | DP | Média | DP | Média | DP | Média | DP | Média | DP |
| DE-Tree | 902,13 | 16,34 | 3,47 | 0,42 | 98,52 | 1,85 | 99,90 | 0,08 | 39,28 | 17,97 |
| MEAN-DE | 639,90 | 40,80 | 3,90 | 0,70 | 79,40 | 8,30 | 55,10 | 1,90 | 16,00 | 10,73 |

Comparando os resultados apresentados na tabela verifica-se que no DE-Tree houve uma convergência prematura da população impedindo que o algoritmo saísse de um ótimo local. Esta convergência prematura indica a necessidade de uma população maior para o Sistema 2 de modo a aumentar a diversidade dos indivíduos da população. No entanto, existe um conflito entre o aumento da população e o tempo de processamento: um amento na população implica em um tempo de processamento maior, o que não é interessante em problema de restauração de SDE. Uma característica importante que pode justificar também esta convergência prematura do DE por lista de movimentos em relação ao DE por tabelas é o método de substituição utilizado pelos dois algoritmos. No MEAN-DE o método utilizado na seleção dos indivíduos da população é o *Steady State* que garante uma diversidade de indivíduos maior, enquanto que o DE-Tree utiliza a substituição geracional que, por sua vez, pode gerar um número excessivo de filhos de um mesmo pai.

### 4.3.4 Simulação do Método de Monte Carlo

De maneira a verificar a robustez e a eficácia do algoritmo DE-Tree na restauração de SDE foi simulado o Método de Monte Carlo considerando que faltas aleatórias pudessem ocorrer em qualquer um dos setores do **Sistema 1**. Para este teste foram simuladas 50 faltas em diferentes setores do sistema e, para cada uma delas, uma nova configuração foi gerada e considerada a configuração inicial para o algoritmo DE-Tree. Para cada falta simulada o algoritmo foi rodado 10 vezes totalizando, assim, 500 simulações para o método de Monte Carlo. A otimização foi realizada com o intuito de restabelecer o SDE. Os melhores valores para os objetivos e restrições foram, então, armazenados para cada simulação realizada.

As Figuras 4.8 à 4.10 ilustram o diagrama de caixas dos objetivos e restrições obtidos na simulação do método de Monte Carlo. A Figura 4.8 mostra o diagrama de caixa para a função agregada e o objetivo perdas resistivas. A Figura 4.9 o diagrama de caixa para as restrições de carregamento da rede e das subestações e, finalmente, na Figura 4.10 o diagrama de caixa para o número de operações de chaveamento e a restrição na queda de tenão no SDE.

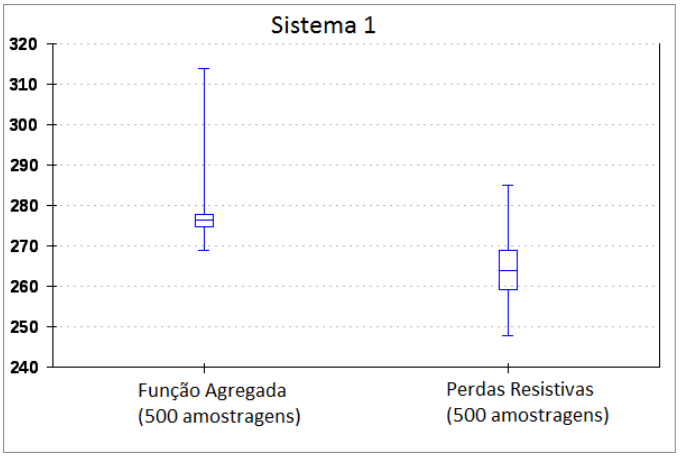



FIGURA 4.8: Diagrama de caixa da Função Objetivo Agregada e Perdas Resistivas.

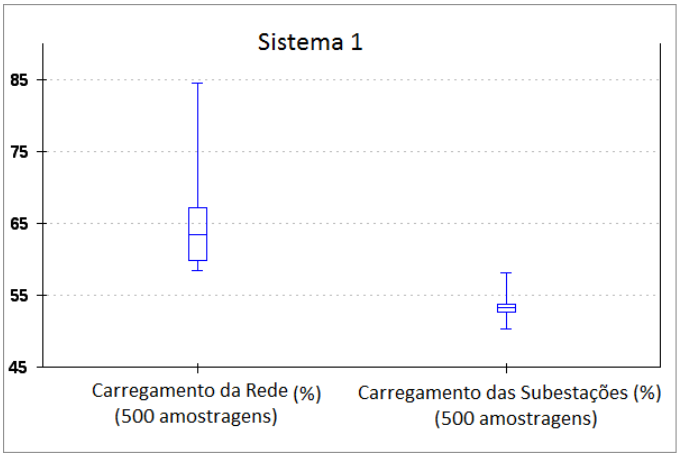



FIGURA 4.9: Diagrama de caixa das restrições de Carregamento da Rede e Carregamento das Subestações.

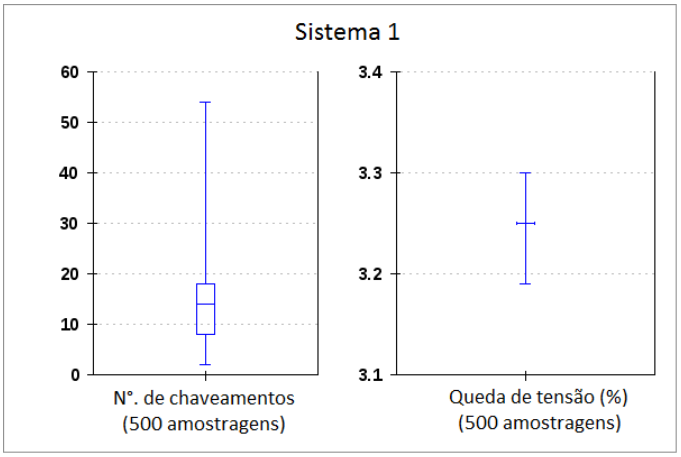



FIGURA 4.10: Diagrama de caixa para o Número de operações de chaveamentos e restrições de queda de tensão.

Observa-se, nestas figuras, que o algoritmo DE-Tree é um método bastante robusto na resolução do problema de restauração de serviço no SDE. As variações nas perdas resistivas, na restrição da queda de tensão, bem como nas restrições de carregamento da rede e das subestações são relativamente pequenas. O algoritmo mostrou-se hábil em restaurar o serviço em uma variedade de situações de falta única no sistema. Nas simulações, o objetivo número de chaveamentos necessários para restaurar o sistema mostrou-se a variável mais sensível. No entanto, a maioria dos valores situaram-se entre 8 e 20 operações de chaveamento.

Melhores soluções, com um menor número de chaveamentos, podem ser obtidas com o algoritmo DE-Tree ($i$) se a importância relativa do objetivo número de operações de chaveamento na função objetivo agregada for aumentada, ver Eq. 3.2; ($ii$) Ou se as restrições de carregamento da rede e das subestações forem relaxadas, o que é usual na prática.

### 4.3.5 Rastreamento do Número de Operações de Chaveamento

Durante todo o processo evolucionário, a Árvore de Ancestralidade utilizada no algoritmo DE-Tree é atualizada com todas as alterações de chaveamento que leva uma solução à outra na árvore. Ao final do processo de otimização, é possível rastrear a sequência de abertura/fechamento de chaves necessária para que, partindo da configuração inicial $I_0$, obtenha-se a melhor solução. Esta informação é bastante útil porque é possível rastrear todo o progresso da tarefa de restauração de serviço do SDE à medida em que a sequência de operações de abertura e fechamento de chaves do sistema é realizada.

Para concluir a etapa de testes realizados neste trabalho, um exemplo é apresentado mostrando como a Árvore de Ancestralidade retornada pelo algoritmo DE-Tree pode ser útil no processo de restauração do SDE. A Fig 4.11 ilustra esta ideia para um teste do algoritmo DE-Tree realizado arbitrariamente no caso de uma falta aleatória ocorrida no **Sistema 1**. Neste exemplo, a melhor solução em todo o processo de otimização foi encontrada após 66 operações de chaveamentos a partir da configuração inicial. Na figura pode-se observar o progresso das restrições de carga e de tensão, bem como do objetivo perdas resistivas à medida que caminha-se da configuração inicial até a melhor solução. Como as operações de chaveamento podem ser feitas tanto remotamente, quando possível, ou realizadas manualmente por uma equipe enviada ao local da falta, é possível avaliar o custo-benefício de cada operação de chaveamento antes da melhor solução ser implementada.

Por exemplo, na Fig 4.11 pode-se ver que com 4 operações de chaveamento obtém-se uma configuração em que a restrição carregamento da rede é levemente violada (em

torno de 115%) mas que as restrições de carregamento das subestações e queda de tensão estão em níveis aceitáveis. A companhia de energia pode, então, decidir por uma pequena operação de chaveamento em detrimento do carregamento da rede estar acima do limite por algum tempo e as perdas resistivas estarem altas ou então, com 22 operações de chaveamento, obter perdas resistivas significantemente abaixas e todas as restrições satisfeitas. Realizando-se todas as 66 operações de chaveamento tem-se reduzidas perdas resistivas e restrições do sistema.

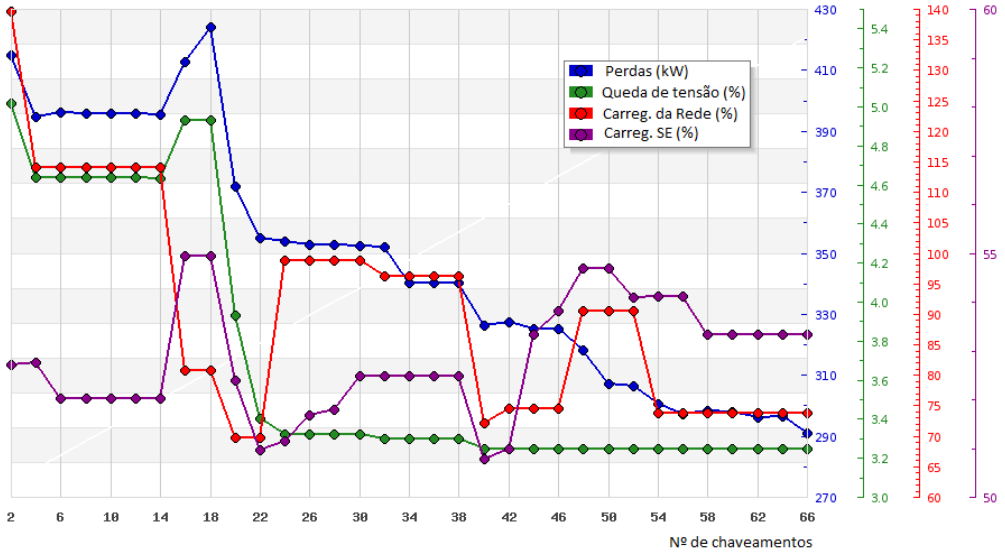



FIGURA 4.11: Operações de chaveamento a partir da configuração inicial até a melhor solução na Árvore de Ancestralidade.

# Capítulo 5

# Conclusões

Os principais objetivos alcançados, estabelecidos no início deste trabalho, são apresentados a seguir: (*i*) realizou-se um criterioso estudo bibliográfico das principais abordagens apresentadas para a evolução diferencial no domínio das variáveis discretas; (*ii*) cada uma das técnicas abordadas foi analisada e suas vantagens e desvantagens avaliadas; (*iii*) propôs-se, então, uma nova abordagem para a evolução diferencial que fosse genérica para os problemas de otimização combinatória e que preservasse aquelas características observadas na evolução diferencial no domínio das variáveis contínuas; e, por fim, (*iv*) a abordagem sugerida foi implementada em um algoritmo denominado DE-Tree e testado em um SDE real de grande porte. A seguir, apresenta-se uma visão geral sobre a pesquisa realizada discorrendo sobre os ganhos atingidos por este trabalho.

No Capítulo 2 realizou-se uma introdução geral sobre os algoritmos para a evolução diferencial discreta, classificando-os em dois grupos: um em que os operadores aritméticos da equação da mutação diferencial não são definidos no espaço das variáveis discretas e, consequentemente, muitas vezes necessitam de reparos nas soluções obtidas; e o outro em que estes operadores são definidos mas são aplicados tão somente em problemas de permutação.

Em contrapartida, na abordagem proposta, a definição dada ao operador diferença é genérica e sempre retorna uma Lista de Movimentos definida segundo o problema combinatório em questão e da estrutura de dados utilizada na representação das soluções. O operador multiplicação por escalar é uma ponderação dos movimentos da lista que serão utilizados, enquanto que, o operador soma é definido de modo a aplicar os movimentos ponderados ao vetor base. Desta forma, o algoritmo proposto pode ser aplicado em uma grande gama de problemas combinatórios, não restringindo-se a, tão somente, um dado

tipo de problema desta classe. Exemplos de possíveis definições para a Lista de Movimentos foram apresentadas para algumas estruturas de dados comumente utilizadas na representação de problemas clássicos discretos.

No Capítulo 3 fez-se uma introdução ao SDE enfatizando o problema de contingências devido a faltas no sistema de distribuição de energia no setor primário. Apresentou-se uma formulação matemática para este tipo de problema e definiu-se uma função objetivo que agregasse todos os objetivos e restrições consideradas do sistema de modo que técnicas de otimização mono-objetivo pudessem ser aplicadas. Propôs-se, então, uma versão do algoritmo com lista de movimentos aplicado na resolução deste tipo de problema. O algoritmo, denominado DE-Tree, utiliza a RNP na representação da estrutura de dados das soluções e seus operadores na realização da mutação diferencial. Uma estrutura de dados denominada Árvore de Ancestralidade foi definida para, além de ser utilizada para construir a lista de movimentos, armazenar as manobras de chaves necessárias para reconfigurar o sistema.

No Capítulo 4 a abordagem sugerida foi implementada e testada em um problema real de restauração de sistemas de energia elétrica e os resultados obtidos foram comparados com outros dois algoritmos utilizados neste tipo de problema (AEMT e MEAN-DE).

## 5.1 Contribuições da Tese

Os pontos que representam importantes contribuições originais deste trabalho são enumerados a seguir:

- Revisão crítica e sistemática dos principais algoritmos para a evolução diferencial discreta relacionados na literatura especializada.
- Proposição de uma nova abordagem para a evolução diferencial discreta que, além de apresentar as vantagens observadas nas abordagens existentes, é mais abrangente no contexto da otimização combinatória.
- Apresentação do algoritmo DE-Tree, baseado na estratégia proposta, e a avaliação deste algoritmo em um problema real de restauração de um SDE.
- Os resultados experimentais realizados mostraram que o algoritmo sugerido produz resultados competitivos aos apresentados por aquelas abordagens utilizadas como comparação.

Partes substanciais deste trabalho foram desenvolvidas ao longo destes anos de doutoramento e encontram-se em [Prado et al. 2010a;b;c; 2013, Sanches et al. 2013, Guimarães et al. 2012]

## 5.2 Sugestões para Trabalhos Futuros

Os itens a seguir apresentam algumas sugestões para trabalhos futuros.

**Investigação da possibilidade de intercâmbio entre as técnicas utilizadas pelos algoritmos aplicados na restauração de SDE:** Os algoritmos AEMT e o MEAN-DE utilizam o recurso de tabelas para o armazenamento das manobras de chaves necessárias para a restauração do sistema e o recurso de subpopulações no armazenamento dos melhores indivíduos para seleção. O DE-Tree, por sua vez, utiliza a árvore de ancestralidade para armazenar as informações necessárias para a manobra de chaves e, como método de seleção, o proposto pelo algoritmo DE clássico, ou seja, a disputa entre o vetor corrente e o vetor experimental. Em princípio, parece razoável investigar os resultados que podem ser obtidos através do intercâmbio destas técnicas entre estes algoritmos. Pode-se citar como exemplos:

- Substituir o método clássico de seleção do DE utilizado no algoritmo DE-Tree pelo método de subpopulações utilizado nos algoritmos AEMT e MEAN-DE;
- Verificar a eficiência da estrutura de dados da Árvore de Ancestralidade no armazenamento das manobras de chaves em um AE multiobjetivo.

**Investigação da aplicação do algoritmo DE-Tree em outros problemas de restauração de redes:** A abordagem proposta para a evolução diferencial, por ser genérica, pode ser aplicada em outros problemas de restauração de redes destacando-se, por exemplo, o estudo e planejamento da expansão do sistema de distribuição, redução de perdas de potência, balanceamento de cargas, etc. Nestes casos, as etapas de cruzamento e busca local omitidas na restauração de sistemas devido a faltas podem ser utilizadas pois o tempo de simulação não são tão determinantes.

**Investigação das possíveis variações do algoritmo DE no algoritmo DE-Tree implementado:** O algoritmo DE-Tree implementado foi baseado na versão clássica do DE, ou seja, o vetor base escolhido randomicamente e uma única diferença vetorial. Outras estratégias passíveis de investigação citadas na literatura são as versões *best* e a *target-to-best*, como definições para o vetor base, e também o número de diferenças vetoriais utilizadas na equação da mutação diferencial.

# Apêndice A

# Exemplo Ilustrativo

Existem problemas estritamente de natureza combinatória nas áreas da engenharia e da computação que são problemas clássicos de difícil solução para muitos algoritmos de otimização. Entre estes, está o problema do caixeiro viajante, um problema de otimização universal, de natureza combinatória, que pode ser estendido a tantos outros problemas estritamente combinatórios do mundo real e, portanto, é o escolhido para exemplificar as diversas abordagens para evolução diferencial no domínio das variáveis discretas apresentada neste trabalho. Além disto, como as abordagens encontradas na literatura são aplicáveis tão somente em problemas combinatórios com permutação optou-se, por esta razão, por este tipo de problema.

O problema do caixeiro viajante TSP[1] consiste em encontrar o menor percurso que um caixeiro viajante pode fazer para percorrer todas as cidades que deve visitar, uma única vez, e retornar a sua cidade de origem. A Figura A.1 abaixo apresenta três rotas possíveis para um TSP constituído por 10 cidades. Nela, supõe-se que o caixeiro viajante parte da cidade de origem, rotulada com o algarismo **1**, percorre as demais nove cidades e retorna à cidade de origem.

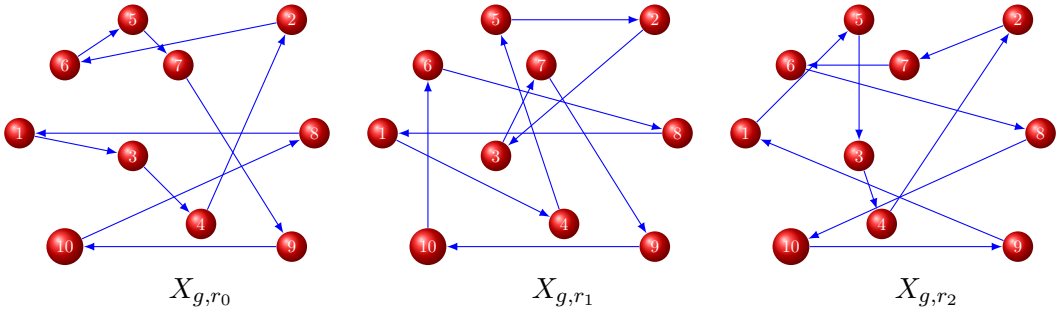



FIGURA A.1: Rotas TSP aleatórias para um problema com 10 cidades

[1] do inglês: *Travelling Salesman Problem*

A Equação A.1 apresenta a codificação dessas três rotas em um vetor com seus elementos rotulados com os números destas cidades. Dentre uma população de Np indivíduos (rotas possíveis), estas são as rotas escolhidas aleatoriamente para representar os três vetores da equação de mutação diferencial, respectivamente, $x_{g,r_0}$, $x_{g,r_1}$ e $x_{g,r_2}$.

$$x_{g,r_0} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 2 \\ 6 \\ 5 \\ 7 \\ 9 \\ 10 \\ 8 \end{bmatrix} \quad x_{g,r_1} = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 2 \\ 3 \\ 7 \\ 9 \\ 10 \\ 6 \\ 8 \end{bmatrix} \quad x_{g,r_2} = \begin{bmatrix} 1 \\ 5 \\ 3 \\ 4 \\ 2 \\ 7 \\ 6 \\ 8 \\ 10 \\ 9 \end{bmatrix} \tag{A.1}$$

## A.1 Indexação por Posição Relativa - IRP

O primeiro passo para implementar a abordagem IPR é transladar os vetores da equação da mutação diferencial para o domínio das variáveis reais dentro do intervalo $[0, 1]$ através da normalização dos seu elementos. A transformação dos parâmetros inteiros em reais é obtida dividindo-se cada parâmetro pelo maior deles, neste caso 10, conforme apresentado na Equação A.2:

$$x^{pf}_{g,r_0} = \frac{x_{g,r_0}}{10} = \begin{bmatrix} 0,1 \\ 0,3 \\ 0,4 \\ 0,2 \\ 0,6 \\ 0,5 \\ 0,7 \\ 0,9 \\ 1,0 \\ 0,8 \end{bmatrix}, \quad x^{pf}_{g,r_1} = \frac{x_{g,r_1}}{10} = \begin{bmatrix} 0,1 \\ 0,4 \\ 0,5 \\ 0,2 \\ 0,3 \\ 0,7 \\ 0,9 \\ 1,0 \\ 0,6 \\ 0,8 \end{bmatrix} \quad e \quad x^{pf}_{g,r_2} = \frac{x_{g,r_2}}{10} = \begin{bmatrix} 0,1 \\ 0,5 \\ 0,3 \\ 0,4 \\ 0,2 \\ 0,7 \\ 0,6 \\ 0,8 \\ 1,0 \\ 0,9 \end{bmatrix} \tag{A.2}$$

Onde:

O sobrescrito $pf$ denota a representação do vetor em ponto-flutuante.

Aplicando a equação da mutação diferencial, com $\mathbf{F} = 0,6$, obtém-se o vetor mutante, $v_{g,i}^{pf}$ dado por:

$$v_{g,i}^{pf} = \begin{bmatrix} 0,1 \\ 0,3 \\ 0,4 \\ 0,2 \\ 0,6 \\ 0,5 \\ 0,7 \\ 0,9 \\ 1,0 \\ 0,8 \end{bmatrix} + 0.6. \begin{bmatrix} 0,0 \\ -0,1 \\ 0,2 \\ -0,2 \\ 0,1 \\ 0,0 \\ 0,3 \\ 0,2 \\ -0,4 \\ -0,1 \end{bmatrix} = \begin{bmatrix} 0,10 \\ 0,24 \\ 0,52 \\ 0,08 \\ 0,66 \\ 0,50 \\ 0,88 \\ 1,02 \\ 0,76 \\ 0,74 \end{bmatrix} \tag{A.3}$$

A transformação de volta para o domínio dos inteiros é realizada atribuindo ao menor parâmetro em ponto-flutuante (0,08) o menor valor dos parâmetros inteiros (no exemplo a cidade de número 1) em seguida, ao segundo menor parâmetro em ponto-flutuante (0,10) é atribuído o segundo menor parâmetro inteiro (cidade 2) e assim sucessivamente como mostrado na Equação A.4:

$$v_{g,i}^{pf} = \begin{bmatrix} 0,10 \\ 0,24 \\ 0,52 \\ 0,08 \\ 0,66 \\ 0,50 \\ 0,88 \\ 1,02 \\ 0,76 \\ 0,74 \end{bmatrix} \rightarrow v_{g,i} = \begin{bmatrix} 2 \\ 3 \\ 5 \\ 1 \\ 6 \\ 4 \\ 9 \\ 10 \\ 8 \\ 7 \end{bmatrix} \tag{A.4}$$

Essa abordagem sempre resultará em soluções válidas, desde que os valores no domínio dos reais não sejam idênticos. Se isto ocorrer, o vetor mutante resultante deve ser reparado ou descartado. É fácil observar, também, que essa abordagem tão somente embaralha os parâmetros do vetor e uma mutação, idêntica àquela da mutação diferencial, não é obtida. Além disso, ela não é capaz de identificar a geração de rotas TSP idênticas [Price et al. 2005].

## A.2 Transformação Direta e Reversa - FBT

Para exemplificar essa abordagem, a Transformação Direta (apresentada na Equação 2.1 no Capítulo 3), utilizando um valor para $\alpha = 0.00001$ é aplicada aos vetores da equação da mutação diferencial para obter os seguintes vetores em ponto-flutuante:

$$x^{pf}_{g,r_0} = \begin{bmatrix} 10^{-5} \\ 2.00003 \\ 3.00004 \\ 1.00002 \\ 5.00006 \\ 4.00005 \\ 6.00007 \\ 8.00009 \\ 9.00010 \\ 7.00008 \end{bmatrix}, \quad x^{pf}_{g,r_1} = \begin{bmatrix} 10^{-5} \\ 3.00004 \\ 4.00005 \\ 1.00002 \\ 2.00003 \\ 6.00007 \\ 8.00009 \\ 9.00010 \\ 5.00006 \\ 7.00008 \end{bmatrix} \quad e \quad x^{pf}_{g,r_2} = \begin{bmatrix} 10^{-5} \\ 4.00005 \\ 2.00003 \\ 3.00004 \\ 1.00002 \\ 6.00007 \\ 5.00006 \\ 7.00008 \\ 9.00010 \\ 800009 \end{bmatrix} \tag{A.5}$$

Uma vez que os vetores estão representados por números em ponto-flutuante, a equação da mutação diferencial, Equação 1.4, com o escalar $\mathbf{F} = 0,6$, é utilizada para obter o vetor mutante, $v^{pf}_{g,i}$:

$$v^{pf}_{g,i} = \begin{bmatrix} 10^{-5} \\ 2.00003 \\ 3.00004 \\ 1.00002 \\ 5.00006 \\ 4.00005 \\ 6.00007 \\ 8.00009 \\ 9.00010 \\ 7.00008 \end{bmatrix} + 0.6. \left( \begin{bmatrix} 10^{-5} \\ 3.00004 \\ 4.00005 \\ 1.00002 \\ 2.00003 \\ 6.00007 \\ 8.00009 \\ 9.00010 \\ 5.00006 \\ 7.00008 \end{bmatrix} - \begin{bmatrix} 10^{-5} \\ 4.00005 \\ 2.00003 \\ 3.00004 \\ 1.00002 \\ 6.00007 \\ 5.00006 \\ 7.00008 \\ 9.00010 \\ 800009 \end{bmatrix} \right) = \begin{bmatrix} 10^{-5} \\ 1.400024 \\ 4.200052 \\ -0.199992 \\ 5.600066 \\ 4.00005 \\ 7.800088 \\ 9.200102 \\ 6.600076 \\ 6.400074 \end{bmatrix} \tag{A.6}$$

A Transformação Reversa, Equação 2.2, retorna o vetor mutante para o domínio dos números inteiros e, assim, obtém-se o seguinte vetor mutante:

$$v_{g,i}^{pf} = \begin{bmatrix} 10^{-5} \\ 1.400024 \\ 4.200052 \\ -0.199992 \\ 5.600066 \\ 4.00005 \\ 7.800088 \\ 9.200102 \\ 6.600076 \\ 6.400074 \end{bmatrix} \rightarrow v_{g,i} = \begin{bmatrix} 2 \\ 5 \\ 10 \\ 2 \\ 13 \\ 10 \\ 18 \\ 20 \\ 15 \\ 15 \end{bmatrix} \tag{A.7}$$

Como observado neste exemplo, esta abordagem geralmente produz rotas inválidas, ora devido a geração de rótulos inválidos para as cidades (como as cidades 15, 18 e 20 em um problema com 10 cidades), ora devido a geração de rótulos repetidos(cidade 2) e, portanto, necessita de um mecanismo de reparo adequado para estes casos. Como na abordagem anterior, esta também simplesmente embaralha os parâmetros do vetor mutante e uma mutação nos moldes da mutação diferencial não é observada. [Price et al. 2005] relata que, apesar dos bons resultados de trabalhos apresentados por Onwubolu usando essa abordagem, há motivos para acreditar que os sucessos obtidos são consequência direta dos mecanismos de reparos adotados e da prudência na escolha das heurísticas utilizadas.

## A.3 Matriz de Permutação - AMP

Aplicando essa abordagem aos vetores da mutação diferencial escolhidos como exemplo (Equação A.1), a matriz de permutação que relaciona $x_{g,r_1}$ e $x_{g,r_2}$ é definida como:

$$x_{g,r_2} = \mathbf{P}.x_{g,r_1} \quad com \quad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{A.8}$$

Definida a mutação diferencial na análise combinatória como sendo a matriz de permutação, o próximo passo agora é aplicar o fator escalar **F** à matriz. O Algoritmo 3 abaixo mostra um pseudocódigo, proposto por [Price et al. 2005], que define o como aplicar o escalar **F** à permutação diferencial.

**Algorithm 3** *Aplicação do Fator **F** à Matriz de Permutação - Pseudocódigo*

```
begin
    // Percorre todas as colunas da matriz P
    for (i = 1; i < N_p; i++) do
        // Se não existe nenhum elemento = 1 na diagonal
        if (p[i, i] == 0) then
            // Se um número aleatório entre (0, 1) é maior que F
            if (rand(0, 1) > F) then
                j ← 1;
                // encontre a linha onde p[j, i] = 1
                while (p[j, 1] ≠ 1) do
                    j++;
                troque_as_linhas (i,j);
end
```

Segundo esse código, escalonar a matriz de permutação consiste em percorrer cada coluna da matriz **P** a procura de elementos zeros na diagonal principal. Encontrando-se um zero na diagonal principal, e se um número sorteado aleatoriamente no intervalo [0,1] for maior que o fator **F**, procuram-se linhas onde exista, nesta coluna, elementos iguais à um. Encontrando-se um elemento 1 nesta posição trocam-se as linhas $(i, j)$.

É fácil verificar que, para um fator $\mathbf{F} = 1$ a matriz de permutação torna-se invariável, enquanto que um fator $\mathbf{F} = 0$ reduz a matriz à sua forma diagonal. Valores intermediários realizam uma fração da permutação definida pela matriz **P**.

Aplicando este algoritmo à matriz de permutação (Equação A.8) e, por simplicidade, supondo que somente o primeiro número aleatório sorteado é maior que o escalar **F**, somente uma troca será realizada. Assim, na $2^{a.}$ coluna encontra-se um elemento na diagonal principal que é igual a zero. Então, procurando-se linhas com elementos iguais a 1 nesta coluna encontra-se um elemento na posição $p[3, 2]$. Estas duas linhas são trocadas entre si e assim por diante. Com esse procedimento, a matriz escalonada **P'** resultante é dada por:

$$\mathbf{P'} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rightarrow 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rightarrow 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{A.9}$$

Finalmente, o vetor mutante $v_{g,i}$ é obtido aplicando a matriz de permutação resultante ao vetor base, $x_{g,r_0}$:

$$v_{g,i}^T = (\mathbf{P'}.x_{g,r_0})^T = \begin{bmatrix} 1 & 3 & 2 & 6 & 4 & 5 & 8 & 10 & 7 & 9 \end{bmatrix} \tag{A.10}$$

Segundo [Price et al. 2005], essa abordagem tende a se estagnar porque movimentos derivados de permutações são raramente produtivos. Além disso, esse método não é capaz de distinguir rotas TSP equivalentes[2]. Como vantagem pode-se dizer que essa abordagem sempre gera soluções válidas pois todas a operações são derivadas de permutações. Porém ela está limitada a ser aplicada tão somente nesta classe de problemas e, assim mesmo, utilizando apenas movimentos de troca para explorar o espaço de busca.

## A.4 Matriz de Adjacências - AMA

Nesta abordagem os vetores $x_{g,r_0}$, $x_{g,r_1}$ *e* $x_{g,r_2}$, utilizados para exemplificar a mutação diferencial, são representados pelas suas matrizes de adjacências, dadas por:

[2] a rota [1 2 3 4 5] é equivalente a rota [3 4 5 1 2]

$$x_{g,r_0} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{A.11}$$

$$x_{g,r_1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.12}$$

$$x_{g,r_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{A.13}$$

O vetor diferença da equação da mutação diferencial é escrito como:

$$M_{x_{g,r_1},x_{g,r_2}} = x_{g,r_1} \oplus x_{g,r_2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{A.14}$$

A matriz resultante da aplicação da equação da mutação diferencial claramente não é uma matriz de adjacências válida para uma percurso do TSP e, consequentemente, ao se adicionar esta matriz à matriz do vetor base $x_{g,r_0}$ resultará em uma matriz de adjacências inválida para o vetor mutante. Portanto, *matrizes mutantes*, matrizes de adjacências resultantes da aplicação da mutação diferencial, com percursos TSP inválidos devem ser reparadas para garantir que cada cidade seja conectada uma única vez a duas outras.

## A.5 Lista de Movimentos - LM

A abordagem por Lista de Movimentos determina que uma lista $M_{r_1r_2}$ de movimentos, apresentada na Definição 2.3, deve ser constituída por movimentos que levem a solução $x_{g,r2}$ à solução $x_{g,r1}$. A Figura A.2 mostra o primeiro passo para a construção desta lista para o TSP da Equação A.1 escritos na sua forma transposta. Nesta figura, supondo que os vetores iniciam-se com o índice zero, trocando-se as cidades de índices 1 e 3 em $x_{g,r_2}$, leva a solução $x_{g,r_2}$ a aproximar-se da solução $x_{g,r_1}$.

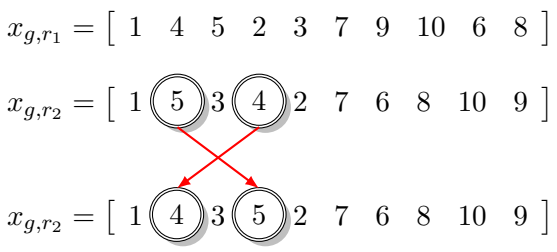



FIGURA A.2: Primeiro passo para construção da lista de movimentos $M_{r_1r_2}$

O primeiro movimento da lista, portanto, é formado pelo par de índices 1 e 3 conforme mostrado na Equação A.15:

$$M_{r_1 r_2} = [(1,3)] \tag{A.15}$$

O próximo movimento, mostrado na Figura A.3, é trocar as cidades nas posições de índices 2 e 3.

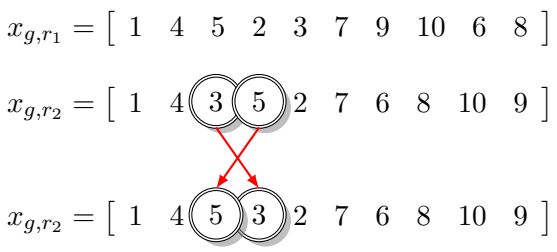



FIGURA A.3: Segundo passo para construção da lista de movimentos $M_{r_1 r_2}$

A lista de movimentos $M_{r_1 r_2}$ é, então, acrescida com o segundo movimento:

$$M_{r_1 r_2} = [(1,3)\,(2,3)] \tag{A.16}$$

Procedendo-se assim, ao final, uma lista composta por pares de índices cujas cidades foram trocadas é obtida conforme mostrado na Equação A.17 abaixo:

$$M_{r_1 r_2} = [(1,3)\,(2,3)\,(3,4)\,(6,9)\,(7,8)\,(8,9)] \tag{A.17}$$

Esta lista representa o "caminho" ou as modificações necessárias que devem ser realizadas na solução $x_{g,r_2}$ de maneira a obter a solução $x_{g,r_1}$. Ela também representa uma informação sobre as diferenças entre pares de soluções. Quando essa lista for aplicada a uma solução ela é lida, ou interpretada, como: primeiramente troque as cidades de índice 1 e 3. A seguir as cidades de índices 2 e 3. E assim por diante.

O próximo passo é aplicar o fator de multiplicação à matriz resultante da "diferença vetorial". Para um fator de multiplicação escalar $\mathbf{F} = 0,6$ e como o tamanho da lista $|M_{r_1 r_2}|$ é igual a 6, a lista de movimentos $M'_{r_1 r_2}$, utilizando a Definição 2.4 é dada por:

$$\begin{aligned} M'_{r_1 r_2} &= round(0,6 \times 6) = 4\ primeiros\ movimentos \\ M'_{r_1 r_2} &= [(1,3)\,(2,3)\,(3,4)\,(6,9)] \end{aligned} \tag{A.18}$$

Finalmente, os passos para obter o vetor mutante através da aplicação da Definição 2.5 são mostrados nas Figuras A.4 à A.7 utilizando a Definição 2.4 como a operação de multiplicação.

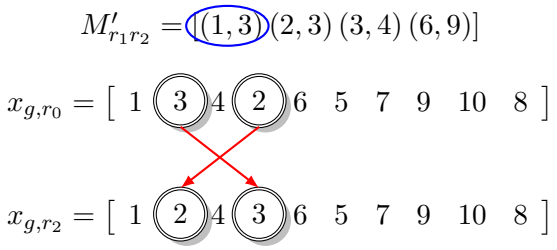



FIGURA A.4: Aplicação do $1^o$ movimento da lista ao vetor base: Trocar as cidades cujos índices são 1 e 3

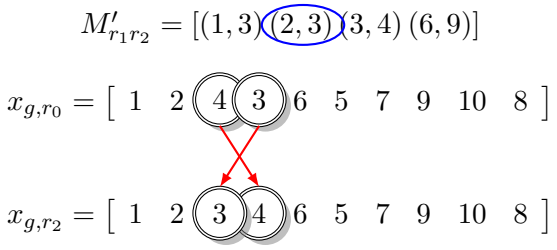



FIGURA A.5: Aplicação do $2^o$. movimento da lista ao vetor base: Trocar as cidades cujos índices são 2 e 3

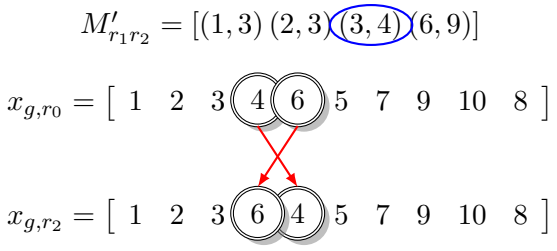



FIGURA A.6: Aplicação do $3^o$. movimento da lista ao vetor base: Trocar as cidades cujos índices são 3 e 4

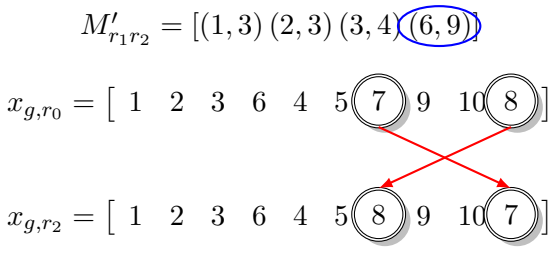



FIGURA A.7: Aplicação do $4^o$. movimento da lista ao vetor base: Trocar as cidades cujos índices são 6 e 9

Observa-se na Figura A.7 que nesta abordagem, o vetor mutante resultante, após a aplicação dos movimentos da lista, é sempre uma solução válida e não é necessário aplicar nenhuma rotina de reparo. Isto porque nenhuma operação aritmética foi realizada com os elementos dos vetores soluções utilizados na equação de mutação diferencial. Portanto,

usar rótulos numéricos ou literais para representar as cidades é transparente para o método e sempre resultará em vetores soluções válidos. Além disso a lista poderia empregar movimentos distintos, tais como o movimento *2opt*, não se restringindo a movimentos de troca.

# Referências Bibliográficas


L. S. Batista, F. G. Guimarães, and J. A. Ramírez. *A differential mutation operator for the archive population of multi-objective evolutionary algorithms*. 2009.

H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35, Nº3:238–308, September 2003.

N. G. Boulaxis and M. P. Papadopoulos. Optimal feeder routing in distribution system planning using dynamic programming technique and gis facilities. *Power Delivery, IEEE Transactions on*, 17(1):242–247, 2002.

E. M. Carreno, R. Romero, and A. Padilha-Feltrin. An efficient codification to solve distribution network reconfiguration for loss reduction problem. *Power Systems, IEEE Transactions on*, 23(4):1542–1551, 2008.

D. Das, H. Nagi, and D. Kothari. Novel method for solving radial distribution networks. *IEE Proceedings-Generation, Transmission and Distribution*, 141(4):291–298, 1994.

S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, 2011.

L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer., 2002.

K. Deb, J. Sundar, N. Udaya Bhaskara Rao, and S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, 2(3):273–286, 2006.

A. Delbem, A. de Carvalho, C. A. Policastro, A. K. Pinto, K. Honda, and A. C. Garcia. Node-depth encoding for evolutionary algorithms applied to network design. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 678–687. Springer, 2004.

A. C. Delbem, A. de Carvalho, and N. Bretas. Main chain representation for evolutionary algorithms applied to distribution system reconfiguration. *Power Systems, IEEE Transactions on*, 20(1):425–436, 2005.

E. Díaz-Dorado and J. C. Pidre. Optimal planning of unbalanced networks using dynamic programming optimization. *Power Systems, IEEE Transactions on*, 19(4): 2077–2085, 2004.

A. dos Santos. *Algoritmo Evolutivo Computacionalmente Eficiente para Reconfiguração de Sistemas de Distribuição*. PhD thesis, Universidade de São Paulo, 2009.

W. El-Khattam, Y. Hegazy, and M. Salama. An integrated distributed generation optimization model for distribution system planning. *Power Systems, IEEE Transactions on*, 20(2):1158–1165, 2005.

M. Farrag, M. El-Metwally, and M. El-Bages. A new model for distribution system planning. *International Journal of Electrical Power & Energy Systems*, 21(7):523–531, 1999.

V. Feoktistov. *Differential Evolution In Search of Solutions*. Springer Science + Business Media, 2006.

D. B. Fogel and L. J. Fogel. An introduction to evolutionary programming. In *Artificial Evolution*, pages 21–33. Springer, 1996.

F. G. Guimarães, R. C. P. Silva, R. S. Prado, O. M. Neto, and D. D. Davendra. Flow shop scheduling using a general approach for differential evolution. In *Handbook of Optimization*, pages 597–614. Springer, 2012.

F. G. Guimarães. *Anais do IX Congresso Brasileiro de Redes Neurais e Inteligência Computacional (IX CBRN/IC)*, chapter Algoritmos de Evolução Diferencial para Otimização e Aprendizado de Máquina, pages 1–17. 2009.

J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

J. R. Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science, 1990.

P. Larranaga and J. A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer, 2002.

D. Lichtblau. *Differential Evolution in Discrete and Combinatorial Optimization - Mathematica tutorial notes*. 2002. Último acesso em 23/04/2013.

M. Mansour, A. Santos, J. London, A. Delbem, and N. Bretas. Node-depth encoding and evolutionary algorithms applied to service restoration in distribution systems. In *Power and Energy Society General Meeting, 2010 IEEE*, pages 1–8. IEEE, 2010.

F. Mendoza, J. L. Bernal-Agustin, and J. A. Dominguez-Navarro. Nsga and spea applied to multiobjective design of power distribution systems. *Power Systems, IEEE Transactions on*, 21(4):1938–1945, 2006.

V.-R. J. Mezura-Montes, E. and C. A. C. Coello. *A comparative study of differential evolution variants for global optimization.* In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (ACM GECCO) (pp. 485-492). ACM Press., 2006.

G. C. Onwubolu and D. Davendra. *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization.* Springer Publishing Company, Incorporated., 2009.

P. Paiva, H. Khodr, J. Dominguez-Navarro, J. Yusta, and A. Urdaneta. Integral planning of primary-secondary distribution systems using mixed integer linear programming. *Power Systems, IEEE Transactions on*, 20(2):1134–1143, 2005.

R. S. Prado, R. C. P. Silva, F. G. Guimarães, and O. M. Neto. A new differential evolution based metaheuristic for discrete optimization. *International Journal of Natural Computing Research (IJNCR)*, 1(2):15–32, 2010a.

R. S. Prado, R. P. Silva, F. G. Guimarães, and O. M. Neto. Using differential evolution for combinatorial optimization: A general approach. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 11–18. IEEE, 2010b.

R. S. Prado, R. P. Silva, F. G. Guimarães, and O. M. Neto. Uma nova abordagem para a evolução diferencial em otimização discreta. In *In: XVIII Congresso Brasileiro de Automática (CBA)*, pages 830–836. CBA, 2010c.

R. S. Prado, R. P. Silva, O. M. Neto, F. G. Guimarães, D. S. Sanches, J. B. A. L. Junior, and A. C. B. Delbem. Differential evolution using ancestor tree for service restoration in power distribution systems. In *Journal Applied Soft Computing*, 2013.

K. V. Price. Differential evolution vs. the functions of the 2nd ICEO. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 153–157. IEEE, 1997.

K. V. Price and R. M. Storn. *Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces.* Journal of Global Optimization, 11(4), 341-359, December, 1997a.

K. V. Price and R. M. Storn. *Differential evolution: A simple evolution strategy for fast optimization.* Dr. Dobb's J, 22(4), 18-24, 1997b.

K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Pratical Approach to Global Optimization.* 1st ed., ser. Natural Computing Series, Berlin: Springer-Verlag, 2005.

A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE, 2005.

D. S. Sanches, J. B. A. L. Junior, A. C. B. Delbem, R. S. Prado, F. G. Guimarães, and O. M. Neto. Multiobjective evolutionary algorithm with differential mutation operator applied to service restoration in distribution systems. In *International Journal of Electrical Power and Energy Systems*, 2013.

A. Santos, A. Delbem, J. London, and N. Bretas. Node-depth encoding and multiobjective evolutionary algorithm applied to large-scale distribution system reconfiguration. *Power Systems, IEEE Transactions on*, 25(3):1254–1265, 2010.

A. C. Santos, A. Delbem, and N. G. Bretas. Energy restoration for large-scale distribution system using ea and a new data structure. In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pages 1–8. IEEE, 2008.

D. Shirmohammadi, H. Hong, A. Semlyen, and G. Luo. A compensation-based power flow method for weakly meshed distribution and transmission networks. *Power Systems, IEEE Transactions on*, 3(2):753–762, 1988.

M. Srinivas. Distribution load flows: a brief review. In *Power Engineering Society Winter Meeting, 2000. IEEE*, volume 2, pages 942–945. IEEE, 2000.

R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. technical report tr-95-012. Technical report, ICSI, 1995.

D. Sun, D. Farris, P. Cote, R. Shoults, and M. Chen. Optimal distribution substation and primary feeder planning via the fixed charge network formulation. *Power Apparatus and Systems, IEEE Transactions on*, (3):602–609, 1982.

F. Xue, A. Sanderson, and R. Graves. Pareto-based multi-objective differential evolution. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, pages 862–869. IEEE, 2003.