Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica

Algoritmos Evolucionários Eficientes para Otimização de Redes

Eduardo Gontijo Carrano

Tese de doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Ricardo Hiroshi Caldeira Takahashi

Co-orientador: Prof. Oriane Magela Neto

Belo Horizonte, Agosto de 2007

Resumo

Neste trabalho apresentam-se novas ferramentas voltadas à otimização de redes. Primeiramente são apresentadas abstrações de conceitos contínuos, capazes de gerar conceitos análogos no espaço discreto, onde as redes são definidas. Estes conceitos conferem ao espaço de redes ferramentas como representações espaciais, cálculos de posição relativa e distância e determinação de direções. Isso torna possível a implementação de técnicas geralmente só empregadas em problemas contínuos, como análises de dispersão, buscas locais, otimizações unidimensionais, etc, que podem ser incorporadas aos algoritmos de otimização através de operadores evolucionários. Estes operadores são utilizados como base para a construção de dois algoritmos: um Algoritmo Genético e um Algoritmo de Seleção Clonal. Estes algoritmos foram aplicados na solução de dois problemas clássicos, reconhecidamente complexos, e em dois problemas práticos. Além disso, são apresentados algoritmos específicos, voltados a três situações distintas do problema de projeto de sistemas de distribuição de energia elétrica: posicionamento de subestações associado ao projeto da topologia de redes, projeto multi-objetivo de redes de distribuição de energia e *scheduling* da expansão de sistemas de distribuição de energia. Estes algoritmos são baseados em operadores que são construídos tendo em conta as características específicas dos problemas tratados.

Os resultados obtidos mostram que as ferramentas desenvolvidas são úteis na solução de problemas de otimização em redes, sendo capazes de obter boas soluções em problemas dificilmente tratáveis por métodos tradicionais.

Abstract

This work presents new tools to help in the solution of network optimization problems. Firstly, some concepts of continuous spaces have been extended to the discrete space, where the networks are defined. These concepts grant to the network space interesting properties, such as spatial representations, relative position calculus, distance measurements and direction definitions. They make possible the implementation of techniques which are usually employed only in continuous problems, such as dispersion analysis, local searches, unidimensional optimizations, etc, which can be embedded in optimization algorithms through evolutionary operators. These operators are used to build two algorithms: a Genetic Algorithm and a Clonal Selection Algorithm. These algorithms have been employed for optimization of two classical problems which are known to be difficult, and two practical cases. Besides, three problem specific algorithms are presented, for three situations of distribution system design: joint facility location and network topology design, multi-objective design of distribution systems and system expansion scheduling. These algorithms are based on operators which have been built taking into account the characteristics of problem.

The results show that the developed tools are useful for network optimization, and that they can obtain good solutions in problems which could not be solved by traditional methods.

Agradecimentos

Antes de tudo agradeço a Deus por tudo que Ele tem proporcionado em minha vida, a minha mãe Enia, por tudo que fez por mim enquanto esteve ao meu lado e a meu pai Jorge, por toda compreensão, amizade, companheirismo e apoio irrestrito dedicados.

Agradeço aos meu orientadores Ricardo Takahashi, Oriane Magela e Carlos M. Fonseca, não só pela orientação recebida e tempo dedicado, mas também por todas as nossas longas discussões que sempre resultaram em boas idéias e são responsáveis por boa parte do conteúdo deste texto.

Agradeço à minha família, principalmente à Maria, Evelin, César, Manoel, Samuel e Estelita, pelo carinho e força e a minha namorada Danielle pelo amor, compreensão e paciência, principalmente nos momentos difíceis.

Agradeço aos meus amigos Frederico Gadelha, Rodrigo Cardoso, Luciano Pimenta, Edgar Pereira, Luís Epifânio, Alexandre Ramos, Ricardo Adriano, Cássia Nunes, Bruno Baeta e Éric Baeta pela ajuda ao longo desta pesquisa.

Agradeço a todos meus professores, principalmente Rodney Saldanha, Jaime Ramirez, José Raphael, Walmir Caminhas, Antônio Braga, João Vasconcelos e Élice Melo por terem feito parte de minha formação acadêmica e científica.

Por fim, agradeço às agências CAPES e CNPq pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

***AIS*:** sistema imunológico artificial

BFGS: Broyden-Fletcher-Goldfarb-Shanno

***Clonal*:** Algoritmo de Seleção Clonal

***ClonalNet*:** *Clonal* com inspiração contínua

***DP*:** Programação dinâmica

***DP-GA*:** *Dynamic Programming Genetic Algorithm*

***EA*:** algoritmo evolucionário

***GA*:** algoritmo genético

***GA-BFGS*:** algoritmo híbrido que combina um *GA* e um quasi Newton BFGS

***GANet*:** *GA* com inspiração contínua

Inc: expansão do sistema utilizando a abordagem incremental

***IBEA*:** *Indicator Based Evolutionary Algorithm*

***MOCSA*:** *Multi-objective Clonal Selection Algorithm*

***MOGA*:** *Multi-objective Genetic Algorithm*

***MOIA*:** *Multi-objective Immune Algorithm*

***MST*:** árvore geradora mínima

ndGA: expansão do sistema utilizando o algoritmo genético não-dinâmico

***NPGA*:** *Niched Pareto Genetic Algorithm*

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

***NSGA-PS*:** *NSGA-II for Power Systems*

***OCST*:** árvore de comunicação ótima

***PAES*:** *Pareto Archived Evolution Strategy*

***PESA*:** *Pareto Envelop-based Selection Algorithm*

***q-MST*:** árvore geradora quadrática mínima

***RWS*:** roleta estocástica

***SPEA*:** *Strength Pareto Genetic Algorithm*

***SS*:** subestações

***SUS*:** amostragem universal estocástica

***TS*:** seleção por torneio

***TSP*:** problema do caixeiro viajante

***VEGA*:** *Vector Enabled Genetic algorithm*

Capítulo 1

Projeto Ótimo de Redes

O projeto ótimo de redes pode ser definido como a tarefa de encontrar o conjunto de arestas que minimiza uma dada função de custo enquanto e conecta todos os vértices de um grafo, enquanto obedece uma estrutura pré-estabelecida. Estes problemas apresentam sérias dificuldades em sua solução, o que se deve principalmente à natureza combinatória dos mesmos (Pierre, 1993).

Essa natureza combinatória implica em um número muito grande de possíveis soluções, que cresce exponencialmente com a dimensão do problema. Além disso, a natureza discreta imposta por este tipo de problema geralmente causa severas descontinuidades no espaço onde as variáveis de decisão estão definidas, o que faz com que não exista um conjunto de soluções factíveis no qual se possa caminhar facilmente, sem saltos no espaço.

Estes aspectos restringem consideravelmente os métodos que podem ser aplicados no projeto ótimo de redes. A utilização de métodos determinísticos, voltados à otimização não-linear de espaços contínuos, se torna impossível, uma vez que os mesmos dependem de cálculos de derivadas que, por definição, não existem no espaço discreto onde as redes são definidas.

Técnicas que montam a árvore de possibilidades (também chamadas de métodos enumerativos), como o *Branch-and-Cut* ou o *Branch-and-Bound* (Grötschel and Holland, 1991; Vanderbei, 2001) têm como principal vantagem a garantia da obtenção do ótimo

do problema. No entanto, sua aplicação se torna inviável em parte dos casos práticos uma vez que alto computacional associado a estes métodos se torna um aspecto restritivo na solução de problemas de médio e grande porte a tempo viável.

Algoritmos polinomiais voltados à otimização de redes, como *Dijkstra*, *Kruskal* and *Ford-Fulkerson* (Dijkstra, 1959; Ahuja et al., 1993; Bazaraa et al., 1991) apresentam grande eficiência na solução dos problemas para os quais foram desenvolvidos (caminhos mínimos, árvore geradora mínima e fluxo máximo respectivamente) tendo entretanto sua aplicação limitada em outros contextos. Versões discretas do algoritmo *Simplex* Vanderbei (2001) também podem ser aplicadas à solução de redes, desde que todos os funcionais sejam lineares. A solução de problemas não-lineares utilizando estes métodos só é possível através de aproximações lineares que, em boa parte dos casos, podem levar a soluções finais que não condizem com o problema real.

A inaplicabilidade da maior parte das técnicas clássica motiva a busca de outros métodos para a solução deste problema. Os algoritmos evolucionários (*EAs*) se destacam como uma alternativa viável dentre esses métodos, uma vez que apresentam várias características favoráveis:

- Não dependem de premissas matemáticas fortes como linearidade, diferenciabilidade, convexidade ou unimodalidade;
- Podem ser aplicados à maior parte dos problemas práticos sem adaptações estruturais;
- Permitem adaptações da estrutura do algoritmo nos casos em que a estrutura tradicional se mostra insuficiente;
- São reconhecidamente eficientes na busca de ótimos globais em problemas multimodais (Duan and Yu, 2002).

Estes aspectos justificam a aplicação de algoritmos evolucionários em problemas de redes, como por exemplo:

Algoritmos genéticos: Os algoritmos genéticos (*GAs*) (Holland, 1975; Goldberg, 1989) vêm sendo aplicados à solução de vários problemas relacionados com redes, tais como:

- Projeto de redes de telecomunicações (Abuali et al., 1994);
- Projeto de redes de distribuição (Ramírez-Rosado and Bernal-Agustín, 1998);
- Projeto de redes de transmissão de energia (Duan and Yu, 2002; Chung et al., 2003);
- Roteamento de veículos (Ferentinos et al., 2002; Baker and Ayechew, 2003);
- Projeto de sistemas de trânsito urbano (Chakroborty, 2003), etc.

O autor da presente tese apresenta algumas contribuições no projeto de sistemas de distribuição de energia elétrica utilizando algoritmos genéticos:

- Projeto multi-objetivo utilizando um *GA* específico (Carrano et al., 2004, 2006b);
- Posicionamento de subestações associado ao projeto da topologia do sistema utilizando *GAs* híbridos (Carrano et al., 2005, 2007e)
- Planejamento do scheduling da expansão do sistema utilizando uma meta-heurística que associa um algoritmo genético a um método de programação dinâmica (Carrano et al., 2007a).

Sistemas imunológicos artificiais: A aplicação de sistemas imunológicos artificiais (*AIS*) (Dasgupta, 1999; de Castro and Timis, 2002; Dasgupta et al., 2003) na solução de problemas de projeto ótimo de redes se encontra restrita a poucas aplicações:

- Solução do problema do caixeiro viajante (do inglês *Traveling Salesman Problem* ou *TSP*) (de Castro and Von Zuben, 2002);
- Problemas de roteamento (Keko et al., 2003);

- Navegação de robôs (Luh and Liu, 2004).

As poucas referências encontradas na bibliografia são provavelmente justificadas pela estrutura destes algoritmos, que é dependente de uma definição adequada de distância. O autor da tese emprega estes algoritmos no projeto de sistemas de distribuição de energia elétrica considerando incertezas na evolução de carga (Carrano et al., 2007c). A implementação proposta é baseada na métrica proposta por Carrano et al. (2007f), capaz de determinar a distância entre redes.

***Ant algorithms*:** A estrutura discreta intrínseca dos *Ant algorithms* (Dorigo, 1992; Dorigo et al., 1996) favorece a utilização destes algoritmos na solução de problemas de redes. Apesar de recentes, estes algoritmos vêm sendo aplicados em vários problemas relacionados:

- Solução do *TSP* (Dorigo and Gambardella, 1997);
- Roteamento de redes *wireless ad hoc* (Güneş and Bouazizi, 2002);
- Projeto de sistemas de distribuição de água (Zecchin et al., 2003);
- Roteamento de veículos (Bell and McMullen, 2004);
- Projeto de redes de distribuição de energia elétrica (Gómez et al., 2004).

Estes algoritmos são utilizados nesta tese para comparação com os resultados obtidos pelos algoritmos apresentados no Cap. 4, quando utilizados para solução de um sistema de distribuição de energia elétrica.

As contribuições do autor citadas acima compõem a maior parte dos resultados apresentados neste trabalho. Os Caps. 4 e 5 apresentam os principais resultados obtidos nestas referências.

Na seqüência deste capítulo é apresentado o conceito de grafo, fundamental para a formulação do problema de projeto ótimo de redes.

1.1 Grafos

O conceito de grafo é essencial para o entendimento do problema de projeto ótimo de redes, uma vez que elas são modeladas por grafos. Um grafo G consiste em um conjunto finito V de vértices ($|V| = n$), um conjunto finito A de arestas ($|A| = m$), e uma matriz de adjacência M, que associa a cada aresta a de G um par não ordenado de vértices (não necessariamente distintos) de G, chamados de extremos de a (Bondy and Murty, 1976; Wilson, 1996).

As redes tratadas ao longo deste trabalho são modeladas por grafos planares, com arestas não-direcionadas e sem realimentação dos nós. Essas propriedades definem a matriz de adjacência que descreve o grafo completo (G_C), como apresentado em (1.1).

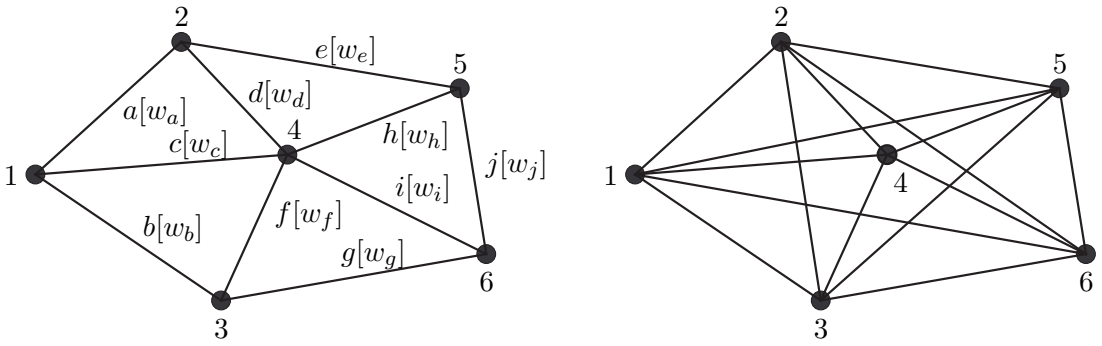
$$M_{G_C}[i,j] = \begin{cases} 0 & \text{se} \quad i = j \\ 1 & \text{se} \quad i \neq j \end{cases} \qquad i,j = 1,\ldots,n \tag{1.1}$$

Neste caso, fica claro que o número de arestas presentes no grafo completo é definido por:

$$m = \sum_{i=1}^{n-1} i = \frac{n \cdot (n-1)}{2}. \tag{1.2}$$

A Fig. 1.1(a) apresenta um exemplo de grafo não-direcional, $G_1(V_{G_1}, A_{G_1})$, com seis vértices ($V_{G_1} = [1,2,3,4,5,6]$) e 10 arestas ($A_{G_1} = [a,b,c,d,e,f,g,h,i,j]$) com pesos $W_{G_1} = [w_a, w_b, w_c, w_d, w_e, w_f, w_g, w_h, w_i, w_j]$ respectivamente. Já a Fig. 1.1(b) apresenta o grafo completo para o conjunto de vértices do grafo da Fig. 1.1(a), que é um sub-grafo de G_C ($G_1 \subset G_C$).

Uma estrutura particular de grafos, as árvores, é discutida de forma detalhada na próxima seção. Esta estrutura constitui o foco principal deste trabalho.



(a) Grafo com 6 vértices e 10 arestas (b) Grafo completo

Figura 1.1: Exemplo de grafo

1.1.1 Árvores

As árvores representam umas das classes mais importantes de grafos, uma vez que muitos problemas práticos apresentam estrutura de árvore ou podem ser modelados por elas. Outros sistemas, embora não envolvendo redes com estrutura de árvore, envolvem árvores em certas etapas do seu projeto. Um grafo representa uma árvore se obedece a seguinte definição:

Definição 1.1 *Um grafo $G(V, A)$ é uma árvore se, e somente se, G é um grafo conexo*[1] *que não contém ciclos.*

Dessa definição, derivam alguns teoremas importantes (Narsingh, 1984):

Teorema 1.1 *Existe um, e apenas um, caminho entre qualquer par de vértices em uma árvore.*

Teorema 1.2 *Em uma árvore com n vértices existem $n-1$ arestas.*

A demonstração desses teoremas é simples, e pode ser encontrada em Narsingh (1984). Destes teoremas percebe-se que a remoção, ou adição, de uma aresta ao grafo faz com que o mesmo deixe de ser uma árvore: a remoção de uma aresta faz com que

[1]Um grafo é dito conexo se existe ao menos um caminho entre qualquer par de vértices do grafo.

o grafo deixe de ser conexo, enquanto que, a adição de uma aresta insere um ciclo no grafo.

Outro teorema importante sobre árvores é apresentado por Cayley (1889):

Teorema 1.3 (Teorema de Cayley) *Em um grafo completo G_C, com n vértices e $m = \frac{n(n-1)}{2}$ arestas, existem n^{n-2} árvores que são sub-grafos de G_C.*

O Teorema de Cayley dá uma noção sobre a dimensão do espaço de busca do problema, que cresce exponencialmente com o número de nós.

Uma árvore particular, que tem grande importância no contexto de otimização, é a árvore geradora mínima (do inglês *Minimum Spanning Tree* ou *MST*), discutida na seqüência (Even, 1989).

Árvore Geradora Mínima

Um grafo $G'(V', A')$ é chamado um sub-grafo de um grafo $G(V, A)$, se $V' \subseteq V$ e $A' \subseteq A$. Claramente, uma escolha arbitrária de $V' \subseteq V$ e $A' \subseteq A$ pode não produzir um sub-grafo, simplesmente porque isto pode não ser um grafo: alguns dos vértices que são extremos de A' podem não estar contidos em V'.

Seja $G(V, A)$ um grafo conexo e suponha que cada aresta $a \in A$ tem peso conhecido $w(a) > 0$. Deseja-se encontrar um sub-grafo conexo $G'(V, A')$ cujo peso total $\sum_{a \in A} w(a)$ é mínimo; ou, em outras palavras, deseja-se remover de G o sub-conjunto de arestas cujo peso total é máximo, e que ainda deixa o grafo conexo. Fica claro que esse sub-grafo não possui ciclos, uma vez que possui peso total mínimo. Como G' é conexo, pelo Teorema 1.1, pode-se constatar que G' é uma árvore, não podendo portanto ser removida nenhuma aresta sem destruir a conectividade do grafo. O sub-grafo de G que contém todos os vértices de G e é uma árvore é chamado de árvore geradora de G (a Fig. 1.2 apresenta um exemplo de árvore geradora para o grafo da Fig. 1.1(a)). A árvore geradora que possui menor peso total é chamada de árvore geradora mínima.

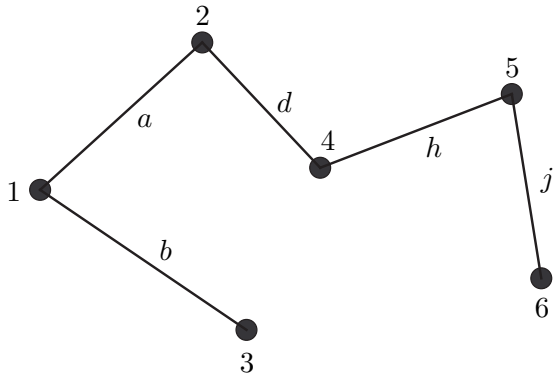


Figura 1.2: Exemplo de árvore geradora para o grafo da Fig. 1.1(a)

1.2 Representação das Variáveis

O espaço de busca do problema de projeto de redes é definido pelo sub-conjunto de sub-grafos de G_C que obedecem a estrutura de interesse (no caso deste trabalho, busca-se sub-grafos de G_C que são árvores). Com isso, cada aresta do grafo completo representa uma variável de decisão do problema de otimização.

Tendo em conta estes aspectos, pode-se representar as soluções do problema de otimização através de um vetor X composto por m variáveis binárias que indicam a presença ou ausência de cada uma das arestas possíveis, como ilustrado em (1.3).

$$\begin{array}{rcccccccccc} & \text{de} & 1 & 1 & \ldots & 2 & 2 & \ldots & n_{n-1} & & \\ & \text{para} & 2 & 3 & \ldots & 3 & 4 & \ldots & n_n & x_i \in [0,1] & \\ X & = & [\ x_1 & x_2 & \ldots & x_k & x_{k+1} & \ldots & x_m &] & \end{array} \tag{1.3}$$

Entretanto esta representação binária não é suficiente em todos os casos, uma vez que, em parte dos problemas práticos, cada aresta pode assumir tipos distintos (problemas *multi-branch*). Isso ocorre por exemplo em redes de energia elétrica, onde se pode utilizar condutores com bitolas diferentes, ou em redes de água, onde geralmente são disponibilizados tubos com diâmetros distintos. Uma forma clara de contornar este problema é a utilização de vários bits para representar cada conexão possível.

Uma alternativa mais compacta para estes problemas é a utilização de inteiros para representar o tipo de conexão utilizado (Ramírez-Rosado and Bernal-Agustín, 1998). Nessa representação cada variável de decisão x_i pode assumir valores inteiros que variam de 0 (ausência de conexão) a b (nós conectados com uma conexão de tipo b, onde b é o número de tipos de conexões possíveis). A Eq. (1.4) ilustra essa representação.

$$\begin{array}{rcccccccccc} & de & 1 & 1 & \ldots & 2 & 2 & \ldots & n_n - 1 & & \\ & para & 2 & 3 & \ldots & 3 & 4 & \ldots & n_n & x_i \in & [0, 1, \ldots, b] \\ X & = & [\ x_1 & x_2 & \ldots & x_k & x_{k+1} & \ldots & x_{nc} & \ ] & \end{array} \tag{1.4}$$

1.3 Formulação Geral do Problema de Projeto Ótimo de Redes

Tendo como base os conceitos descritos anteriormente e sendo $f_c(X)$ uma função X que se deseja minimizar, pode-se definir a seguinte formulação geral para o problema de otimização de redes:

$$\begin{aligned} X^* &= \arg\min_X f_c(X) \\ \text{sujeito a} &: X \in \mathcal{F}_X \end{aligned}$$

onde $\mathcal{F}_X$ é o conjunto de redes factíveis.

No Cap. 3 são discutidas técnicas de otimização potencialmente capazes de resolver este problema, tendo em vista as dificuldades já citadas.

Na seqüência, serão discutidas as formulações de três problemas clássicos tratados ao longo desta tese: árvore geradora mínima, árvore de comunicação ótima (do inglês *Optimal Communication Spanning Tree* ou *OCST*) e árvore geradora quadrática mínima (do inglês *Quadratic Minimum Spanning Tree* ou *q-MST*.

1.3.1 Árvore Geradora Mínima - *MST*

Como já foi citado na Seção 1.1.1, o problema de encontrar a árvore geradora mínima consiste da busca da árvore geradora na qual a soma dos pesos das arestas é mínima. Este problema apresenta a seguinte definição formal (Lin and Gen, 2006):

Seja $G(V, A)$ um grafo conexo, não direcionado e seja $w_{i,j} \in W$ o peso associado à aresta que conecta o par de vértices (i, j). Sejam $T \subseteq A$ e S os vértices induzidos por T (i.e., S é o conjunto de vértices que são extremos das arestas de T). O problema *MST* pode ser definido como buscar o conjunto T que minimiza:

$$T^* = \arg\min_{T} \sum_{i,j \in V} w_{i,j} \cdot t_{i,j} \tag{1.5}$$

$$\text{sujeito a:} \begin{cases} \sum_{i,j \in V} t_{i,j} & = & |V| - 1 \\ \sum_{i,j \in V} t_{i,j} & \leq & |S| - 1 \\ t_{i,j} \in \{0, 1\} & , & \forall i, j \in V \end{cases} \tag{1.6}$$

onde $(i, j) \in T$ se, e somente se $t_{i,j} = 1$.

Como exemplo, para este problema o custo da árvore T apresentada na Fig. 1.2 é:

$$C_T = w_a + w_b + w_d + w_h + w_j$$

A solução do problema *MST* não constitui uma tarefa computacional complexa, uma vez que o mesmo pode ser resolvido em tempo polinomial utilizando algoritmos clássicos, como Kruskal (Kruskal, 1956) ou Prim (Prim, 1957). No entanto, muitas variantes do problema *MST* são computacionalmente complexas. Dentre elas se destacam o *OCST* e o *q-MST*, para os quais ainda não existem algoritmos polinomiais exatos (Soak et al., 2006).

1.3.2 Árvore de Comunicação Ótima - *OCST*

O problema de encontrar árvore de comunicação ótima consiste na busca da árvore geradora de mínimo custo que cumpre com requisitos de comunicação previamente conhecidos entre os pares de vértices do grafo (Hu, 1974; Soak et al., 2006). O *OCST* foi inicialmente provado como *NP-hard* (Garey and Johnson, 1979) e posteriormente foi provado ser *MAX SNP-hard* (Arora et al. (1998) mostram que não existem métodos aproximados com custo polinomial para solução de problemas *MAX SNP-hard* com $NP \neq P$.). O problema apresenta a seguinte formulação:

Seja $R_{i,j}$ a demanda de comunicação existente entre (i,j) e $C^X_{i,j}$ o peso acumulado do caminho (i,j) em T[2]. A formulação apresentada para o *MST* (Eqs. (1.5) e (1.6)) pode ser estendida para *OCST* apenas substituindo-se (1.5) por (1.7).

$$T^* = \arg\min_{T} \sum_{i,j \in V} R_{i,j} \cdot C^X_{i,j} \tag{1.7}$$

Para este problema, o custo da árvore T apresentada na Fig. 1.2 é:

$$\begin{aligned} C_T \;=\; & R_{1,2}\; w_a + R_{1,3}\; w_b + R_{1,4}\; (w_a + w_d) + R_{1,5}\; (w_a + w_d + w_h) + \\ & R_{1,6}\; (w_a + w_d + w_h + w_j) + R_{2,3}\; (w_a + w_d) + R_{2,4}\; w_d + \\ & R_{2,5}\; (w_d + w_h) + R_{2,6}\; (w_d + w_h + w_j) + R_{3,4}\; (w_b + w_a + w_d) + \\ & R_{3,5}\; (w_b + w_a + w_d + w_h) + R_{3,6}\; (w_b + w_a + w_d + w_h + w_j) + \\ & R_{4,5}\; w_h + R_{4,6}\; (w_h + w_j) + R_{5,6}\; w_j \end{aligned}$$

1.3.3 Árvore Geradora Quadrática Mínima - *q-MST*

O problema de encontrar a árvore geradora quadrática mínima, inicialmente proposto por Assad and Xu (1992), consiste da busca pela árvore geradora que minimiza uma função quadrática, que depende dos pesos das arestas e da interação entre as mesmas (Soak et al., 2006). Este problema também foi provado como *NP-hard*

[2] O peso acumulado do caminho (i,j) é definido pela soma dos pesos de todas as arestas presentes no caminho único que conecta o nó i ao nó j na árvore T.

(Assad and Xu, 1992) e apresenta a seguinte formulação:

Seja $w_{i,j}^{k,l}$ o peso induzido na aresta (i,j) pela aresta (k,l). Mais uma vez pode-se extender a formulação do *MST* (Eqs. (1.5) e (1.6)) para o *q-MST* substituindo a função objetivo (1.5) por (1.8).

$$T^* = \arg\min_{T} \sum_{i,j \in V} \left(\sum_{k,l \in V} w_{i,j}^{k,l} \cdot t_{i,j} \cdot t_{k,l} \right) + w_{i,j} \cdot t_{i,j} \tag{1.8}$$

Neste caso, o custo da árvore T (Fig. 1.2) é:

$$\begin{aligned} C_T \;=\; & w_a + w_a^b + w_a^d + w_a^h + w_a^j + w_b + w_b^a + w_b^d + w_b^h + w_b^j + \\ & w_d + w_d^a + w_d^b + w_d^h + w_d^j + w_h + w_h^a + w_h^b + w_h^d + w_h^j + \\ & w_j + w_j^a + w_j^b + w_j^d + w_j^h \end{aligned}$$

1.4 Modelagem de Sistemas Reais Utilizando Redes

Vários problemas de projeto em engenharia são caracterizados por estruturas de redes. Devido a isso, algumas estruturas de grafos são utilizadas na modelagem destes problemas:

Redes com redundância: Em problemas de projeto geralmente se buscam grafos conexos, sendo que as árvores se caracterizam como grafos conexos mínimos. No entanto, em alguns problemas práticos deseja-se que a rede apresente alternativas de conexão em todos, ou na maior parte, dos vértices (deseja-se que exista mais de um caminho entre dois extremos a e b). Nestes casos são inseridas arestas que criam redundâncias, dando aos vértices do grafo alternativas de alimentação. Redes de gás, telecomunicações, transmissão de energia e computadores são problemas práticos que aplicam essa estrutura. A Fig. 1.3 mostra um dos possíveis grafos com redundância que podem ser obtidos do grafo apresentado na Fig. 1.1(a).

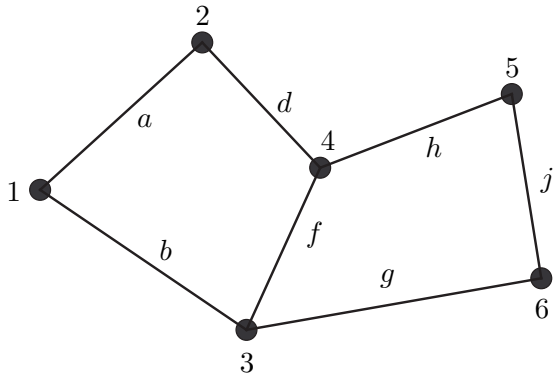


Figura 1.3: Exemplo de grafo com redundância

Ciclos de Hamilton: Outra classe relevante de grafos tem a estrutura de ciclos de Hamilton. Seja G um grafo conexo, um ciclo de Hamilton de G é definido como um caminho fechado que atravessa cada vértice uma única vez, exceto o vértice de partida do curso, que é atravessado duas vezes. Os problemas mais clássico que empregam essa estrutura de grafos são o *TSP* e problemas de *Scheduling*. Variantes do *TSP* também empregam este tipo de estrutura, como planejamento de limpeza urbana e planejamento de rotas aéreas e terrestres. A Fig. 1.4 mostra um dos possíveis ciclos de Hamilton que podem ser obtidos do grafo apresentado na Fig. 1.1(a).

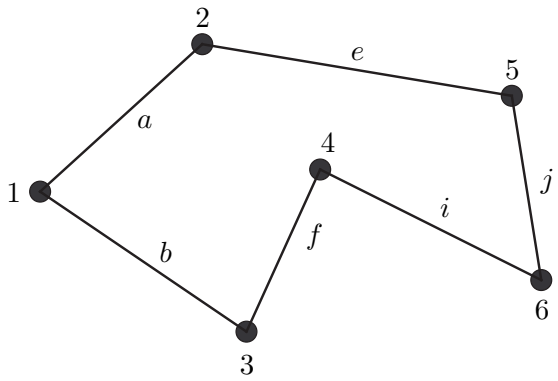


Figura 1.4: Exemplo de ciclo de Hamilton

Redes em árvore: Como discutido neste capítulo, as árvores representam uma das classes mais importantes de grafos. A estrutura de uma árvore é discutida na seção 1.1.1. Dentre os problemas modelados por redes em árvore, pode-se citar projeto de redes de distribuição de água, projeto de redes de distribuição de energia elétrica e projeto de redes de sensores sem fio. Uma das possíveis árvores obtidas do grafo apresentado na Fig. 1.1(a) é apresentada na Fig. 1.2.

Capítulo 2

Formulação Geral de Problemas de Otimização Contínua e Algoritmos Determinísticos

Ao longo deste capítulo são apresentadas as formulações gerais de problemas de otimização contínua para casos mono e multi-objetivo. Também são discutidos algoritmos capazes de resolver estes problemas. Estas formulações e algoritmos não são diretamente aplicáveis a problemas de redes, devido à característica combinatória dos mesmos. No entanto, o estudo destes tópicos constitui uma parte importante deste trabalho, uma vez que parte dos métodos desenvolvidos aqui são construídos com inspiração em abordagens contínuas. Além disso, no Cap. 5 são apresentadas aplicações destes algoritmos, que são executados juntamente com algoritmos evolucionários para posicionamento de nós e projeto da topologia da rede simultâneos.

2.1 Formulação Geral de Problemas de Otimização Contínua

2.1.1 Caso Mono-objetivo

Seja $x \in \mathbb{R}^n$ o vetor de parâmetros que devem ser escolhidos em um determinado problema, e $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função que avalia a adequação de cada vetor x. Pode-se

definir o problema de otimização como:

$$x^* = \arg\min_x f(x) \tag{2.1}$$

ou seja, busca-se o valor de x para o qual $f(x)$ é mínimo.

No entanto, na maior parte dos problemas também é necessário o atendimento de restrições relacionadas ao problema. Para atendimento dessas restrições, definem-se regiões no espaço de parâmetros $\mathbb{R}^n$ através de desigualdades e igualdades:

$$\begin{aligned} g_i(x) \leq 0 \quad \forall \quad i = 1, \ldots, r \\ h_j(x) = 0 \quad \forall \quad j = 1, \ldots, p \end{aligned} \tag{2.2}$$

Portanto, o problema de otimização restrita, em um contexto geral, apresenta a seguinte formulação:

$$x^* = \arg\min_x f(x) \tag{2.3}$$

$$\text{sujeito a:} \begin{cases} g_1(x) & \leq & 0 \\ & \vdots & \\ g_r(x) & \leq & 0 \\ h_1(x) & = & 0 \\ & \vdots & \\ h_p(x) & = & 0 \end{cases} \tag{2.4}$$

As características da função objetivo e do conjunto de restrições do problema usualmente definem o método a ser utilizado em sua solução. Uma característica muito importante para a escolha desses algoritmos é a modalidade.

Modalidade de Funções

Com base nas definições apresentadas no Ap. A, pode-se definir uma categorização importante para as funções (Takahashi, 2004):

Definição 2.1 (Função Unimodal) *Seja* $f(\cdot) : C \subset \mathbb{R}^n \mapsto \mathbb{R}$. *Diz-se que* $f(\cdot)$ *é unimodal se* $R(f,\alpha)$[1] *é conexo para todo* $\alpha \in \mathbb{R}$. *Diz-se ainda que* $f(\cdot)$ *é estritamente unimodal se, além disso,* $R(f,\alpha)$ *é um conjunto compacto para todo* $\alpha \in \mathbb{R}$.

Por simetria, define-se ainda:

Definição 2.2 (Função Multimodal) *Seja* $f(\cdot) : C \subset \mathbb{R}^n \mapsto \mathbb{R}$. *Diz-se que* $f(\cdot)$ *é multimodal se existe* $\alpha \in \mathbb{R}$ *tal que* $R(f,\alpha)$ *não é conexo.*

Uma função unimodal geralmente apresenta um único mínimo. No entanto, é importante destacar que uma função unimodal pode possuir múltiplos mínimos, desde que o conjuntos destes forme um conjunto conexo. Uma função estritamente unimodal também pode assumir múltiplos mínimos, desde que o conjunto destes forme um conjunto conexo compacto (Takahashi, 2004). As funções multimodais possuem múltiplos mínimos em todos os casos.

2.1.2 Caso Multi-objetivo

Seja $x \in \mathbb{R}^n$ o vetor de parâmetros que devem ser escolhidos em um problema multi-objetivo. Seja ainda $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ o vetor de m funções objetivo desse problema. O conjunto $\mathcal{X}^*$ das soluções eficientes descrito por:

$$\mathcal{X}^* = \arg\min_x \begin{cases} f_1(x) \\ \vdots \\ f_m(x) \end{cases} \tag{2.5}$$

[1] $R(f,\alpha)$ é a região de sub-nível de f associada ao nível α. As definições formais de conjuntos e curvas de nível podem ser encontradas no Ap. A.

$$\text{sujeito a:} \begin{cases} g_1(x) & \leq & 0 \\ & \vdots & \\ g_r(x) & \leq & 0 \\ h_1(x) & = & 0 \\ & \vdots & \\ h_p(x) & = & 0 \end{cases} \tag{2.6}$$

fica definido univocamente.

Seja $\mathcal{F}_x$ a região na qual o problema se encontra restrito (definida pelo conjunto de restrições apresentado em (2.6)). Em geral, não existe um único ponto $x \in \mathcal{F}_x$ em que $f(\cdot)$ atinge valor mínimo para todas as funções. Então:

$$\mathcal{X}^* = \{x^* \in \mathcal{F}_x \mid \nexists\ z \in \mathcal{F}_x \text{ onde } f(z) \leq f(x^*) \text{ e } f(z) \neq f(x^*)\} \tag{2.7}$$

onde os operadores relacionais $\leq$ e $\neq$ são definidos para vetores $u, v \in \mathbb{R}^m$, tal que:

$$\begin{aligned} u \leq v &\Leftrightarrow u_i \leq v_i \quad \forall \quad i = 1, \ldots, m \\ u \neq v &\Leftrightarrow \exists i \mid u_i \neq v_i \quad , \quad i = 1, \ldots, m \end{aligned} \tag{2.8}$$

Os pontos $x \in \mathcal{F}_x$ que não pertencem a $\mathcal{X}$ são chamados de *dominados*, uma vez que existem outros pontos $z \in \mathcal{F}_x$, tal que $f(z) \leq f(x)$ e $f(z) \neq f(x)$, o que significa que $f(z)$ é melhor que $f(x)$ em ao menos uma coordenada sem ser pior em nenhuma outra. Neste caso, diz-se que z *domina* x. As soluções x^* que pertencem ao conjunto $\mathcal{X}^*$, são chamadas de *soluções eficientes*, já que não são dominadas por nenhum outro ponto. Portanto, na otimização multi-objetivo busca-se um conjunto de soluções eficientes para um problema de otimização de um vetor de funções. Este conjunto, $\mathcal{X}^*$, é chamado de conjunto de Pareto, e define a fronteira Pareto, $\mathcal{Y}^*$, no espaço de objetivos. Um exemplo de fronteira Pareto, para um problema contínuo com duas funções objetivo, é apresentado na Fig. 2.1.

O mapeamento completo do conjunto de Pareto, ou de parte significativa dele,

oferece suporte para decisão em problemas de projeto, uma vez que a posição relativa das soluções na fronteira Pareto permite uma avaliação de compromisso dentre os objetivos tratados. Com isso, o projetista pode avaliar o efeito de substituir uma solução por outra, tendo em vista a perda em um objetivo com o simultâneo ganho em outro (ou outros). A abordagem mono-objetivo não permite esse tipo de análise.

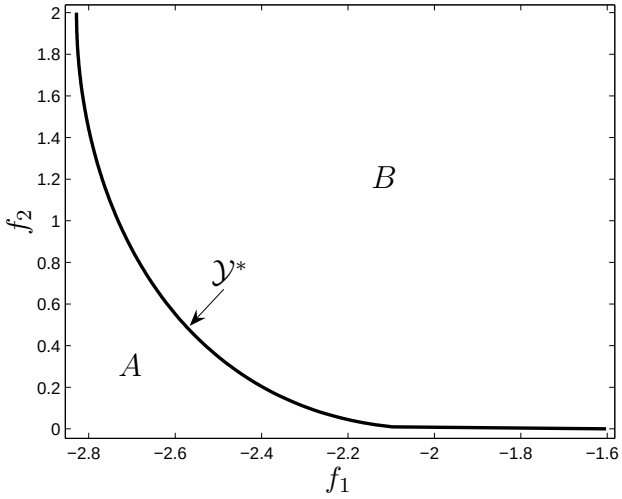


Figura 2.1: Pode-se perceber que não existe dominância entre as soluções do conjunto $\mathcal{Y}^*$. As soluções da região A estão fora do conjunto imagem ($\mathcal{I}$) de $f(\cdot)$, não podendo portanto ser alcançadas. Já as soluções da região B pertencem à $\mathcal{I}$, mas estão fora de $\mathcal{Y}^*$, pois cada uma delas é dominada por ao menos uma solução de $\mathcal{Y}^*$.

2.2 Algoritmos Determinísticos para Problemas Mono-objetivo

Nesta seção é apresentado o algoritmo quasi-Newton BFGS (Broyden-Fletcher-Goldfarb-Shanno), utilizado para otimização de funções não-lineares contínuas.

2.2.1 Algoritmo quasi-Newton BFGS

Introdução

Os algoritmos de direção de busca, em sua maioria, têm como suporte a mesma idéia básica:

1. Escolher uma direção na qual se espera que seja possível decrescimento da função;
2. Encontrar um ponto sobre essa direção na qual a função assume valor mínimo (otimização unidimensional);
3. Definir o ponto encontrado como ponto corrente e repetir o processo até que se atinja um critério de convergência pré-estabelecido.

Uma escolha óbvia para a direção em que será realizada a busca unidimensional é o vetor que anula o vetor gradiente no ponto corrente (isto é válido em problemas de minimização).

Nas décadas de 50 e 60 foram propostas melhorias desse método, utilizando informações da curvatura da função para correção da direção de busca. Essas informações de curvatura são estimadas através de algum método de aproximação da inversa da matriz Hessiana da função objetivo. No método quasi-Newton BFGS, a aproximação da inversa da matriz Hessiana é definida por:

$$\begin{aligned} r_k &= x_{k+1} - x_k \\ v_k &= \nabla f(x_{k+1}) - \nabla f(x_k) \\ H_{k+1} &= H_k + \left(1 + \frac{r_k' \cdot H_k \cdot r_k}{r_k' \cdot v_k}\right) \cdot \frac{v_k \cdot v_k'}{v_k' \cdot v_k} - \frac{v_k \cdot r_k' \cdot H_k + H_k \cdot r_k \cdot v_k'}{r_k' \cdot v_k} \end{aligned} \tag{2.9}$$

onde:

x_k é o ponto avaliado na iteração k;

$\nabla f(x_k)$ é o gradiente da função objetivo no ponto x_k;

H_k é a aproximação da inversa da matriz Hessiana calculada na iteração k.

Na prática, as Eqs. (2.9) fazem uma aproximação iterativa da inversa da Hessiana, porém com um custo computacional muito inferior ao cálculo exato da mesma, uma vez que não é necessário nenhum cálculo adicional de função objetivo, e nem a inversão da matriz. Informações adicionais sobre métodos de direção de busca, incluindo outros métodos que utilizam aproximações da inversa da Hessiana, podem ser encontradas em Takahashi (2004); Bazaraa et al. (1993).

Algoritmo

Uma descrição geral do algoritmos BFGS é apresentada abaixo.

Estrutura Básica do Algoritmo quasi-Newton BFGS

1: $H_0 \leftarrow I$;
2: $k \leftarrow 0$;
3: calcular $\nabla f(x_0)$;
4: **while** $\|x_{k+1} - x_k\| < \varepsilon$ **do**
5: $\quad \alpha^* \leftarrow arg\ min_\alpha f\left(x_k - \alpha \cdot H_k \cdot \nabla f(x_k)\right)$;
6: $\quad x_{k+1} \leftarrow x_k - \alpha^* \cdot H_k \cdot \nabla f(x_k)$;
7: $\quad$ calcular $\nabla f(x_{k+1})$;
8: $\quad r_k \leftarrow x_{k+1} - x_k$;
9: $\quad v_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$;
10: $\quad H_{k+1} \leftarrow H_k + \left(1 + \dfrac{r_k' \cdot H_k \cdot r_k}{r_k' \cdot v_k}\right) \cdot \dfrac{v_k \cdot v_k'}{v_k' \cdot v_k} - \dfrac{v_k \cdot r_k' \cdot H_k + H_k \cdot r_k \cdot v_k'}{r_k' \cdot v_k}$;
11: $\quad k \leftarrow k + 1$;
12: **end while**

Onde:

x_0 é a solução inicial;

I é a matriz identidade.

A determinação de α^* é realizada através de algum método de otimização unidimensional, como o algoritmo da seção áurea, descrito abaixo (Takahashi, 2004).

Algoritmo da Seção Áurea

1: $x_a \leftarrow b - 0.618(b-a)$;
2: $x_b \leftarrow a + 0.618(b-a)$;
3: $f_a \leftarrow f(x_a)$;
4: $f_b \leftarrow f(x_b)$;
5: **while** $b - a > \epsilon$ **do**
6: **if** $f_a < f_b$ **then**
7: $b \leftarrow x_b$;
8: $x_b \leftarrow x_a$;
9: $x_a \leftarrow b - 0.618(b-a)$;
10: $f_b \leftarrow f_a$;
11: $f_a \leftarrow f(x_a)$;
12: **else**
13: $a \leftarrow x_a$;
14: $x_a \leftarrow x_b$;
15: $x_b \leftarrow a + 0.618(b-a)$;
16: $f_a \leftarrow f_b$;
17: $f_b \leftarrow f(x_b)$;
18: **end if**
19: **end while**
20: $x_f \leftarrow \dfrac{a+b}{2}$;

Onde:

a e b são os limites considerados pela seção áurea;

x_f é o ponto que minimiza f na direção considerada.

O algoritmo quasi-Newton BFGS depende da diferenciabilidade da função objetivo e das restrições para funcionar adequadamente. Além disso, esse algoritmo se mostra ineficiente em problemas multimodais, uma vez que a solução ótima encontrada é altamente dependente do ponto inicial considerado.

2.3 Algoritmos Determinísticos para Problemas Multiobjetivo

Algoritmos determinísticos não-lineares, como o algoritmo quasi-Newton discutido na Sec. 2.2.1, não são capazes de mapear todo o conjunto de Pareto diretamente, uma vez que os mesmos operam em uma única função objetivo. Entretanto, pode-se associar a esses algoritmos técnicas que tornem os mesmos capazes de obter o conjunto de Pareto em problemas contínuos. Com o uso destas técnicas é possível obter uma solução após cada execução completa do algoritmo. Isso implica na necessidade de se executar o algoritmo ao menos P vezes para obter P pontos no conjunto de Pareto (diz-se que o algoritmo deve ser executado ao menos P vezes porque nem todas as soluções obtidas pertencem necessariamente ao conjunto de Pareto).

Na seqüência, são discutidos três métodos capazes de estimar o conjunto de Pareto em problemas contínuos, utilizando algoritmos determinísticos.

2.3.1 Abordagem via Problema Ponderado - $P\lambda$

Uma técnica intuitiva para solução de problemas é o $P\lambda$, que propõe uma soma ponderada dos objetivos. Os pontos do conjunto de Pareto são obtidos através da variação do vetor de pesos, utilizando a formulação apresentada abaixo:

$$x = \arg\min_{x} \sum_{i=1}^{m} \lambda_i f_i(x) \tag{2.10}$$

$$\text{sujeito a: } \begin{cases} x & \in & \mathcal{F}_x \\ \lambda_i & \geq & 0 \\ \sum_{i=1}^{m} \lambda_i & = & 1 \end{cases}$$

Uma vantagem deste método é a manutenção da estrutura de restrições do problema original. No entanto, esse método apresenta como principal limitação a incapacidade de mapear todo o Pareto em problemas não-convexos: o $P\lambda$ só é capaz de mapear as soluções do conjunto de Pareto que pertençam à casca convexa de $\mathcal{F}$. Isto pode ser visto na interpretação geométrica do algoritmo, extraída de Takahashi (2004):

Montando a formulação do $P\lambda$ no espaço de objetivos:

$$y^* = \arg\min_{y \in \mathcal{F}_y} \sum_{i=1}^{m} \lambda_i y_i \tag{2.11}$$

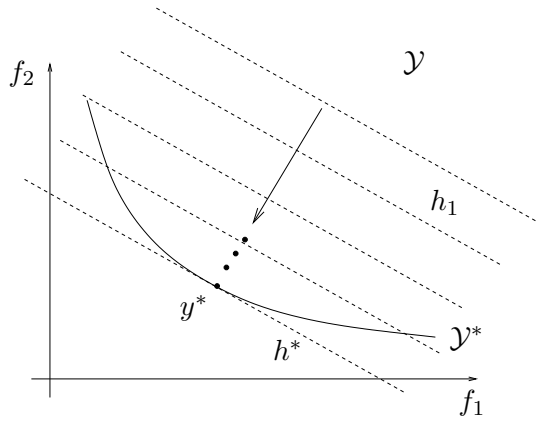
Este problema pode ser visto como: encontrar o ponto $y^* \in \mathcal{F}_y$ que minimiza o produto escalar $\lambda' y$. Se α^* é o valor mínimo desse produto, então:

$$\lambda' y^* = \alpha^* \tag{2.12}$$

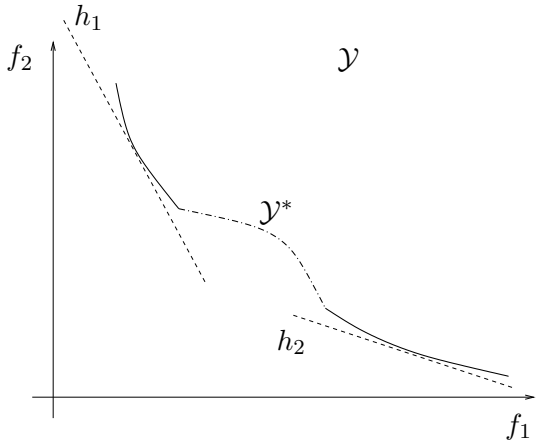
é a equação do hiperplano suporte a $\mathcal{F}_y$ no ponto y^*. Essa interpretação é ilustrada na Fig. 2.2(a).

Percebe-se portanto que essa técnica só é capaz de encontrar soluções y^* da fronteira Pareto $\mathcal{Y}^*$ que admitam hiperplanos de suporte, o que só ocorre nas soluções pertencentes à casca convexa do conjunto $\mathcal{F}_y$. A Fig. 2.2(b) ilustra um problema onde parte das soluções não pode ser obtida utilizando a abordagem ponderada.

É importante salientar que diferentes vetores de pesos podem levar à mesma solução (ou uma solução muito próxima), o que pode obrigar a mais de P execuções do algoritmo para obtenção de P pontos do Pareto.



(a) Partindo de um hiperplano inicial h_1, cuja inclinação é definida pelo vetor de ponderações λ, um algoritmo de otimização mono-objetivo determina uma seqüencia de hiperplanos paralelos a h_1, buscando a minimização da distância de cada hiperplano em relação à origem do espaço $\mathcal{Y}$, com a restrição de que o hiperplano contenha algum ponto de $\mathcal{F}_y$. O algoritmo converge para o hiperplano suporte h^*, cujo único ponto de interseção com $\mathcal{F}_y$ é o ponto y^*, pertencente ao conjunto de Pareto $\mathcal{Y}^*$.



(b) A fronteira Pareto $\mathcal{Y}^* \in \mathcal{Y}$ possui trechos, representados por linha contínua, nos quais todos os pontos possuem hiperplano suporte, e trechos, representados por linha tracejada, nos quais nenhum ponto admite hiperplano suporte. Nenhum desses pontos pode ser obtido utilizando a formulação $P\lambda$.

Figura 2.2: $P\lambda$

2.3.2 Abordagem via Problema ϵ Restrito - Pϵ

Outra formulação alternativa para obtenção do conjunto de Pareto em problemas contínuos é apresentada em (2.13). Neste caso, o problema atua na otimização de apenas uma das funções objetivo, tratando as outras como restrições. Para se obter novas soluções varia-se o vetor das restrições ϵ. A Fig. 2.3 ilustra essa abordagem (Takahashi, 2004).

$$x^* = \arg\min_x f_i(x) \tag{2.13}$$

$$\text{sujeito a: } \begin{cases} x & \in & \mathcal{F}_x \\ f_j(x) & \leq & \epsilon_j, \quad j = 1, \ldots, m, \quad j \neq i \end{cases}$$

Pode ser demonstrado que esse método é capaz de encontrar todas as soluções do conjunto de Pareto (Takahashi, 2004). No entanto, os problemas gerados por esse método podem se tornar computacionalmente complexos, uma vez que o mesmo altera a estrutura de restrições do problema original (objetivos não-lineares se tornam restrições não-lineares).

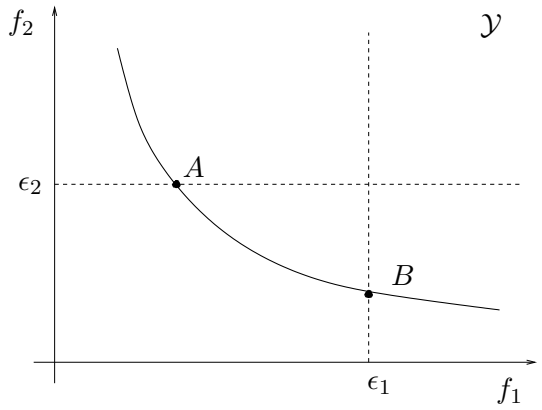


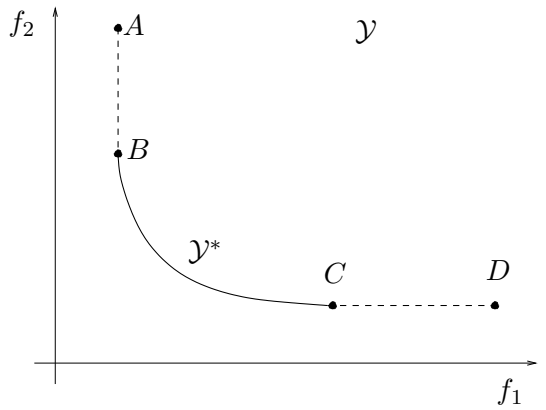
Figura 2.3: A fronteira Pareto $\mathcal{Y}^*$ é representado pela linha contínua. Os pontos A e B pertencem à fronteira Pareto, sendo encontrados através do problema $P\epsilon$ respectivamente: com a minimização de f_1 sujeita a $f_2 \leq \epsilon_2$ e com a minimização de f_2 sujeita a $f_1 \leq \epsilon_1$.

Além disso, este método se depara com mais dois problemas (Takahashi, 2004):

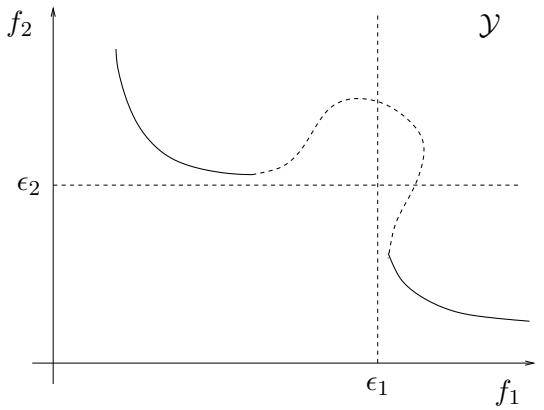
- Alguns vetores ϵ podem resultar em soluções não-eficientes (Fig. 2.4(a));
- Alguns vetores ϵ podem gerar problemas infactíveis (Fig. 2.4(b)).

2.3.3 Abordagem Híbrida

Pode-se combinar as duas técnicas descritas anteriormente para gerar uma técnica menos complexa que a $P\epsilon$, e que seja capaz de mapear todo o conjunto de Pareto em problemas contínuos não-convexos.



(a) O conjunto imagem da função $f(\cdot)$ possui trechos de sua fronteira não pertencentes ao conjunto de Pareto: os trechos AB e CD. Como o valor da função f_1 é igual em todo o trecho AB, e corresponde ao mínimo de f_1, a minimização desta função com a restrição $f_2 < \epsilon_2$ pode gerar pontos pertencentes a AB e portanto não pertencentes ao conjunto de Pareto. O mesmo ocorre no trecho CD considerando f_2 como função objetivo.



(b) A fronteira do conjunto imagem da função $f(\cdot)$ é representada por uma linha contínua no trecho que corresponde ao conjunto de Pareto do problema, e por uma linha tracejada no trecho que não faz parte do mesmo. Caso sejam estabelecidas as restrições $f_1 < \epsilon_1$ e $f_2 < \epsilon_2$ simultaneamente, não haverá solução factível, embora cada restrição isoladamente admita soluções.

Figura 2.4: Falhas do $P\epsilon$

Essa abordagem híbrida é composta de dois passos:

1. Executar o $P\lambda$ para uma estimativa inicial do Pareto;

2. Caso existam "buracos" na fronteira Pareto encontrado pelo $P\lambda$, então utilizar o $P\epsilon$ apenas nesses buracos, visando preencher completamente o Pareto, caso seja possível.

A principal vantagem dessa técnica é que problemas de maior complexidade que o original somente serão resolvidos quando forem realmente necessários. Além disso o método auxilia diretamente na determinação dos limites dos vetores ϵ. Por outro lado, sua aplicação se torna inviável em problemas com mais de duas funções objetivo, uma vez que a detecção de buracos na fronteira Pareto pode se tornar complexa em três ou mais dimensões.

Descrições de outras técnicas voltadas à solução de problemas multi-objetivo utilizando algoritmos determinísticos podem ser encontradas em Chankong and Haimes

(1983); Ehrgott (2000); Takahashi (2004).

2.4 Tratamento de Restrições em Algoritmos Determinísticos

Dois métodos são usualmente empregados no tratamento de restrições através de algoritmos baseados em direções de busca:

- Método de barreira;
- Método de penalidade.

Uma breve descrição desses métodos é apresentada abaixo. Detalhes sobre estes e outros métodos podem ser encontradas em Luenberger (1984); Bazaraa et al. (1993); Takahashi (2004).

2.4.1 Método de Barreira

Seja o problema de otimização restrita:

$$x^* = \arg\min_x f(x)$$

$$\text{sujeito a: } \begin{cases} g_1(x) & \leq & 0 \\ g_2(x) & \leq & 0 \\ \vdots & & \\ g_p(x) & \leq & 0 \end{cases}$$

Uma das possíveis formulações do método de barreira é apresenta na seqüência (Takahashi, 2004):

$$x^* = \arg\min_{x,\alpha} F(x,\alpha) \tag{2.14}$$

onde:

$$\begin{aligned} F(x,\alpha) &= f(x) - \sum_{i=1}^{p} \frac{\alpha_i}{g_i(x)} \\ \alpha &= [\ \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \dots \quad \alpha_p\ ]' \end{aligned} \tag{2.15}$$

sendo $1 \gg \alpha_i > 0$.

Esta formulação modificada do problema aproxima o problema original, sendo que $F(x,\alpha)$ assume um valor próximo de infinito para pontos na fronteira da região factível do problema.

Para que esta formulação funcione, deve-se partir de um ponto inicial dentro da região factível. $F(x,\alpha)$ é uma função que apresenta as seguintes propriedades:

- $\lim_{g(x)\to 0^-} F(x,\alpha) = +\infty$
- $F(x,\alpha) \approx f(x)\ ,\ \alpha_i \ll g(x)\ \forall i$
- $\lim_{\alpha\to 0^+} F(x,\alpha) = f(x)\ ,\ \forall\ g(x) < -\epsilon$

onde $\epsilon > 0$ é um escalar pequeno.

2.4.2 Método de Penalidade

O mesmo problema de otimização restrita, quando modelado pelo método de penalidade, pode apresentar a seguinte formulação Takahashi (2004):

$$x^* = \arg\min_{x,\alpha} F(x,\alpha) \tag{2.16}$$

onde:

$$\begin{aligned} F(x,\alpha) &= f(x) + \sum_{i=1}^{p} G_i(x,\alpha_i) \\ G_i(x,\alpha_i) &= \begin{cases} 0 & , \quad se\ g_i(x) \leq 0 \\ \alpha_i \cdot (g_i(x) + g_i(x)^2) & , \quad se\ g_i(x) > 0 \end{cases} \\ \alpha &= [\ \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \ldots \quad \alpha_p\ ]' \end{aligned} \tag{2.17}$$

sendo $\alpha_i \gg 0$ um escalar grande.

Nesta formulação, não existe a necessidade de se partir de um ponto inicial restrito à região factível do problema. Pode-se notar que $F(x,\alpha)$ é uma função que apresenta as seguintes propriedades:

- $F(x,\alpha) \gg f(x)\ ,\ \forall\ g(x) > 0$
- $(g(x_1) - g(x_2) > 0\ ;\ g(x_2) > 0) \Rightarrow F(x_1,\alpha) > F(x_2,\alpha)$
- $F(x,\alpha) \approx f(x)\ ,\ \forall\ g(x) < \epsilon$
- $\lim_{\alpha \to 0^+} F(x,\alpha) = f(x)\ ,\ \forall\ g(x) < -\epsilon$

onde $\epsilon > 0$ é um escalar pequeno.

Capítulo 3

Algoritmos Evolucionários

Em problemas de otimização em espaços contínuos, os métodos de otimização determinísticos clássicos, como métodos de direção de busca (discutidos no Cap. 2) ou exclusão de semi-espaços (informações sobre estes métodos podem ser encontrados nas referências (Bazaraa et al., 1993; Luenberger, 1984)), são construídos sobre premissas que limitam sensivelmente sua aplicabilidade. Isso motiva a busca por classes de algoritmos mais gerais, capazes de encontrar boas soluções em problemas nos quais os métodos clássicos se mostram ineficazes. Os algoritmos evolucionários apresentam essa característica, uma vez que conseguem resolver de forma satisfatória grande parte dos problemas sem grandes esforços de adaptação (Johnson and Ramat-Semii, 1997; Man et al., 1996).

No caso de otimização combinatória, geralmente não se conhecem métodos exatos para solução de problemas *NP-hard* em tempo polinomial. Isso faz com que a obtenção de boas soluções em tempo aceitável geralmente só seja possível através de algoritmos que empregam algum tipo de randomização. Os algoritmos evolucionários constituem portanto boas alternativas para tratar estes problemas. Uma grande vantagem que se espera obter dos algoritmos evolucionários é a facilidade de sua adaptação a problemas com estruturas complexas.

Neste capítulo são apresentados três algoritmos evolucionários, sendo dois voltados à otimização de problemas mono-objetivo e um dedicado à problemas multi-objetivo.

Estes algoritmos são utilizados ao longo deste trabalho para solução de diversos problemas de projeto ótimo de redes. Por fim são discutidas as adaptações utilizadas nestes algoritmos para o tratamento de redes.

3.1 Algoritmos Evolucionários Mono-objetivo

3.1.1 Algoritmo Genético

Introdução

Algoritmos Genéticos são algoritmos estocásticos inspirados na teoria da evolução de Charles Darwin (Goldberg, 1989). Estes trabalham sobre uma população de soluções candidatas e modelam artificialmente processos característicos da evolução de espécies, como seleção, cruzamento e mutação. A cada indivíduo da população é atribuída uma aptidão (ou *fitness*) que tem como base sua avaliação. Dá-se aos indivíduos mais fortes maior chance de sobreviver e conseqüentemente gerar descendentes. A melhoria da população é conseqüência da repetida seleção dos indivíduos mais aptos, que tem maior chance de produzir bons descendentes Fonseca (1995).

O livro publicado por Holland (1975) é considerado o ponto inicial para desenvolvimento dos GAs. David Goldberg, aluno de Holland, obteve sucesso na aplicação industrial de GAs em meados dos anos 80. Desde então os mesmos vêm tendo crescente aplicação.

Características dos GAs, como as citadas no Cap. 1, fizeram dos mesmos uma das ferramentas mais utilizadas na solução de problemas otimização. Estes algoritmos vêm sendo aplicados em diversas áreas, como:

- Planejamento de processos (Awadh et al., 1995);
- Projeto da aerodinâmica de asas para aviões subsônicos (Anderson, 1996);
- Reconhecimento de objetos (Aherne et al., 1997);
- Planejamento de rotas espaciais (Hartmann et al., 1998);

- Problemas de *scheduling* (Cardon et al., 1999);
- Planejamento urbano (Feng and Lin, 1999);
- Projeto da aerodinâmica de mísseis (Anderson et al., 2000);
- Geração de imagens (Aguirre et al., 2001).

Algoritmo

A estrutura básica de um GA é apresentada abaixo. Os aspectos mais importantes desta estrutura são discutidos na seqüencia. A descrição dos mesmos foi adaptada de (Fonseca, 1995).

Estrutura Básica do Algoritmo Genético

1: $pop_0 \leftarrow$ gerar população(N);
2: $f_0 \leftarrow$ avaliar(pop_0);
3: $fit_0 \leftarrow$ calcular *fitness*(f_0);
4: $conv \leftarrow$ false;
5: $ger \leftarrow 1$;
6: **while** $conv =$ false **do**
7: $\quad pop_{ger} \leftarrow$ seleção($fit_{ger-1}, pop_{ger-1}, s$);
8: $\quad pop_{ger} \leftarrow$ shuffle(pop_{ger});
9: $\quad$ **for** $i = 1$ até $\frac{N}{2}$ **do**
10: $\quad\quad$ **if** random $\leq p_{cruz}$ **then**
11: $\quad\quad\quad (pop^i_{ger}, pop^{\frac{N}{2}+1}_{ger}) \leftarrow$ cruzamento($pop^i_{ger}, pop^{\frac{N}{2}+1}_{ger}$);
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad$ **for** $i = 1$ até N **do**
15: $\quad\quad$ **if** random $\leq p_{mut}$ **then**
16: $\quad\quad\quad (pop^i_{ger}) \leftarrow$ mutação(pop^i_{ger});

```
    end if
  end for
  f_ger ← avaliar(pop_ger);
  fit_ger ← calcular fitness(f_ger);
  if convergiu then
    conv ← true;
  else
    ger ← ger + 1;
  end if
end while
```

Parâmetros de Entrada:

Quatro parâmetros devem ser definidos a priori, para execução do GA:

- N - Tamanho da população: Inteiro, par;
- s - Número de cópias esperadas para o melhor indivíduo: Real;
- p_{cruz} - Probabilidade de cruzamento por par de indivíduos: Real, $0 \leq p_{cruz} \leq 1$;
- p_{mut} - Probabilidade de mutação por indivíduo: Real, $0 \leq p_{mut} \leq 1$.

População:

Nos GA's, a população deve ser considerada em 2 níveis: fenótipo e genótipo. O fenótipo de cada indivíduo representa o mesmo no domínio em que a função está definida (o espaço $\mathbb{R}^n$ em problemas contínuos por exemplo). Já o genótipo do indivíduo representa o mesmo em um domínio codificado, onde o algoritmo atua. Qualquer estrutura conveniente pode ser utilizada para codificar o fenótipo em genótipo (Rothlauf, 2005). Na maior parte dos casos, os genótipos são construídos através de seqüências de bits, o que tem como inspiração as seqüências genéticas contidas em cromossomos biológicos (Goldberg, 1989). A distinção entre genótipo e fenótipo é importante, uma vez que

facilita a generalização do algoritmo. No entanto, em determinados casos esta distinção não precisa ser rígida, podendo haver uma correspondência direta entre genótipo e fenótipo: o algoritmo genético real polarizado (Ramos et al., 2003) por exemplo, utiliza uma representação real na qual genótipo e fenótipo são coincidentes. A relação genótipo-fenótipo, deve obedecer um mapeamento $1 \rightarrow 1$ (codificações não-redundantes), ou $n \rightarrow 1$ (codificações redundantes).

Avaliação da População e Atribuição da *Fitness*:

Os indivíduos da população são avaliados através da função objetivo que define o problema. Os valores de função objetivo calculados são utilizados para atribuição da *fitness*.

A distinção do valor de função objetivo e a *fitness* é capaz realizar a interface entre problemas de minimização e a natureza de maximização dos GA's. No entanto, a principal vantagem deste processo é a possibilidade de controle da pressão seletiva do algoritmo. Isso permite a manutenção da representatividade da população. O uso de técnicas, como o ranking linear descrito na seqüência, age nesse intuito.

Ranking Linear:

1. Avaliar os N indivíduos da população utilizando a função objetivo;

2. Ordenar os indivíduos do melhor para o pior, seguindo o valor de função objetivo calculado;

3. Atribuir o *ranking* (r) de cada um dos indivíduos da população ordenada, sendo que o melhor indivíduo recebe $r = 0$ e o pior $r = N - 1$;

4. Determinar a fitness de cada um dos indivíduos através de:

$$fit(r) = s - (s - 1) \cdot \frac{2r}{N - 1} \tag{3.1}$$

onde s é o número de cópias desejado para o melhor indivíduo.

O *ranking* linear impede que uma solução que se destaque muito do restante da população, e receba um número muito grande de cópias na seleção.

Uma descrição sobre outros métodos de ranking pode ser encontrada em (Goldberg, 1989; Fonseca, 1995; Tanomaru, 1995; Silva et al., 2006).

Seleção:

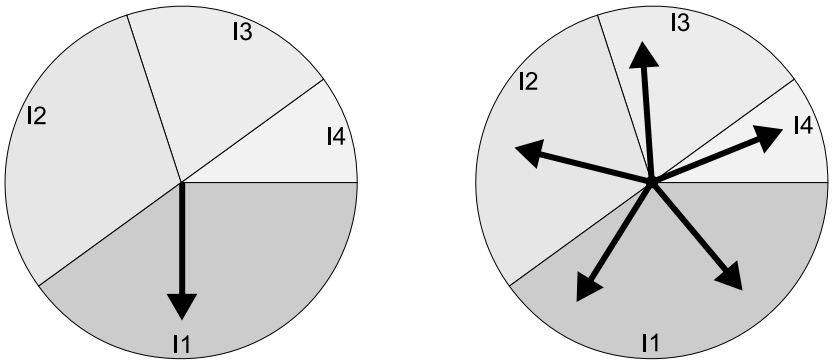
A seleção em GA's tem como função escolher os indivíduos que deverão participar do cruzamento e mutação, seguindo como proporção a *fitness*. No entanto, uma vez que o número de cópias deve ser um inteiro, o número de indivíduos selecionados será maior, ou menor, que o valor desejado, o que causa um erro de seleção.

A seleção em GA's é geralmente realizada de forma estocástica, visando evitar erros de seleção sistemáticos. O número esperado de operações realizadas sobre um indivíduo deveria ser exatamente correspondente a sua *fitness*, enquanto que, o erro de seleção estocástico deveria ser o menor possível.

O método mais utilizado para seleção é a roleta estocástica (do inglês, *Roulette Wheel Selection* ou RWS) Goldberg (1989). A roleta estocástica consiste de uma seqüência de eventos de seleção independentes, onde a probabilidade de cada indivíduo ser selecionado é proporcional a sua *fitness* relativa (a *fitness* relativa define a área do slot referente a cada indivíduo na roleta). Ao todo são realizadas N seleções independentes.

No entanto a roleta estocástica geralmente provoca grandes erros de seleção. Para reduzir estes erros de seleção, e os erros de amostragem, Baker (1987) propôs um método de seleção chamado amostragem universal estocástica (do inglês, *Stochastic Universal Sampling* ou SUS). Neste método, a roleta é preenchida da mesma forma que na roleta estocástica, com a área do slot referente a cada indivíduo proporcional à sua *fitness*. No entanto, ao invés de N seleções independentes é realizada uma única seleção, com uma roleta composta por N ponteiros igualmente espaçados. Após a execução do SUS as soluções selecionadas são ordenadas aleatoriamente, para evitar polarizações durante o cruzamento e mutação.

Como exemplo, seja uma população de 5 indivíduos cujos valores de função objetivo são $f_{ob} = [1\ \ 2\ \ 3\ \ 4\ \ 10]$ e os valores de fitness são $fit = [2\ \ 1.5\ \ 1\ \ 0.5\ \ 0]$ (após aplicação do *ranking* linear). Uma possível realização do RWS e do SUS para este problema pode ser vista na Fig. 3.1.



(a) RWS (b) SUS

Figura 3.1: Métodos de seleção em GA's: RWS *vs.* SUS

Outro método usualmente utilizado para seleção é o Torneio (do inglês *Tournament Selection* ou TS) Hancock (1994). Em uma implementação simplificada, para cada passo da seleção são extraídos dois indivíduos da população e o melhor deles é selecionado. Assim como no RWS, o TS pode apresentar grandes erros de seleção. No entanto pode-se reduzir estes erros aumentando o número de indivíduos por comparação, e utilizando algum outro método de seleção para escolher os indivíduos que serão submetidos ao torneio.

Detalhes sobre outros métodos de seleção disponíveis podem ser encontradas em (Goldberg, 1989; Fonseca, 1995; Tanomaru, 1995; Soares, 1997).

Mutação e Cruzamento:

A simples seleção de indivíduos não é capaz de produzir nada melhor que o melhor indivíduo da população. Devido a isso, são necessários mecanismos de variação, capazes de inserir "novidades" na população. Os operadores genéticos, que modificam os indivíduos da população através de seu genótipo, podem ser divididos em dois tipos:

Mutação:

Na mutação, o genótipo do indivíduo é modificado conforme alguma regra probabilística. Usualmente apenas uma pequena parte do genótipo é modificada, o que idealmente implica em pequenas modificações no fenótipo. Com isso, o indivíduo resultante herda a maior parte das características do pai. Métodos dedicados a mutação de indivíduos reais e binários podem ser encontrados em (Goldberg, 1989; Tanomaru, 1995; Soares, 1997; Ramos et al., 2003).

Cruzamento:

No cruzamento (ou recombinação), os genótipos de dois indivíduos "pais" são combinados para criação de dois novos "filhos", também seguindo algum princípio estocástico. Nesta operação espera-se que os indivíduos resultantes tragam informações herdadas dos pais.

Convergência:

Vários critérios podem ser estabelecidos para análise de convergência de GA's, como:

- Obtenção do ótimo conhecido: este critério interrompe a execução do algoritmo após a obtenção de um ótimo previamente conhecido. Sua aplicação é útil para avaliação de algoritmos evolucionários;

- Estabilização do valor de função objetivo: este critério interrompe o algoritmo após uma não-melhoria da melhor solução encontrada durante um número pré-determinado de avaliações de função. Neste caso parte-se do princípio que o algoritmo convergiu para o ótimo ou chegou a uma região da qual não se consegue melhora significativa;

- Execução de um número pré-determinado de gerações: este critério interrompe a execução do algoritmo após a execução de um número pré-determinado de gerações. A principal vantagem deste método é a fixação do custo computacional do algoritmo.

3.1.2 Algoritmo de Seleção Clonal

Introdução

Os algoritmos de seleção clonal são algoritmos estocásticos baseados em sistemas imunológicos artificiais (do inglês *Artificial Immunology System*, ou AIS). Um modelo básico do funcionamento do sistema imunológico, que serve de base para o funcionamento deste algoritmo, é brevemente descrito abaixo.

O Sistema Imunológico

Em um modelo simplificado, o sistema imunológico é composto basicamente por 3 tipos de células: macrófagos, células-B e células-T (Fig. 3.2). O macrófago faz parte do sistema imunológico inato do indivíduo e forma a primeira linha de defesa do organismo. Ele fagocita os invasores. Entretanto, somente isso não é suficiente para proteção, pois alguns invasores se aproveitam deste fato: vírus, por exemplo, se utilizam dos recursos de macrófagos para criarem cópias de si mesmos.

Portanto, esquemas adicionais de proteção devem atuar. Os linfócitos, células-B e células-T, formam a segunda linha de defesa. As células-T são capazes de examinar o que o macrófago comeu e de reconhecer agentes estranhos. Caso um agente estranho seja identificado pela célula-T, esta aciona as células-B, que produzem anticorpos específicos para o agente invasor. Esses anticorpos são capazes de se ligar a determinadas proteínas na superfície do agente invasor, denominadas antígenos. Antígenos com anticorpos ligados a ele podem ser ingeridos pelos macrófagos. Dessa forma é combatida uma infecção no sistema imunológico.

Os algoritmos baseados em sistemas imunológicos artificiais, apesar de recentes (cerca de dez anos), têm recebido cada vez mais atenção. As referências (Dasgupta, 1999; de Castro and Timis, 2002; Dasgupta et al., 2003) apresentam uma visão geral de algoritmos baseados em sistemas imunológicos artificiais. Estes algoritmos têm apresentado resultados promissores, e vêm sendo aplicados em áreas como:

- Reconhecimento de padrões (Forrest et al., 1993);

- Segurança de computadores (Dasgupta and Gonzalez, 2002);
- Navegação de robôs (Michelan and Von Zuben, 2002);
- Detecção de falhas em sistemas de distribuição (Huang, 2002);
- Otimização em geral (de Castro and Timis, 2002);
- Otimização de dispositivos magnéticos (Campelo et al., 2005).

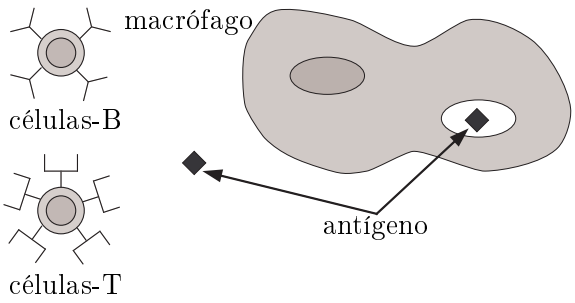


Figura 3.2: Células do sistema imunológico

Algoritmo

A estrutura básica do Algoritmo de Seleção Clonal é apresentada abaixo. Os aspectos mais importantes desta estrutura são discutidos na seqüencia.

Estrutura Básica do Algoritmo de Seleção Clonal

1: $pop_0 \leftarrow$ gerar população(N);
2: $f_0 \leftarrow$ avaliar(pop_0);
3: $fit_0 \leftarrow$ calcular *fitness*(f_0);
4: $conv \leftarrow$ false;
5: $iter \leftarrow 1$;
6: **while** $conv =$ false **do**
7: $\quad (pop_{iter-1}, f_{iter-1}) \leftarrow$ ordenar(pop_{iter-1}, f_{iter-1});

8: **for** $i \leftarrow 1$ até round$(b \cdot N)$ **do**
9: $pop^i_{iter} \leftarrow pop^i_{iter-1}$;
10: $f^i_{iter} \leftarrow f^i_{iter-1}$;
11: $Nc^i \leftarrow \text{round}\left(\frac{\beta \cdot N}{i}\right)$;
12: **for** $j \leftarrow 1$ até Nc^i **do**
13: $cl \leftarrow \text{clone}(pop^i_{iter})$;
14: $\alpha = \gamma \cdot e^{-fit^i_{iter-1}}$;
15: $cl \leftarrow \text{mutação}(cl, \alpha)$;
16: $f_{cl} \leftarrow \text{avaliar}(cl)$;
17: **if** $f_{cl} < f^i_{iter}$ **then**
18: $pop^i_{iter} \leftarrow cl$;
19: $f^i_{iter} \leftarrow f_{cl}$;
20: **end if**
21: **end for**
22: **end for**
23: **for** $i \leftarrow \text{round}(b \cdot N) + 1$ até N **do**
24: $pop^i_{iter} \leftarrow$ gerar indivíduo;
25: $f^i_{iter} \leftarrow \text{avaliar}(pop^i_{iter})$;
26: **end for**
27: **if** convergiu **then**
28: $conv \leftarrow$ true;
29: **else**
30: $fit_{iter} \leftarrow$ calcular $fitness(f_{iter})$;
31: $iter \leftarrow iter + 1$;
32: **end if**
33: **end while**

Parâmetros de Entrada:

Quatro parâmetros devem ser definidos à priori, para execução do Clonal:

- N - Tamanho da população: Inteiro;
- b - Taxa de seleção: Real, $0 < b \leq 1$;
- β - Taxa de clonagem do melhor indivíduo: Real, $0 < \beta \leq 1$;
- γ - Raio de mutação base.

População:

As mesmas características descritas para a população do GA também são válidas para o algoritmo de Seleção Clonal. No entanto, em problemas reais, geralmente é utilizada a codificação real uma vez que este algoritmo requer a definição de uma métrica no espaço de genótipos.

Avaliação da População e Atribuição da *Fitness*:

Os indivíduos da população são avaliados através da função objetivo do problema. Diferentemente do GA, não há seleção estocástica no Clonal, o que dispensa a necessidade de controle da pressão seletiva. No Clonal, a *fitness* é aplicada no controle do raio de mutação de algoritmo: espera-se uma busca local em torno dos melhores indivíduos e uma busca global em torno dos indivíduos menos aptos. Após a ordenação da população, a *fitness* dos indivíduos é definida por:

$$fit^i = 1 - \frac{r}{N-1} \tag{3.2}$$

Onde:

r é um inteiro que define o *ranking* do indivíduo i, que varia de 0 (melhor indivíduo) e $N-1$ (pior indivíduo).

Seleção:

A seleção no Clonal é controlada pelo parâmetro b que determina qual a porcentagem dos melhores indivíduos devem permanecer na população. Os $b \cdot N$ melhores indivíduos são mantidos na população, e são posteriormente submetidos à clonagem. O restante dos indivíduos são descartados, e substituídos por novos indivíduos, gerados aleatoriamente. A seleção do algoritmo Clonal não pressiona a população para uma região do espaço, uma vez que não gera cópias dos melhores indivíduos. A inserção de novos indivíduos aleatórios tem como objetivo inserir constantemente diversidade na população, e eventualmente alcançar áreas ainda não exploradas pelo algoritmo.

Clonagem:

A clonagem, juntamente com a mutação, constitui o principal processo de diferenciação do algoritmo Clonal. A taxa de clonagem utilizada é proporcional à *fitness*, o que intensifica a busca em torno dos melhores indivíduos. Essa abordagem é coerente, uma vez que é razoável assumir que as melhores soluções encontradas até um dado momento estejam mais próximas das melhores soluções alcançáveis para o problema. Após o ordenamento das soluções, define-se o número de clones de cada indivíduo i por:

$$Nc^i = \text{round}\left(\frac{\beta \cdot N}{i}\right) \tag{3.3}$$

Como exemplo, em uma população de 50 indivíduos, com $\beta = 0,7$, ter-se-ão 35 clones do melhor indivíduo, 18 do segundo, 12 do terceiro, 9 do quarto e assim por diante.

Mutação:

Em implementações reais, os clones são geralmente definidos por:

$$C^* = C + \zeta \tag{3.4}$$

onde:

C^* é o clone após mutação;

C é o clone a ser mutado;
ζ é uma pertubação aleatória de módulo $|\zeta| \approx R_U \cdot \gamma \cdot e^{-fit_C}$;
R_U é uma variável aleatória uniforme.

De (3.4), percebe-se outra característica interessante deste algoritmo: em torno dos melhores indivíduos a busca tem caráter local, enquanto que, em torno dos indivíduos com menor *fitness* a busca tem caráter global. A Eq. (3.4) ressalta outro aspecto: a necessidade da definição de uma métrica no espaço de códigos e a possibilidade de se obter pontos aleatórios à dadas distâncias, tendo em conta a métrica definida.

Mapeamento de Sub-ótimos:

As características do Algoritmo Clonal conferem ao mesmo uma propriedade interessante: o mapeamento e manutenção de soluções sub-ótimas juntamente com a melhor solução encontrada. Esse aspecto se deve principalmente a três aspectos:

- A seleção do algoritmo não exerce pressão global seletiva sobre a população;
- O algoritmo não utiliza operadores de interpolação (como cruzamento);
- A estrutura de mutação, com taxas de clonagem e raios de mutação adaptativos, favorece o refinamento de cada solução isoladamente (seleção global).

Estes aspectos tornam este algoritmo uma ferramenta aplicável em problemas em que existe uma grande sensibilidade quanto às variáveis de decisão ou em problemas em que se deseja tratar incertezas (como será discutido na Sec. 5.3). É importante destacar que é possível adaptar um algoritmo genético para realizar este mapeamento, eliminando a pressão seletiva ($s = 1$) e utilizando apenas operadores de mutação. No entanto, a estrutura do Clonal, conceitualmente pensada para estes casos, se mostra imediatamente adequada para estes tipos de problemas.

Convergência:

Os mesmos critérios estabelecidos para análise de convergência de GA's são válidos para o Clonal.

3.2 Algoritmos Evolucionários Multi-objetivo

Uma grande vantagem dos algoritmos evolucionários quando aplicados à problemas multi-objetivo é a capacidade dos mesmos em evoluir simultaneamente um conjunto de soluções na direção do conjunto de Pareto do problema. Isto geralmente permite um mapeamento significativo do conjunto de Pareto em uma única execução do algoritmo, o que atenua a razão entre o custo computacional de um algoritmo evolucionário e um algoritmo determinístico (Fonseca and Fleming, 1995; Coello, 2000).

A utilização destes algoritmos em problemas multi-objetivo depende da adaptação do critério de atribuição da *fitness*, uma vez que a relação de melhor ou pior, utilizada para comparação de grandezas escalares, perde sentido em comparações vetoriais.

Essa relação deve ser substituída por alguma relação que expresse a dominância de um indivíduo sobre outro, como a apresentada na Sec. 2.1.2. Os operadores de mutação e cruzamento (no caso dos GA's) têm estruturas que não dependem da natureza mono ou multi-objetivo do problema (Takahashi et al., 2004).

Diferentemente dos algoritmos determinísticos mais clássicos, aplicáveis a problemas contínuos, os algoritmos evolucionários podem ser utilizados para solução de problemas multi-objetivo discretos e contínuos. O conceito de fronteira Pareto para um problema de otimização com variáveis discretas é ilustrado na Fig. 3.3. Em problemas discretos o conjunto de Pareto é formado por pontos isolados ao invés de uma curva contínua, o que se deve à descontinuidade do espaço de parâmetros. Problemas de projeto de redes multi-objetivo apresentam conjuntos de soluções com estas características.

Na seqüência desta seção é apresentado o *NSGA-II* (Deb et al., 2002), que é utilizado neste trabalho para o projeto multi-objetivo de redes de distribuição de energia elétrica. No entanto, existem vários outros algoritmos evolucionários multi-objetivo:

- *VEGA: Vector Enabled GA* (Schaffer, 1984, 1985);
- *MOGA: Multiobjective GA* (Fonseca and Fleming, 1993, 1995);
- *NPGA: Niched Pareto GA* (Horn and Nafpliotis, 1993; Horn et al., 1994);

- *NSGA: Non-dominated Sorting GA* (Srinivas and Deb, 1994);
- *SPEA: Strength Pareto EA* (Zitzler and Thiele, 1999);
- *PAES: Pareto Archived Evolution Strategy* (Knowles and Corne, 1999);
- *PESA: Pareto Envelop-based Selection Algorithm* (Corne et al., 2000);
- *PESA-II* (Corne et al., 2001);
- *SPEA2* (Zitzler et al., 2001);
- *MOIA: Multiobjective Immune Algorithm* (Luh et al., 2003);
- *MOCSA: Multiobjective Clonal Selection Algorithm* (Campelo et al., 2004);
- *IBEA: Indicator-Based EA* (Zitzler and Künzli, 2004).

Informações sobre estes métodos podem ser encontradas nas respectivas referências.

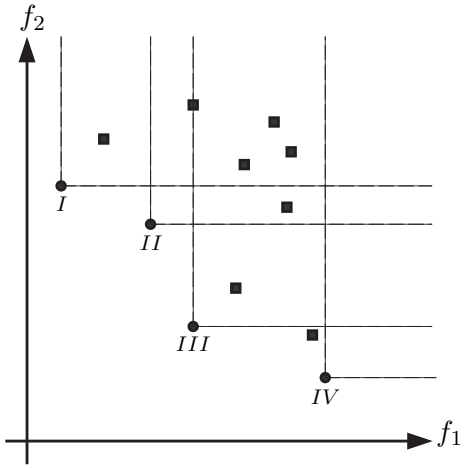


Figura 3.3: Exemplo de fronteira Pareto em um problema bi-objetivo discreto. As soluções I, II, III e IV constituem o conjunto de soluções eficientes do problema.

3.2.1 *NSGA-II*

A estrutura do *NSGA-II*, proposto por Deb et al. (2002), é apresentada abaixo. A seleção empregada neste algoritmo é discutida na seqüência:

Estrutura Básica do *NSGA-II*

$P_1 \leftarrow$ gerar população(N);
$O_1 \leftarrow$ encontrar pareto(P_1);
$P_0 \leftarrow \emptyset$;
$f_{P_0} \leftarrow \emptyset$;
for $ger \leftarrow 1$ até $nger$ **do**
  $f_{P_{ger}} \leftarrow$ avaliar(P_{ger});
  $R_{ger} \leftarrow P_{ger} \bigcup P_{ger-1}$;
  $f_{R_{ger}} \leftarrow f_{P_{ger}} \bigcup f_{P_{ger-1}}$;
  $fit_{R_{ger}} \leftarrow$ calcular fitness(R_{ger});
  $T_{ger} \leftarrow$ stochastic tournament($fit_{R_{ger}}, R_{ger}, N$);
  $T_{ger} \leftarrow$ shuffle(P_{ger});
  for $i = 1$ até $\dfrac{N}{2}$ **do**
    if random $\leq p_{cruz}$ **then**
      $\left(T^i_{ger}, T^{\frac{N}{2}+1}_{ger}\right) \leftarrow$ cruzamento $\left(T^i_{ger}, T^{\frac{N}{2}+1}_{ger}\right)$;
    end if
  end for
  for $i = 1$ até N **do**
    if random $\leq p_{mut}$ **then**
      $\left(T^i_{ger}\right) \leftarrow$ mutação $\left(T^i_{ger}\right)$;
    end if
  end for
  $P_{ger+1} \leftarrow T_{ger}$;
  $O_{ger+1} \leftarrow$ encontrar pareto($O_{ger} \bigcup P_{ger+1}$);

24: $ger \leftarrow ger + 1$;

25: **end for**

Onde:

nger é o número pré-definido de gerações;

O_{nger} é a aproximação do conjunto de Pareto para o problema.

Seleção e *Fitness* no *NSGA-II*

A atribuição da *fitness* em algoritmos evolucionários multi-objetivo desta categoria é baseada no seguinte princípio:

Inicialmente determina-se o conjunto de soluções não dominadas $\mathcal{F}_1$ que é extraído da população. Um novo conjunto de soluções não-dominadas é extraído da população restante, para composição de $\mathcal{F}_2$. O processo se repete até que não existam mais soluções no população. Cada subconjunto da população, $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{k-1}$, contém indivíduos que são dominados apenas pelos indivíduos dos subconjuntos precedentes. O conjunto $\mathcal{F}_1$ contém as soluções não dominadas da população.

Essa regra estabelece um *ranking*, onde os indivíduos da primeira fronteira ($\mathcal{F}_1$) recebem a melhor posição, e os indivíduos da última fronteira ($\mathcal{F}_{k-1}$) recebem a pior posição. Esse ranking é utilizado para atribuição da *fitness* da população.

No caso do *NSGA-II*, esse procedimento é realizado através do *fast-non-dominated sorting*, que têm sua estrutura descrita abaixo. Este método apresenta um custo computacional inferior ao *non-dominated sorting* ($\mathcal{O}(mN^2)$ *vs.* $\mathcal{O}(mN^3)$, onde m é o número de objetivos e N é o tamanho da população), proposto na primeira versão do NSGA (Srinivas and Deb, 1994).

Fast-non-dominated sorting

```
1: P ← população atual;
2: for each p ∈ P do
3:    S_p ← empty;
4:    n_p ← 0;
5:    for each q ∈ P do
6:       if p < q then
7:          S_p ← S_p ⋃ {q};
8:       else if (q < p) then
9:          n_p ← n_p + 1;
10:      end if
11:   end for
12:   if n_p = 0 then
13:      p_rank ← 1;
14:      F_1 ← F_1 ⋃ {p};
15:   end if
16: end for
17: i ← 1;
18: while F_i ≠ ∅ do
19:   Q ← ∅;
20:   for each p ∈ F_i do
21:      for each q ∈ S_p do
22:         n_q ← n_q − 1;
23:         if n_q = 0 then
24:            q_rank ← i + 1;
25:            Q ← Q ⋃ {q};
26:         end if
```

27: **end for**
28: **end for**
29: $i \leftarrow i + 1$;
30: $\mathcal{F}_i \leftarrow Q$;
31: **end while**

---

onde:
$\mathcal{F}_i$ é a i-ésima fronteira do conjunto de soluções.

Outro operador utilizado na seleção do *NSGA-II* é o *crowding-distance-assignment*, que estima a densidade das soluções em cada uma das fronteiras que compõem a população. Este procedimento, associado ao **stochastic tournament**, compõem uma técnica de nicho, que não depende da definição de parâmetros externos. A estrutura do *crowding-distance-assignment* é apresentada abaixo:

Crowding-distance-assignment

1: $l \leftarrow |P|$;
2: $\mathcal{I}[*]_{dist} \leftarrow 0$;
3: **for** m **from** 1 **to** n_{obj} **do**
4: $\mathcal{I} \leftarrow sort(\mathcal{I}, m)$;
5: $\mathcal{I}[1]_{dist} \leftarrow \mathcal{I}[l]_{dist} \leftarrow \infty$;
6: **for** i **from** 1 **to** $l - 1$ **do**
7: $\mathcal{I}[i]_{dist} \leftarrow \mathcal{I}[i]_{dist} + \frac{(\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)}{f_m^{max} - f_m^{min}}$;
8: **end for**
9: **end for**

Onde:
l é o número de indivíduos na população;

$\mathcal{I}$ é a matriz com os valores de função objetivo (vetores de objetivos concatenados);
$sort(\mathcal{I}, m)$ ordena $\mathcal{I}$ pelo m-ésimo objetivo;
$\mathcal{I}[i].m$ retorna o valor do objetivo m para indivíduo i;
$\mathcal{I}[i]_{dist}$ é o valor de *crowding* do indivíduo i.

Por fim, o operador de seleção **stochastic tournament** faz a seleção dos indivíduos através do processo descrito abaixo (que deve ser repetido N vezes):

Stochastic tournament

if $\mathcal{F}^{p_1} < \mathcal{F}^{p_2}$ **then**
    escolher(p_1);
else if $\mathcal{F}^{p_2} < \mathcal{F}^{p_1}$ **then**
    escolher(p_1);
else
    if $\mathcal{I}[p_1]_{dist} > \mathcal{I}[p_2]_{dist}$ **then**
        escolher(p_1);
    else if $\mathcal{I}[p_2]_{dist} > \mathcal{I}[p_1]_{dist}$ **then**
        escolher(p_2);
    else
        escolher aleatoriamente entre p_1 e p_2;
    end if
end if

Onde:
p_1 e p_2 são os indivíduos escolhidos para seleção;
$\mathcal{F}^{p_i}$ é a fronteira a que pertence o ponto p_i.

3.3 Tratamento de Restrições em Algoritmos Evolucionários

Geralmente, os algoritmos evolucionários tratam restrições através de três formas:

- Penalidade de soluções;
- Eliminação de soluções;
- Reparo de soluções.

3.3.1 Penalidade de Soluções

Neste caso todas as potenciais soluções são geradas desconsiderando as restrições do problema. Durante a avaliação das soluções verifica-se quais delas violam as restrições. A essas soluções acrescenta-se uma parcela de penalização. Esse procedimento é semelhante ao método de penalidade apresentado na Sec. 2.4.

3.3.2 Eliminação de Soluções

Nesse método de tratamento de restrições, todas as soluções não restritas ao conjunto factível são substituídas por novas, que estejam dentro da região factível do problema. Caso necessário, parte dessas novas soluções pode ser descartada por baixo desempenho.

Esta técnica só é viável em problemas com baixa probabilidade de geração de soluções infactíveis, uma vez que a mesma pode causar perda de diversidade da população.

3.3.3 Reparo de Soluções

Este método propõe a retificação das soluções através de um algoritmo específico, pelo qual soluções infactíveis do problema são deslocadas para a região factível.

Em alguns casos, o reparo de soluções podem acarretar em um alto custo computacional, o que pode inviabilizar a aplicação desta técnica.

3.4 Dificuldades na Aplicação de Algoritmos Evolucionários para Projeto Ótimo de Redes

Como foi discutido no Cap. 1, os algoritmos evolucionários se apresentam como ferramentas potencialmente capazes de resolver problemas de projeto de redes, uma vez que possuem uma estrutura flexível, capaz de ser adaptada para os vários tipos de problemas.

No entanto, a estrutura tradicional destes algoritmos, inicialmente concebida para o tratamento de problemas contínuos, se mostra incapaz de funcionar adequadamente em problemas de rede. Essa incapacidade provém da dificuldade de se obter redes factíveis através de modificações em outras redes. Isso faz com que as operações empregadas em algoritmos evolucionários, como mutação e cruzamento, resultem em redes cuja estrutura não interessa ao problema tratado (Smith and Walters, 2000; Carrano et al., 2006b).

Cada um dos métodos de tratamento de restrições citados na Seção 3.3 tende a apresentar um resultado indesejável nessa classe de problemas:

- A penalização das soluções infactíveis tende a gerar uma propagação das soluções infactíveis ao longo do processo evolutivo. Com isso, a busca real do algoritmo se concentra apenas na pequena parcela factível da população, o que obviamente reduz a eficiência do algoritmo.

- A substituição das soluções infactíveis por novas soluções tende a reduzir a performance (*fitness* média) da população, uma vez que uma parcela considerável da população é substituída por soluções completamente novas, ainda não submetidas a nenhum processo de melhoria. Isso implica na redução da eficiência do algo-

ritmo, além de agregar ao mesmo um custo computacional adicional, de gerar novas soluções após cada iteração.

- A substituição das soluções infactíveis por outras soluções factíveis já existentes na população tende a diminuir significativamente a diversidade da população, o que conseqüentemente compromete a busca realizada pelo algoritmo.

- A viabilidade do uso de operadores de retificação de soluções deve ser analisada em cada problema específico, uma vez que, em parte dos casos, seu custo computacional pode ser inviável.

Tendo em vista as limitações citadas acima, outras estratégias devem ser adotadas para resolver o problema de manutenção da factibilidade em problemas de rede. Duas alternativas são tratadas neste trabalho:

- Uso de operadores desenvolvidos para redes;

- Uso de codificações específicas para redes.

3.4.1 Uso de Operadores Desenvolvidos para Redes

Uma alternativa viável para manutenção da factibilidade das soluções é o tratamento da mesma através dos operadores do algoritmo. A utilização de operadores de geração, mutação e cruzamento capazes de garantir a factibilidade da população pode aumentar consideravelmente a eficiência de algoritmos evolucionários em problemas de redes (Carrano et al., 2006b, 2007f). Neste sentido, duas linhas distintas podem ser adotadas:

- Operadores generalizados: Os operadores podem levar em conta apenas os aspectos gerais que fundamentam problemas de grafos, ou levar em consideração apenas as características construtivas de uma determinada estrutura de rede. Neste caso, os operadores assumem um caráter mais geral, e têm sua aplicação estendida para vários problemas práticos distintos.

- Operadores específicos: Por outro lado, podem-se construir operadores tendo como idéia central as características específicas do problema que se deseja tratar. Estes operadores tendem a ser capazes de explorar as características principais do problema, podendo provavelmente ajudar na convergência do algoritmo. No entanto, deve-se ter em conta que a aplicabilidade destes operadores se torna restrita ao problema para o qual foram desenvolvidos.

Ambas as linhas de operadores citadas acima são exploradas neste trabalho:

- No Ap. B.2 é apresentada uma família de operadores generalizados, construídos sobre uma proposta de extensão de conceitos contínuos para problemas de redes.

- No Ap. B.3, são apresentados operadores desenvolvidos para otimização de sistemas de distribuição de energia elétrica.

Estes operadores são aplicados em algoritmos evolucionários apresentados nos Caps. 4 e 5. Em ambos os casos, eles garantem a obtenção de redes factíveis, desde que partam de redes factíveis.

3.4.2 Uso de Codificações Específicas para Redes

Outra alternativa para manutenção da factibilidade da população é o uso de codificações específicas. Com estas codificações espera-se obter, ao menos na maior parte das vezes, soluções factíveis após o uso dos operadores, tornando o algoritmo capaz de tratar problemas de redes.

Estes esquemas de codificação usualmente são desenvolvidos para cada estrutura específica de rede, o que impede sua aplicação em outros tipos de problemas. Na seqüência, são discutidos cinco métodos clássicos, utilizados para codificação de árvores em algoritmos evolucionários.

Characteristic Vector:

A *Characteristic Vector* (Rothlauf, 2005) codifica as árvores em vetores binários de dimensão m. Cada componente do vetor representa uma possível aresta e pode assumir valor 0 (a aresta não existe) ou 1 (a aresta está presente na árvore). Esta codificação é não-polarizada e permite a representação de todas as árvores possíveis. Entretanto, a maior parte dos códigos possíveis representa redes infactíveis.

Prüfer Numbers:

A *Prüfer Numbers* (Prüfer, 1918) codifica as árvores em vetores inteiros de dimensão $n-2$, e cada componente assume valores entre 1 e n (onde n é o número de nós). Esta codificação é baseada no Teorema de Cayley (Teor. 1.3) e não apresenta polarização. Todo código representa uma árvore válida e toda árvore possui um código correspondente. Essa codificação só funciona adequadamente em problemas onde o espaço de busca é definido pelo o grafo completo.

Network Random Keys:

A *Network Random Keys* (Routhlauf et al., 2002) codifica as árvores em vetores reais de dimensão m. Cada componente do vetor representa o peso de sua respectiva aresta. A árvore correspondente é obtida utilizando um algoritmo *MST*, como Prim (Prim, 1957) ou Kruskal (Kruskal, 1956), para os pesos do vetor codificado. A codificação cobre todas as árvores possíveis e apresenta garantia de factibilidade. No entanto, a mesma apresenta alta redundância, uma vez que o número de códigos que representam uma mesma árvore é muito alto (idealmente infinito).

Edge Sets:

A *Edge Sets* (Raidl and Julstrom, 2003) codifica as árvores em vetores inteiros de dimensão $2n-2$, sendo que cada componente do vetor pode assumir valores entre 1 e n. Cada par de posições do vetor representa uma aresta na árvore. A codificação é redundante, não-polarizada e cobre todas as árvores possíveis. Apesar da taxa de infactibilidade desta codificação ser menor que a da *Characteristic Vector*, ela ainda é

um aspecto restritivo desta codificação. Mecanismos adicionais de reparação devem ser aplicados em problemas com grafos não-completos.

Link and Node Biased Encoding:

Na *Link and Node Biased Encoding* (Palmer and Kershenbaum, 1994) as árvores são codificadas em vetores reais de dimensão $n+m$. As posições do vetor representam os pesos dos n nós e m conexões. Estes pesos são utilizados para atualizar a matriz de custo original do problema. A rede é obtida através de um algoritmo MST para estes pesos modificados. A codificação é polarizada, redundante, e cobre todas as soluções possíveis. Por outro lado a mesma apresenta complexidade quadrática, uma vez que todos os pesos da matriz devem ser atualizados em cada codificação.

As descrições detalhadas destes métodos, incluindo os algoritmos de decodificação, podem ser encontradas nas respectivas referências. Um estudo comparativo destes métodos de codificação, incluindo o estudo das características importantes e a performance em um problema de otimização, é apresentado em Carrano et al. (2007b).

Um abordagem alternativa às duas apresentadas acima é o uso conjunto de codificações específicas e operadores desenvolvidos especificamente para estas codificações. Nestes casos pode-se contornar problemas recorrentes nas codificações, como infactibilidade, redundância e localidade através dos operadores utilizados. Este procedimento pode ser utilizado para obter vizinhanças com propriedades mais adequadas à exploração do espaço de busca utilizando o algoritmo evolucionário.

3.4.3 Controle Dimensional em Problemas de Redes

Outra dificuldade inerente ao problema de otimização de redes é o crescimento dimensional do problema, já discutido no Cap. 1. O número de conexões possíveis cresce com o quadrado do número de nós $\left(m = \frac{n \cdot (n-1)}{2}\right)$, o que dificulta a solução direta de problemas de grande dimensão.

Conhecimentos "a priori" do problema tratado podem ser aplicados para reduzir o

conjunto de variáveis do problema. Em problemas cujos custos das conexões, e conseqüentemente da rede, estão fortemente relacionados com a distância das mesmas, pode-se ignorar as conexões mais longas, sem grandes riscos de perda da solução ótima.

Abordagens encontradas na bibliografia, como Miranda et al. (1994); Cossi et al. (2005); Ramírez-Rosado and Bernal-Agustín (1998), estabelecem o conjunto de conexões possíveis do problema manualmente, excluindo conexões tidas como improváveis. No entanto esta técnica pode não ser a mais adequada, uma vez que, em problemas de grande dimensão, o conjunto de conexões é muito grande, tornando a não inclusão de uma conexão presente na rede ótima um evento provável.

Ao longo dessa seção serão apresentados quatro técnicas automatizadas para redução do conjunto de arestas do problema. Essas técnicas utilizam como único princípio básico a eliminação de conexões excessivamente longas. Obviamente, conhecimentos "a priori" sobre o problema podem ser aplicados, reduzindo ainda mais o número de variáveis do problema de otimização.

O resultado da aplicação dessas técnicas é um conjunto de arestas menor que o inicialmente considerado, o que tende a facilitar a execução de algoritmos evolucionários.

Geração do Conjunto de Arestas Utilizando o Raio de Abrangência

Apresentada por Soares (2001), esta técnica propõe a utilização de um raio que determina a abrangência de cada um dos nós do problema. O conjunto de nós a que cada nó i pode se conectar é determinado seguindo os passos apresentados abaixo (a Fig. 3.4 ilustra essa técnica).

Geração do Conjunto de Arestas por Raio de Abrangência

1: **for** i **from** 1 **to** n **do**
2: $cn_i \leftarrow \emptyset$;
3: **for** j **from** 1 **to** n **do**
4: **if** $no_i \neq no_j$ **then**

```
5:          if dist(no_i, no_j) ≤ R_ab then
6:              cn_i ← cn_i ⋃ no_j;
7:          end if
8:       end if
9:    end for
10: end for
```

Onde:
cn_i é o conjunto de nós a que o nó i pode se conectar;
$dist(no_i, no_j)$ é distância euclideana entre os nós i e j; e
R_{ab} é o raio de abrangência definido pelo projetista.

Apesar desse método de codificação reduzir sensivelmente o número de conexões em sistemas de grande porte, o mesmo apresenta dois problemas críticos:

- **Definição do raio de abrangência:** a escolha adequada do raio de abrangência (R_{ab}) é fundamental para a criação de um conjunto de arestas eficiente. A utilização de um raio excessivamente pequeno pode excluir conexões que estariam presentes na rede ótima, ou ainda criar problemas inválidos (com nós ilhados), conforme mostra a Fig. 3.5(a). Já a utilização de um raio de abrangência muito grande pode levar a um conjunto de variáveis muito próximo do grafo completo. A Fig. 3.5(b) apresenta um caso onde isso ocorre.

- **Interceptação de conexões:** da forma com que a técnica é definida, a mesma permite que conexões se interceptem em determinados pontos, como mostra a Fig. 3.5(c). Isto não é desejável, uma vez que essa interceptação não deve ocorrer em redes ótimas.

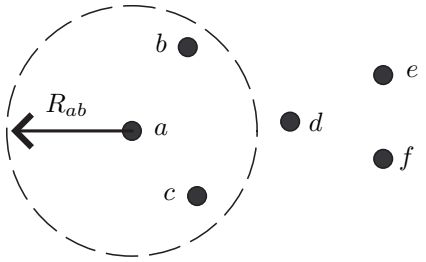


Figura 3.4: A circunferência de raio R_{ab} centrada em a determina que o mesmo só pode se conectar à b e c. O mesmo procedimento deve ser executado para cada nó para definir a codificação.

Geração do Conjunto de Arestas Utilizando a Controlled-Greedy Encoding

A *Controlled-Greedy Encoding* proposta por Carrano et al. (2006b) tem como objetivo atenuar as limitações apresentadas na técnica discutida acima.

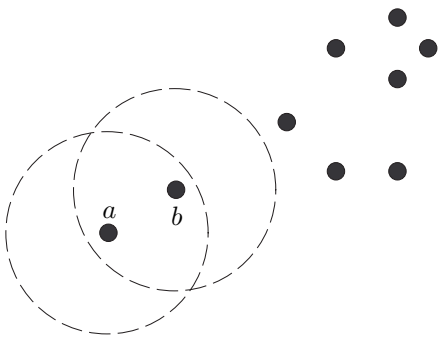
Sua construção é inspirada em algoritmos gulosos cuja idéia básica é conectar cada nó ao nó mais próximo, o que geralmente leva à ótimos locais. Para evitar isso buscou-se controlar a "gula" do algoritmo utilizando 2 parâmetros:

- mnv que é o número mínimo de nós a que cada nó pode se conectar;
- mxv que é o número máximo de nós a que cada nó pode se conectar.

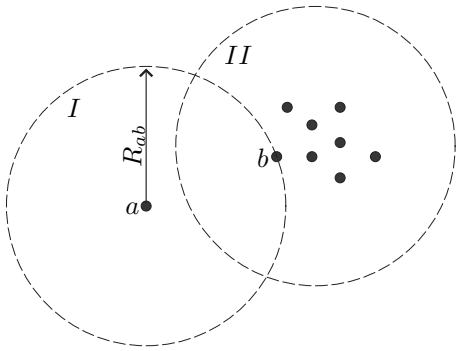
Para determinar o número de conexões de cada nó determina-se primeiramente a distância média de cada nó i (d_i), utilizando (3.5).

$$d_i = \frac{1}{n} \cdot \sum_{j=1}^{n} dist(i,j) \tag{3.5}$$

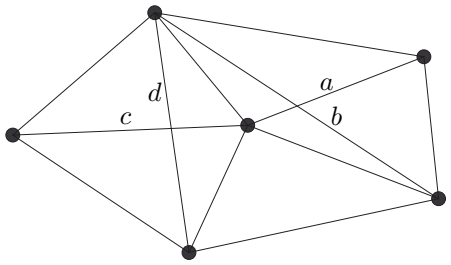
Assim, define-se que o nó com menor distância média pode se conectar com mxv nós, enquanto que nó com maior distância média pode se conectar com mnv nós. Para os nós intermediários, utiliza-se uma interpolação linear discretizada, como apresentado na Eq. (3.6).



(a) Neste caso o raio de abrangência definido é menor que o mínimo necessário. Com isso o conjunto de arestas resultante não é válido, uma vez que os nós a e b só se conectam entre si, não "chegando" aos outros nós.



(b) Neste caso o nó a só pode se conectar ao nó b, uma vez que o R_{ab} escolhido foi o menor admissível. Entretanto todos os outros nós se encontram internos à circunferência II, fazendo com que todas conexões entre os mesmos sejam consideradas.



(c) Nesta codificação as conexões a e b, c e d se interceptam, o que é indesejável e aumenta o número de variáveis consideradas.

Figura 3.5: Limitações da codificação por raio de abrangência

$$nc_i = \left\lfloor \left(\frac{mnv - mxv}{d_{max} - d_{min}} \right) \cdot (d_i + d_{min}) + mxv \right\rfloor \tag{3.6}$$

Onde:

$d_{min} = \min(d)$;

$d_{max} = \max(d)$.

A estrutura básica que descreve este método de redução dimensional é apresentada na seqüencia.

Controlled-Greedy Encoding

1: **for** i **from** 1 **to** n **do**
2: $cn_i \leftarrow \emptyset$;
3: $d_i \leftarrow \frac{1}{n_n} \cdot \sum_{j=1}^{n_n} dist(i,j)$;
4: $nc_i \leftarrow \left\lfloor \left(\frac{mnv - mxv}{d_{max} - d_{min}} \right) \cdot (d_i + d_{min}) + mxv \right\rfloor$;
5: **for** j **from** 1 **to** nc_i **do**
6: $cn_i \leftarrow cn_i \bigcup vz_j$;
7: **end for**
8: **end for**

Onde:
vz_j é o j-ésimo vizinho mais próximo do nó i.

Esta técnica elimina o problema de definição do raio de abrangência, e ainda ameniza a interceptação de conexões.

Geração do Conjunto de Arestas Utilizando a *Delaunay Encoding*

Outra forma viável de se controlar a dimensão do problema, é a utilização de métodos de triangulação, como a triangulação de Delaunay, descrita abaixo;

Dado um conjunto de n pontos, P, no plano, é possível definir uma *máxima subdivisão planar*, S, para o grafo planar tal que não possa ser acrescentada uma aresta conectando dois vértices a S sem que a mesma intercepte ao menos uma das arestas existentes, i.e., sem destruir a planaridade do grafo. Uma máxima subdivisão planar de um conjunto de vértices P define uma *triangulação* de P. A *triangulação de* Delaunay é um tipo especial de triangulação, e possui a seguinte definição:

Definição 3.1 (Triangulação de *Delaunay*) *Dado um conjunto de pontos P no plano e uma triangulação $\mathcal{T}$ de P, $\mathcal{T}$ é chamada triangulação de* Delaunay *se, e somente se,*

o circuncírculo de qualquer triângulo de $\mathcal{T}$ não contém nenhum ponto de P em seu interior (de Berg et al., 2000).

A Fig. 3.6 mostra a triangulação de *Delaunay* para um conjunto de 5 pontos no plano. Também é possível definir a triangulação de *Delaunay* como o dual do Diagrama de *Voronoi* (de Berg et al., 2000). Devido a suas características favoráveis, a triangulação de *Delaunay* vem sendo empregada em várias aplicações, como interpolação de funções, geração de malhas, reconstrução de formas, etc. Neste trabalho, a triangulação de *Delaunay* é utilizada para controle dimensional do problema de otimização de redes. Acredita-se que a principal vantagem dessa técnica nesse problema é a redução do número de variáveis proporcionada por ela, como estabelece o Teorema 3.1.

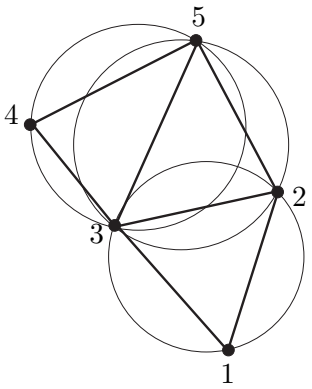


Figura 3.6: Triangulação de *Delaunay* de um conjunto de 5 pontos no plano

Teorema 3.1 *Seja P um conjunto de n pontos no plano, nem todos colineares, e seja k o número de pontos em P que pertencem a casca convexa de P. Então, qualquer triangulação de P possui $2n-2-k$ triângulos e $3n-3-k$ arestas.*

A prova do Teorema 3.1 é simples e pode ser encontrada em de Berg et al. (2000). É importante notar que o número de arestas presentes no grafo resultante da triangulação de *Delaunay* tem crescimento linear, enquanto que o grafo completo tem crescimento quadrático.

Outra propriedade importante da triangulação de Delaunay é definida pelo seguinte lema:

Lema 3.1 *A árvore geradora mínima de um conjunto de pontos P é um sub-grafo da triangulação de* Delaunay *(Shamos and Hoey, 1975).*

Prova do Lema 3.1

Dois pontos de P formam uma aresta do grafo de *Delaunay* de P se, e somente se, existe um disco fechado que contém os dois pontos em sua borda e não contém nenhum outro ponto de P (de Berg et al., 2000). Isto também é válido para os vértices da árvore geradora mínima (Eppstein, 1996).

Uma vez que as redes ótimas geralmente apresentam similaridades com a árvore geradora mínima, acredita-se que a triangulação de *Delaunay* não seja, na maior parte dos casos, prejudicial à solução do problema.

Por suas características de construção, esta técnica não apresenta nenhum dos problemas citados anteriormente. No entanto, podem aparecer algumas conexões extensas, que não ocorreriam nos outros métodos.

Capítulo 4

Projeto de Redes Utilizando Algoritmos com Inspiração Contínua

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

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

Estas operações baseadas nas propriedades do espaço vetorial garantem aos algoritmos evolucionários contínuos uma certa generalidade: o mesmo algoritmo pode ser aplicado em problemas com funções objetivo de diferentes características. A mesma

situação geralmente não ocorre em algoritmos evolucionários voltados a otimização discreta. Recentemente, vem sendo mostrado na literatura que algoritmos evolucionários desenvolvidos para aplicações generalizadas tendem a não apresentar boa performance em várias classes de problemas discretos: operadores específicos, que são construídos sobre a estrutura intrínseca do problema, são usualmente necessários para a construção de algoritmos com alta performance (Smith and Walters, 2000; Duan and Yu, 2002; Carrano et al., 2006b).

Este capítulo descreve uma técnica que "insere" problemas de redes no espaço vetorial de Hilbert $\mathbb{R}^n$. Essa "transformação" é baseada em uma definição adequada de norma, chamada aqui de *T-norm*, que é deduzida de um produto escalar, também definido convenientemente. A inserção do problema de redes no espaço contínuo permite definir entidades como posições relativas, distâncias, direções, vizinhanças, e assim por diante. As redes são interpretadas como pontos discretos do espaço $\mathbb{R}^n$. Estas entidades são utilizadas para construção de algoritmos evolucionários voltados a otimização da topologia de grafos em geral.

É importante destacar que existem outros trabalhos que estudam o estabelecimento de analogias entre os espaços contínuo e discreto. No entanto, estes trabalhos são voltados a contextos diferentes do apresentado aqui, como comparação de *strings* (Christodoulakis et al., 2005) e comparação de preferências em tomadas de decisão (Lafage and Lang, 2005). Galton (1999, 2000) propõe uma estrutura topológica para o espaço discreto e introduz o conceito de "deslocamentos aproximadamente contínuos" nestes espaços. Billera et al. (2001) apresenta uma caracterização geométrica para o espaço de árvores filogenéticas binárias.

4.1 Conceitos Contínuos em Projeto de Redes

Seja $G(V, A)$ um grafo não direcionado com b tipos de conexões distintos e ordenados (1 se refere a aresta mais fraca e b se refere a mais forte). Seja N um sub-grafo de G que contém necessariamente todos os vértices V. Tendo em conta os conceitos de espaços

métricos (Lima, 1995), é possível representar N como um vetor $\overrightarrow{N}$ embarcado no espaço $\mathbb{R}^m$ de Hilbert, como proposto em (4.1).

$$\overrightarrow{N} = \sum_{i=1}^{m} \left(w_i^N \cdot p_1 + N_i \cdot w_i^N \cdot p_2 \right) \overrightarrow{e_i} \tag{4.1}$$

onde:

w_i^N é o peso topológico da aresta i na rede N. Este peso é um real positivo se a conexão i existe ou 0 caso contrário;

p_1 é um fator constante, tal que: $p_1 \geq b$;

p_2 é 1 se o problema é *multibranch* e 0 se o problema é *monobranch*;

N_i é o tipo conexão da aresta i na rede N, tal que: $N_i \in [0, 1, \ldots, b]$;

$\overrightarrow{e_i}$ é o i-ésimo vetor da base canônica.

Esta expressão atribui uma dimensão no espaço a cada aresta possível no grafo. No caso de problemas monobranch, $b = 1$, o valor de p_2 é definido como $p_2 = 0$, e o valor de p_1 pode ser definido, sem perda de generalidade, como $p_1 = 1$, reduzindo a expressão (4.1) para:

$$\overrightarrow{N} = \sum_{i=1}^{m} w_i^N \cdot \overrightarrow{e_i} \tag{4.2}$$

Se a aresta i não existe, $w_i^N = 0$, e a i-ésima coordenada do espaço é nula. Quando a aresta i existe, w_i^N é definido de forma a representar "a importância topológica" da conexão i na rede N. Em árvores por exemplo, alterações próximas à raiz tendem a apresentar maior impacto na rede que alterações próximas às folhas. Portanto, uma conexão i próxima a raiz deve ser associada à um valor mais alto de w_i^N. Tendo em conta este aspecto, pode-se atribuir w_i^N com base na distância da conexão para a raiz, ou a distância para a folha mais distante, ou ainda o número de nós acima da conexão por exemplo.

As regras para definir os pesos das conexões devem ser estabelecidas tendo em conta

a estrutura topológica da rede. Uma regra simples, aplicável em qualquer estrutura, é utilizar w_i^N igual ao comprimento da conexão i. Esta regra é particularmente útil em problemas que apresentam uma alta dependência do comprimento das conexões, mas, no entanto, não leva em conta a estrutura da rede.

Na Eq. (4.1), o produto da constante p_2 pelo inteiro N_i expressa a força da conexão i (tipos de conexões mais fortes estão associados a valores maiores de N_i). A desigualdade $p_1 \geq b$ garante que $p_1 \geq p_2 \cdot N_i$. Isto significa que o efeito de introduzir ou remover uma conexão é maior que o efeito de mudar o tipo de conexão.

O valor resultante de $w_i^N \cdot (p_1 + p_2 \cdot N)$, que é atribuído à i-ésima coordenada do espaço, possui as seguintes características:

- Apresenta pequenas variações para mudanças no tipo de conexão;
- Apresenta grandes variações quando uma conexão é inserida ou removida;
- A variação é maior para conexões com peso topológico mais alto, i.e., para conexões que são "mais importantes" em uma dada topologia da rede.

4.1.1 Posições Relativas e Medidas de Distância em Problemas de Redes

Seja O uma rede sem nenhuma aresta, definida no mesmo grafo $G(V, A)$, que representa a origem do espaço das redes. É possível constatar que esta rede é representada pelo vetor nulo no espaço $\mathbb{R}^m$, como:

$$\overrightarrow{O} = \sum_{i=1}^{m} (0 \cdot p_1 + 0 \cdot 0 \cdot p_2)\, \overrightarrow{e_i} = \sum_{i=1}^{m} 0 \overrightarrow{e_i} \tag{4.3}$$

De (4.1) e (4.3), é possível notar que todas as redes (factíveis ou infactíveis) representáveis em $G(V, A)$ estão contidas em um hipercubo de dimensão m, com hiperfaces $(p_1 + b \cdot p_2) \times (p_1 + b \cdot p_2)$ (assumindo pesos topológicos normalizados entre 0 e 1).

A Eq. (4.1) insere o problema de redes no espaço $\mathbb{R}^m$, fazendo com que ele "herde" as propriedades deste espaço.

Sejam A e B duas redes quaisquer, tais que:

$$\begin{aligned} \overrightarrow{A} &= \sum_{i=1}^{m} \left(w_i^A \cdot p_1 + A_i \cdot w_i^A \cdot p_2\right) \overrightarrow{e_i} &= \sum_{i=1}^{m} c_i^{AO} \cdot \overrightarrow{e_i} \\ \overrightarrow{B} &= \sum_{i=1}^{m} \left(w_i^B \cdot p_1 + B_i \cdot w_i^B \cdot p_2\right) \overrightarrow{e_i} &= \sum_{i=1}^{m} c_i^{BO} \cdot \overrightarrow{e_i} \end{aligned}$$

Por conseqüência, a posição relativa de A em relação a B é definida por:

$$\begin{aligned} \overrightarrow{r_p(A,B)} &= \sum_{i=1}^{m} \left[p_2 \left(A_i \cdot w_i^A - B_i \cdot w_i^B\right) + p_1 \left(w_i^A - w_i^B\right)\right] \overrightarrow{e_i} \\ &= \sum_{i=1}^{m} c^{AB} \cdot \overrightarrow{e_i} \end{aligned} \tag{4.4}$$

É importante notar que $\overrightarrow{A}$ também pode ser visto como a posição relativa de A em relação a O, ou simplesmente $\overrightarrow{A} = \overrightarrow{r_p(A,O)}$.

Deve-se destacar que, apesar de não existir correspondência entre vetores com coordenadas negativas e redes realizáveis, essas entidades existem espaço vetorial embarcado R^m, e são importantes para definir as operações de otimização propostas neste trabalho. É fácil verificar que o conjunto de vetores definidos por (4.1), com a origem definida em (4.3), e posição relativa definida (4.4), cumpre com as propriedades de espaços vetoriais (Tab. 4.1).

Tabela 4.1: Propriedades de espaços vetoriais

P.1	$\overrightarrow{u} + \overrightarrow{v} = \overrightarrow{v} + \overrightarrow{u}$
P.2	$\overrightarrow{u} + (\overrightarrow{v} + \overrightarrow{w}) = (\overrightarrow{u} + \overrightarrow{v}) + \overrightarrow{w}$
P.3	$\overrightarrow{u} + \overrightarrow{O} = \overrightarrow{u}$
P.4	$\overrightarrow{u} + (-\overrightarrow{u}) = \overrightarrow{O}$
P.5	$k \cdot (l \cdot \overrightarrow{u}) = k \cdot l \cdot (\overrightarrow{u})$
P.6	$l \cdot (\overrightarrow{u} + \overrightarrow{v}) = l \cdot \overrightarrow{u} + l \cdot \overrightarrow{v}$
P.7	$(k + l) \cdot \overrightarrow{v} = k \cdot \overrightarrow{v} + l \cdot \overrightarrow{v}$
P.8	$1 \cdot \overrightarrow{u} = \overrightarrow{u}$

Um produto escalar dos vetores A e B pode ser definido por (4.5):

$$\overrightarrow{A} \cdot \overrightarrow{B} = \sum_{i=1}^{m} c_i^{AO} \cdot c_i^{BO} \tag{4.5}$$

Este produto escalar define uma norma:

$$\overrightarrow{A} \cdot \overrightarrow{A} = \| \overrightarrow{A} \|^2 \tag{4.6}$$

que, por sua vez, define uma distância:

$$\begin{aligned} d_s(A,B) = \|\overrightarrow{(A,B)}\| &= \sqrt{\sum_{i=1}^{m} \left[p_2 \left(A_i \cdot w_i^A - B_i \cdot w_i^B \right) + p_1 \left(w_i^A - w_i^B \right) \right]^2} \\ &= \sqrt{\sum_{i=1}^{m} \left(c_i^{AB} \right)^2} \end{aligned} \tag{4.7}$$

que é referida neste trabalho como *T-norm*.

O produto escalar (4.5) cumpre com as propriedades de produto escalar (Tab. 4.2) e os Teoremas 4.1 e 4.2, como mostrado na seqüência.

Tabela 4.2: Propriedades de produtos escalares

P.1	$\overrightarrow{u} \cdot \overrightarrow{v} = \overrightarrow{v} \cdot \overrightarrow{u}$
P.2	$(\overrightarrow{u} + \overrightarrow{v}) \cdot \overrightarrow{w} = \overrightarrow{u} \cdot \overrightarrow{w} + \overrightarrow{v} \cdot \overrightarrow{w}$
P.3	$(l \cdot \overrightarrow{u}) \cdot \overrightarrow{v} = l \cdot (\overrightarrow{u} \cdot \overrightarrow{v})$
P.4	$\overrightarrow{u} \cdot \overrightarrow{u} \geq 0$, sendo 0 somente se $\overrightarrow{u} = \overrightarrow{O}$

Teorema 4.1 *Uma operação define um produto escalar se:*

$$\overrightarrow{u} \cdot \overrightarrow{v} = \frac{1}{4} \cdot \| \overrightarrow{u} + \overrightarrow{v} \| - \frac{1}{4} \cdot \| \overrightarrow{u} - \overrightarrow{v} \|$$

Teorema 4.2 *(4.5) define um produto escalar.*

Prova do Teorema 4.2

$$\begin{aligned}
\overrightarrow{A} \cdot \overrightarrow{B} &= \frac{1}{4}\left(\sqrt{\sum_{i=1}^{m}\left(c_i^{AO}+c_i^{BO}\right)^2}\right)^2 - \frac{1}{4}\left(\sqrt{\sum_{i=1}^{m}\left(c_i^{AO}-c_i^{BO}\right)^2}\right)^2 \\
&= \frac{1}{4}\sum_{i=1}^{m}\left[\left(c_i^{AO^2}+2c_i^{AO}c_i^{BO}+c_i^{BO^2}\right)-\left(c_i^{AO^2}-2c_i^{AO}c_i^{BO}+c_i^{BO^2}\right)\right] \\
&= \frac{1}{4}\sum_{i=1}^{m} 4c_i^{AO}c_i^{BO} \\
&= \sum_{i=1}^{m} c_i^{AO}\ c_i^{BO}
\end{aligned}$$

Finalmente, de (4.7), é fácil perceber que esta medida de distância cumpre com as propriedades **P.1**, **P.2** e **P.3** de medidas de distância em espaços vetoriais (Tab. 4.3).

Tabela 4.3: Propriedades de medidas de distância em espaços vetoriais

P.1	$d_s(\overrightarrow{u}, \overrightarrow{v}) \geq 0$
P.2	$d_s(\overrightarrow{u}, \overrightarrow{v}) = 0$ se, e somente se $\overrightarrow{u} = \overrightarrow{v}$
P.3	$d_s(\overrightarrow{u}, \overrightarrow{v}) = d_s(\overrightarrow{v}, \overrightarrow{u})$
P.4	$d_s(\overrightarrow{u}, \overrightarrow{v}) \leq d_s(\overrightarrow{u}, \overrightarrow{w}) + d_s(\overrightarrow{w}, \overrightarrow{v})$

Entretanto (4.7) pode ser considerada uma medida de distância válida, se, e somente se, cumprir as quatro propriedades simultaneamente, incluindo a desigualdade triangular (**P.4**).

De Anton and Rorres (2000), é possível afirmar que:

$$\|\overrightarrow{u} + \overrightarrow{v}\|^2 \leq (\|\overrightarrow{u}\| + \|\overrightarrow{v}\|)^2 \tag{4.8}$$

é equivalente à desigualdade triangular.

De (4.5) e (4.8), é possível provar que (4.7) cumpre com a desigualdade triangular, como mostrado abaixo.

Prova que (4.7) cumpre com a Desigualdade Triangular

Sejam $\overrightarrow{u} = \overrightarrow{A} - \overrightarrow{C}$ e $\overrightarrow{v} = \overrightarrow{C} - \overrightarrow{B}$ respectivamente (onde C é uma rede qualquer):

$$\begin{aligned}
\left\|\overrightarrow{A} - \overrightarrow{C} + \overrightarrow{C} - \overrightarrow{B}\right\|^2 &\leq \left(\left\|\overrightarrow{A} - \overrightarrow{C}\right\| + \left\|\overrightarrow{C} - \overrightarrow{B}\right\|\right)^2 \\
\left\|\overrightarrow{A} - \overrightarrow{C} + \overrightarrow{C} - \overrightarrow{B}\right\| &\leq \left\|\overrightarrow{A} - \overrightarrow{C}\right\| + \left\|\overrightarrow{C} - \overrightarrow{B}\right\| \\
\left\|\overrightarrow{A} - \overrightarrow{B}\right\| &\leq \left\|\overrightarrow{A} - \overrightarrow{C}\right\| + \left\|\overrightarrow{C} - \overrightarrow{B}\right\| \\
\sqrt{\sum_{i=1}^{m}\left(c_i^{AB}\right)^2} &\leq \sqrt{\sum_{i=1}^{m}\left(c_i^{AC}\right)^2} - \sqrt{\sum_{i=1}^{m}\left(c_i^{CB}\right)^2} \\
\sqrt{\sum_{i=1}^{m}\left(c_i^{AB}\right)^2} &\leq \sqrt{\sum_{i=1}^{m}\left(c_i^{AC}\right)^2} - \sqrt{\sum_{i=1}^{m}\left(-c_i^{BC}\right)^2} \\
\sqrt{\sum_{i=1}^{m}\left(c_i^{AB}\right)^2} &\leq \sqrt{\sum_{i=1}^{m}\left(c_i^{AC}\right)^2} - \sqrt{\sum_{i=1}^{m}\left(c_i^{BC}\right)^2} \\
d_s(A,B) &\leq d_s(A,C) + d_s(B,C)
\end{aligned}$$

Então, é possível dizer que:

Teorema 4.3 *(4.7) é uma medida de distância válida.*

É importante notar que a distância Hamming (descrita abaixo) usualmente empregada para comparação de *strings*, pode ser obtida como um caso particular da *T-norm*. Para isso, basta definir o "espaço vetorial de Hamming" usando a Eq. (4.1) com $p_1 = 1$, $p_2 = 0$ e $w_i^N = 1 \ \forall \ i$. A distância Hamming é obtida com a norma-1 do vetor em tal espaço. Esta observação sugere que (4.1) pode ser interpretado com uma transformação de coordenadas do "espaço vetorial de Hamming".

Distância Hamming

Sejam duas strings, s_a e s_b, de mesmo comprimento l. A distância Hamming entre s_a e s_b é definida pelo número de alterações necessárias para transformar s_a em s_b, ou vice-versa (Hamming, 1950, 1980). Como exemplo, a distância Hamming entre:

- 1011101 e 1001001 é 2;

- 2143896 e 2233796 é 3;
- "toned" e "roses" é 3.

A distância Hamming entre dois grafos pode ser definida como o número de alterações (inserção e remoção de arestas) que devem ser feitas em um dos grafos para se alcançar o outro. Esta métrica é usualmente utilizada para comparação de soluções em problemas discretos.

4.1.2 Pesos Topológicos em Árvores

Os conceitos propostos nas seções anteriores são gerais, e podem ser empregados em qualquer estrutura de grafos, desde que seja escolhido um método adequado para definição dos pesos topológicos. Nesta seção são apresentadas algumas alternativas para determinação dos pesos topológicos em árvores com raiz.

Como discutido na Sec. 4.1, é razoável assumir que, em árvores com raiz, as mudanças próximas à raiz tem maior impacto que mudanças próximas as folhas. Neste caso, o peso topológico pode ser estimado através da distância entre os extremos da conexão e a raiz, como mostrado em (4.9) e (4.10).

$$w_i^N = \frac{s_a^N + s_b^N}{2} \tag{4.9}$$

$$s_x^N = 1 - \frac{{d_{x,root}}^N}{\max_j \left({d_{j,root}}^N\right)} \tag{4.10}$$

onde:

a e b são os extremos da conexão i na rede N;

s_x^N é o peso do nó x na rede N;

${d_{x,root}}^N$ é a distância total do caminho $x - root$ na rede N;

$\max_j \left({d_{j,root}}^N\right)$ é a máxima distância entre os caminhos $j - root$, $\forall\ j \in [1, \ldots, n]$.

As Eq. (4.9) e (4.10) deixam claro que o peso topológico das conexões varia entre 0 e 1. A raiz (*root*) recebe peso topológico máximo, 1, e a folha mais distante recebe peso topológico mínimo, 0.

É importante destacar que diferentes métodos podem ser aplicados a problemas com características particulares. O método apresentado acima é geral, e pode ser aplicado à qualquer problema de projeto de árvores.

4.2 Operações no Espaço Vetorial

Nesta seção são descritas algumas operações típicas de espaços vetoriais contínuos, que podem ser definidas adequadamente para redes, com o espaço vetorial de redes proposto neste capítulo.

4.2.1 Interpolação em Linha e Busca Unidimensional

O conceito de posição relativa apresentado em (4.4) pode ser usado para definir uma "interpolação em linha" em conjuntos de redes. Sejam S e D duas redes quaisquer definidas em $G(V, A)$ (S é a rede de partida e D é a rede direção). É possível gerar redes correspondentes a pontos sobre uma "trajetória aproximada", que começa em S e termina em D. O problema é definido por:

- Considerando um dado escalar $\gamma \in [0, 1]$ e uma tolerância $\epsilon > 0$;
- Busca-se uma rede R, associada ao vetor $\overrightarrow{R}$ que é aproximadamente situado na posição espacial $\overrightarrow{P} = \overrightarrow{S} + \gamma \cdot (\overrightarrow{S, D})$, tal que: $\|\overrightarrow{(R, P)}\| \leq \epsilon$.

Tal rede pode ser gerada através do seguinte procedimento:

1. Definir o contador $j = 1$, e a rede resultante atual $R_j = S$;
2. Encontrar uma nova rede R_{j+1} que resulta de mudanças de uma ou mais conexões de R_j, de forma que $\|\overrightarrow{(R_{j+1}, P)}\| < \|\overrightarrow{(R_j, P)}\|$ e R_{j+1} mantenha factbilidade;

3. Se $\|\overrightarrow{(R_{j+1}, P)}\| < \epsilon$, ir para o passo 4. Caso contrário, fazer $j = j + 1$ e ir para o passo 2;

4. Se o problema é *monobranch*, fazer $R = R_{j+1}$ e parar;

5. Se o problema é *multibranch*, encontrar uma rede R_{j+2} que tem a mesma topologia de R_{j+1}, mas têm tipos de conexões tais que $\|\overrightarrow{(R_{j+2}, P)}\|$ atinge o mínimo valor possível, com a topologia da rede fixa e os tipos de conexões variáveis. Fazer $R = R_{j+2}$ e parar.

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

Deve-se notar que, neste procedimento, a rede atual R_j não caminha na direção de D sobre uma linha. R_j caminha sobre dois conjuntos (ou hipercones) S_S e S_D. Cada um destes conjuntos é composto por todas as redes que podem ser obtidas de S à uma distância específica, de forma que a distância para S aumenta e a distância para D diminui. Uma vez que existe uma direção que deve ser seguida, o conjunto de redes alcançáveis deve ser reduzido à interseção dos conjuntos $(S_S \bigcap S_D)$, que são as redes que podem ser construídas apenas com as conexões de $S \bigcup D$. A Fig. 4.1 ilustra esse procedimento.

A operação de interpolação descrita acima torna possível a implementação de métodos otimização unidimensional. Esse procedimento pode ser implementado utilizando o algoritmo de seção áurea proposto na Sec. 2.2.1.

Assim como no espaço de variáveis contínuas, esta otimização unidimensional garante encontrar a rede ótima R no segmento que une S e D, dado que a função objetivo

$f(R)$ é unimodal neste segmento. No caso de comportamento não-unimodais, os limites unimodais máximo e mínimo definem os limites de erro na minimização.

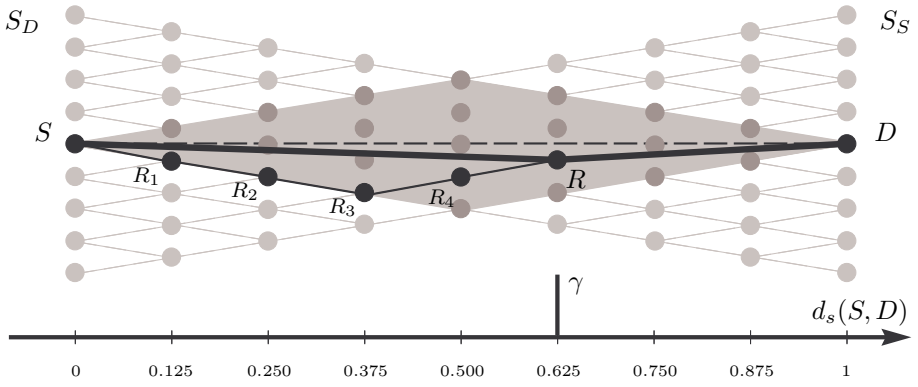


Figura 4.1: Interpolação de redes

4.2.2 Vizinhança e Busca Local

O conceito de distância imediatamente induz uma vizinhança, que pode ser definida como:

$$\mathcal{V}(\overrightarrow{X_0}, \epsilon) = \left\{ X \quad | \quad \|\overrightarrow{(X_0, X)}\| < \epsilon \right\} \tag{4.11}$$

em torno do vetor $\overrightarrow{X_0}$ que representa a rede X_0, com raio ϵ. A operação de busca local pode ser definida como a busca dentro de uma dada vizinhança.

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

1. Determinar qual é o conjunto de conexões que podem ser removidos sem sair da vizinhança;

2. Escolher aleatoriamente uma conexão deste conjunto e remover, realizando na seqüência uma operação capaz de restaurar a factibilidade da rede;

3. Avaliar a função objetivo da nova rede;

4. Se a nova rede é melhor que a anterior, então redefinir o centro da vizinhança para esta rede e ir para o passo 1. Caso contrário, se existirem conexões da vizinhança que ainda não foram testadas, ir para o passo 2. Senão, ir para o passo 5;

5. Se o problema é *monobranch*, parar. Caso contrário, determinar o conjunto de conexões que podem ter seus tipos modificados, sem sair da vizinhança estabelecida;

6. Escolher aleatoriamente um ramo neste conjunto e mudar seu tipo;

7. Se a nova rede é melhor que a anterior, redefinir o centro da vizinhança para esta rede e ir para o passo 5. Caso contrário, se existem conexões na vizinhança que ainda não foram testadas, então ir para o passo 6. Caso esse conjunto seja vazio, então parar.

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

4.2.3 Pontos Aleatórios a Distâncias Pré-definidas

A geração aleatória de um ponto que se encontra a uma distância pré-definida de um dado ponto é uma operação fundamental em vários algoritmos estocásticos. Seja uma rede inicial P e uma distância requerida γ. Esta operação pode ser executada através do seguinte procedimento:

1. Definir $\overrightarrow{R_x} \leftarrow \overrightarrow{P}$;

2. Determinar o conjunto de conexões de R_x que podem ser removidas tal que a rede resultante R (após ser corrigida para manutenção da factibilidade) cumpra $\|\overrightarrow{(R,P)}\| \leq \gamma$;

3. Se esse conjunto não é vazio, ir para o passo 4; caso contrário ir para o passo 5;

4. Escolher aleatoriamente uma conexão deste conjunto e remover de R_x (corrigindo a factibilidade) e copiar o resultado para R. Ir para o passo 2;

5. Determinar o conjunto de conexões de R_x que podem ter seus tipos mudados, de forma que a rede resultante R cumpra $\|\overrightarrow{(R,P)}\| \approx \gamma$. Realizar estas alterações.

Este procedimento é ilustrado na Fig. 4.2.

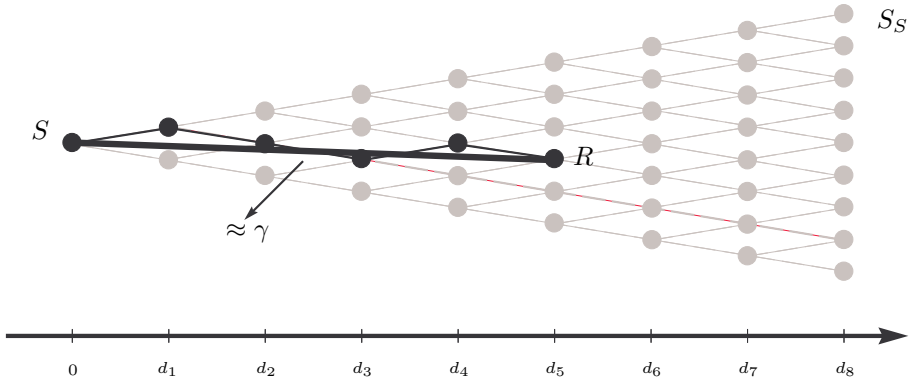


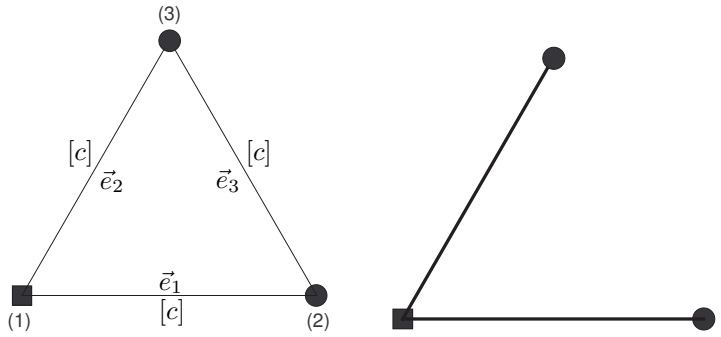
Figura 4.2: Geração de redes a distâncias pré-definidas

4.3 Exemplos

Sistema de 3 Nós

Seja $\mathcal{S}$ o sistema mono-branch de 3 nós e 3 conexões possíveis apresentado na Fig. 4.3(a). A, B e C (Fig. 4.3(b), 4.3(c) e 4.3(d) respectivamente) são as três árvores que podem ser encontradas em $\mathcal{S}$.

De (4.1), (4.9) e (4.10), é possível determinar os vetores que representam A, B e C no espaço $\mathbb{R}^3$, e suas posições relativas, como mostrado em (4.12) e (4.13). A Fig. 4.4 mostra estes vetores.



(a) Variáveis (b) Rede A

(c) Rede B (d) Rede C

Figura 4.3: Sistema de 3 nós

$$\begin{aligned} \overrightarrow{A} &= 0.50\overrightarrow{e_1} + 0.50\overrightarrow{e_2} + 0.00\overrightarrow{e_3} \\ \overrightarrow{B} &= 0.75\overrightarrow{e_1} + 0.00\overrightarrow{e_2} + 0.25\overrightarrow{e_3} \\ \overrightarrow{C} &= 0.00\overrightarrow{e_1} + 0.75\overrightarrow{e_2} + 0.25\overrightarrow{e_3} \end{aligned} \tag{4.12}$$

$$\begin{aligned} \overrightarrow{r_p(A,B)} &= -0.25\overrightarrow{e_1} + 0.50\overrightarrow{e_2} - 0.25\overrightarrow{e_3} \\ \overrightarrow{r_p(A,C)} &= 0.50\overrightarrow{e_1} - 0.25\overrightarrow{e_2} - 0.25\overrightarrow{e_3} \\ \overrightarrow{r_p(B,C)} &= 0.75\overrightarrow{e_1} - 0.75\overrightarrow{e_2} + 0.00\overrightarrow{e_3} \end{aligned} \tag{4.13}$$

A Eq. (4.7) pode ser usada para encontrar um escalar que mede a distância entre as redes A, B e C, como mostrado em (4.14).

$$\begin{aligned} d_s(A,B) &= 0.61 \\ d_s(A,C) &= 0.61 \\ d_s(B,C) &= 1.06 \end{aligned} \tag{4.14}$$

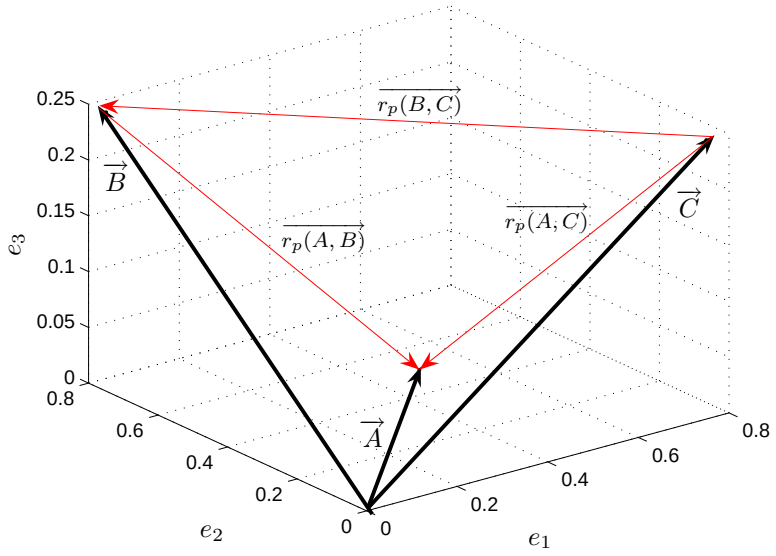


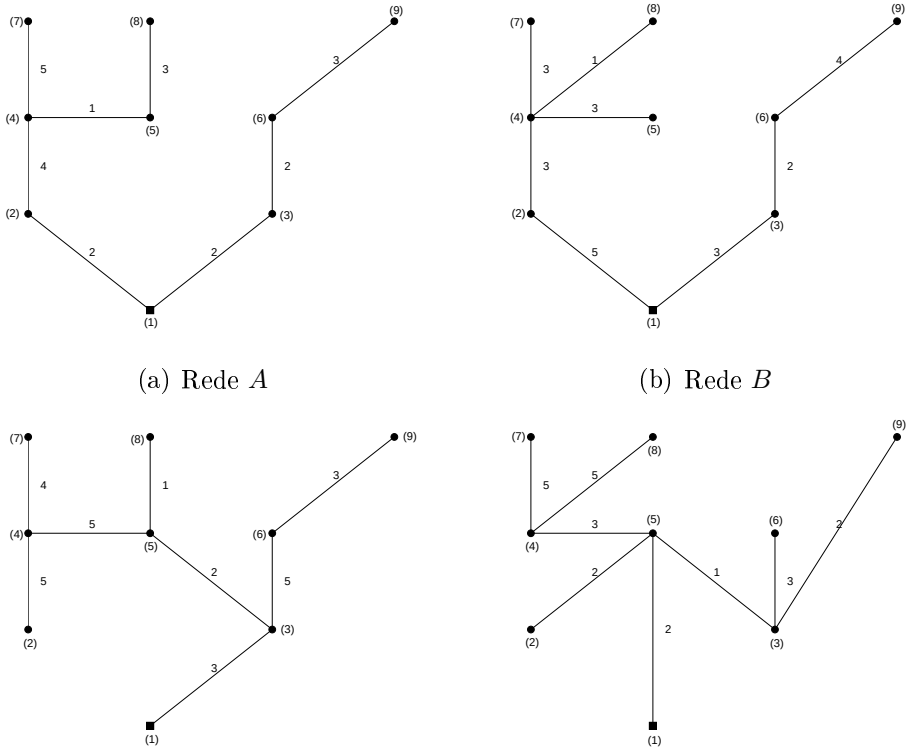
Figura 4.4: Sistema de 3 Nós - Representação dos vetores em $\mathbb{R}^3$

Sistema de 9 Nós

Sejam A, B, C e D quatro das redes que podem ser obtidas em um sistema *multi-branch* de 9 nós (Fig. 4.5(a), 4.5(b), 4.5(c) e 4.5(d) respectivamente). De (4.7) é possível encontrar as distâncias entre estas redes, as quais são mostradas na Tab 4.4[1]. Deve-se notar que, apesar da distância *Hamming* entre A e B ser igual a distância *Hamming* entre A e C, para a *T-norm*, a distância entre A e C se torna maior, pois a modificação ocorre próxima a raiz. Também é possível concluir que a métrica apresenta um comportamento esperado para grandezas deste tipo:

[1]Neste exemplo as distâncias foram normalizadas. A normalização das distâncias é empregada em todos os casos práticos tratados neste trabalho, uma vez que permite a comparação em problemas de diferentes dimensões.

- A distância entre A e D é próxima a distância entre B e D, uma vez que A e B são redes próximas;
- A distância entre A e C é menor que a distância entre B e C, uma vez que A está mais próxima de C que B;
- A distância entre A e C, adicionada à distância entre C e D é maior que a distância entre A e D.



(a) Rede A (b) Rede B

(c) Rede C (d) Rede D

Figura 4.5: Sistema de 9 nós

O conceito de interpolação proposto na Sec. 4.2.1 também é ilustrado neste sistema. Foi requerida uma rede que se encontra à meia distância entre A e D ($f = 0.5$, conseqüentemente $\|\overrightarrow{(R,A)}\| = f \cdot \|\overrightarrow{(A,D)}\| = (1-f) \cdot \|\overrightarrow{(A,D)}\| = 0.5$. A rede encontrada ($R$) é apresentada na Fig. 4.6. A mesma se encontra às seguintes distâncias: $\|\overrightarrow{(R,A)}\| = 0.59$ e $\|\overrightarrow{(R,D)}\| = 0.57$.

Tabela 4.4: Sistema de 9 nós - Distância entre as redes

	A	**B**	**C**	**D**
A	-	0.3467	0.5127	1.0000
B	0.3467	-	0.6598	0.9635
C	0.5127	0.6598	-	0.8981
D	1.0000	0.9635	0.8981	-

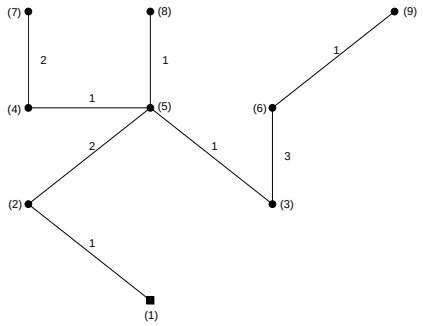


Figura 4.6: Sistema de 9 nós - Rede R

4.4 Algoritmos Evolucionários Desenvolvidos

Os conceitos apresentados nessa seção foram utilizados na construção de dois algoritmos evolucionários:

- Algoritmo Genético Mono-objetivo (*GANet*);
- Algoritmo de Seleção Clonal Mono-objetivo (*ClonalNet*).

Antes de discutir estes algoritmos, é importante apresentar os operadores que foram utilizados como base para suas construções.

4.4.1 Operadores

Os conceitos apresentados na Sec. 4.1 foram incorporados aos algoritmos evolucionários através de operadores de geração de indivíduos, mutação e cruzamento. Como discutido na Sec. 3.4.1 esses operadores garantem a factibilidade da população, eliminando a avaliação desnecessária de estruturas infactíveis.

Mais uma vez, é importante salientar que, como estes operadores foram desenvolvidos sob conceitos gerais, os mesmos podem ser aplicados, ou estendidos, a qualquer classe de problemas de grafos. Uma descrição esquemática destes operadores é apresentada no Ap. B.

Geração de Indivíduos

Foi desenvolvido um operador para geração aleatória de indivíduos:

***KruskGenInd*:** O *KruskGenInd* gera árvores aleatórias através do Algoritmo de Kruskal (Kruskal, 1956). Para garantir a aleatoriedade, são utilizados pesos aleatórios na execução do algoritmo.

Este operador é usado tanto no GA, para geração da população inicial, quanto no Clonal, para geração da população inicial e inserção periódica de diversidade na população. O *Algoritmo de Kruskal* foi implementado utilizando a estrutura de dados *Union-Find with Path Compression and Union by Rank* para conjuntos disjuntos (Cormem et al., 2001). Esta estrutura reduz o custo computacional de cada passo do algoritmo para $O(\alpha(N))$[2], que é aproximadamente constante. Isso faz com que o custo computacional total do algoritmo seja aproximadamente linear.

O uso de códigos Cayley (Cayley, 1889) também constitui uma alternativa viável para a geração da população inicial em problemas de árvores. A principal vantagem destes métodos é a garantia de uma amostragem uniforme do espaço. No entanto, os mesmos não foram utilizados neste trabalho porque só funcionam em grafos completos, o que impediria sua aplicação em parte dos casos apresentados, que empregam grafos esparsos.

[2] $\alpha(N)$ é a inversa da função $A(n,n)$, onde $A(n,n)$ é a função de Ackermann, que tem crescimento extremamente rápido.

Mutação

Dois operadores de mutação foram desenvolvidos:

***AnyDstMut*:** Dados uma rede pai P e um raio de mutação base r_b, o *AnyDstMut* gera um raio de mutação[3] $r_m = \text{rand} \cdot r_b$ e encontra uma rede que diste de P em aproximadamente r_m.

***LocSeaMut*:** Dados uma rede pai P e um raio de mutação base r_b, o *LocSeaMut* encontra k redes, com distância entre 0 a r_b ($r_m = \text{rand} \cdot r_b$), as avalia e escolhe a melhor dentre elas.

Percebe-se claramente que ambos os operadores são baseados nas operações vetoriais de geração de pontos a distâncias pré-definidas e busca local, apresentadas nas Secs. 4.2.3 e 4.2.2 respectivamente. Ambos os operadores podem ser vistos como adaptações discretas de operadores de mutação contínuos, como os apresentados nas referências (Ramos et al., 2003; Campelo et al., 2005).

Cruzamento

Dois operadores de cruzamento foram implementados:

***DirecCrOv*:** Dados duas redes pais P_1 e P_2, e dois fatores de distância f_1 e f_2, o *DirecCrOv* encontra duas redes: O_1 que se encontra a aproximadamente $f_1 \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_1 e $(1 - f_1) \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_2; e O_2 que se encontra a aproximadamente $(1 - f_2) \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_1 e $f_2 \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_2.

***GoldSecCrOv*:** Dados duas redes pais P_1 e P_2, e um fator de distância f, o *GoldSecCrOv* encontra duas redes: O_1 que é obtida utilizando o *GoldSecMut* a partir de P_1 na direção de P_2; e O_2 que se encontra a aproximadamente $(1 - f) \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_1 e $f \cdot \|\overrightarrow{(P_1, P_2)}\|$ de P_2.

[3] rand é uma variável aleatória com distribuição uniforme.

Ambos os operadores são baseados na operação de interpolação apresentada na Sec. 4.2.1, e podem ser vistos como extensões discretas de operadores propostos para genótipos contínuos, como o cruzamento real-polarizado (Ramos et al., 2003). Estes operadores são utilizados apenas no GA, uma vez que o Clonal não depende de operações de cruzamento.

4.4.2 *GANet*

Os operadores apresentados acima foram utilizados na construção de um algoritmo genético mono-objetivo, voltado a otimização de redes. Todos os operadores foram utilizados no algoritmo, sendo a escolha do operador a ser utilizado em cada operação realizada através de uma variável aleatória uniforme.

A estrutura geral do algoritmo é a mesma apresentada na Sec. 3.1.1. A única operação adicional inserida é uma busca local em torno melhor indivíduo encontrado até o momento, que deve ser executada periodicamente pelo algoritmo (a cada k gerações, onde k é definido pelo projetista). A pressão seletiva é controlada pelo método de *Ranking Linear* e a seleção é feita pelo *SUS*.

O *GANet* é utilizado no estudo comparativo apresentado na Sec. 4.5 deste capítulo e na solução dos sistemas apresentados nas Secs. 5.2 e 5.3 do Cap. 5.

4.4.3 *ClonalNet*

Assim como no GA, os operadores discutidos na Sec. 4.4.1 foram utilizados na construção de um algoritmo de seleção clonal mono-objetivo, voltado à otimização de redes. Este algoritmo apresenta uma estrutura muito semelhante à apresentada na Sec. 3.1.2, o que só foi possível devido à definição da métrica de redes proposta neste capítulo. A substituição da norma Euclideana pela *T-norm* (Eq. (4.7)) e o uso do operador *AnyDstMut* foram as únicas adaptações requeridas para construção do algoritmo.

O *ClonalNet* foi aplicado na solução do sistema teste apresentado na Sec. 5.2 e no mapeamento de soluções robustas para o sistema apresentado na Sec. 5.3, ambos

tratados no Cap. 5. Neste último caso, o conjunto de soluções sub-ótimas obtidas pelo algoritmo é usado para dar mais suporte à decisão em um problema com variáveis que envolvem incertezas.

4.5 Resultados

[4]O *GANet*, apresentado na seção anterior, foi testado em dois problemas:

- Instâncias aleatórias de 10 nós (45 variáveis), 25 nós (300 variáveis) e 50 nós (1.225 variáveis) do *OCST*;
- Instâncias aleatórias de 10 nós (45 variáveis), 25 nós (300 variáveis) e 50 nós (1.225 variáveis) do *q-MST*.

Os resultados obtidos pelo algoritmo foram comparados com outros seis algoritmos genéticos, testados para os mesmos problemas em (Carrano et al., 2007b). Cinco destes algoritmos são baseados nos mecanismos de codificação apresentados Sec. 3.4.2:

- Characteristic vector;
- Prüfer numbers;
- Network random keys;
- Edge sets;
- Node biased encoding;
- Kruskal operators.

O último algoritmo utiliza dois operadores desenvolvidos para árvores, um de cruzamento e um de mutação, também propostos em Carrano et al. (2007b). Estes operadores têm como idéia central o Algoritmo de Kruskal, e dão nome ao algoritmo: *KrukalGA*. A descrição esquemática desses operadores é apresentada no Ap. B.

[4]Os resultados apresentados nesta seção foram retirados das referências (Carrano et al., 2007f,b).

A metodologia adotada para comparação dos algoritmos testados é descrita detalhadamente na seqüência.

4.5.1 Metodologia de Comparação dos Algoritmos

A metodologia de comparação proposta neste trabalho é baseada na abordagem multi-objetivo de avaliação de algoritmos proposta em (Takahashi et al., 2003). A referência propõe a avaliação dos algoritmos tendo em conta dois critérios:

- Taxa de falha do algoritmo, que é estimada por:

$$f_r = \frac{n_f}{n_r} \tag{4.15}$$

- Custo computacional do algoritmo, que é estimado pelo número médio de avaliações de função objetivo requerido para alcançar o ótimo global do problema (Eq. (4.16)):

$$c_c = \frac{1}{n_s} \sum_{i=1}^{n_s} fe_i \tag{4.16}$$

onde:

n_f é o número de vezes em que o algoritmo não atingiu o ótimo;

n_r é o número de execuções do algoritmo;

n_s é o número de vezes em que o algoritmo obteve sucesso $(n_r - n_f)$;

fe_i é o número de avaliações de função objetivo realizadas na execução i.

Após a avaliação de cada um dos algoritmos em ambos os critérios, eles são classificados através de uma análise de Pareto-dominância (Ehrgott, 2000). Os algoritmos pertencentes ao conjunto de Pareto são considerados ótimos.

O uso destes dois critérios é coerente, uma vez que existe uma relação custo-benefício que deve ser levado em conta: algoritmos excessivamente rápidos usualmente levam a ótimos locais, enquanto que, algoritmos com convergência mais lenta tendem a proporcionar uma melhor exploração do espaço. Entretanto, a simples comparação de valores

médios apresenta algumas limitações, que podem levar a conclusões sem o suporte estatístico adequado. Por exemplo, suponha que dois algoritmos, $\mathcal{A}_1$ e $\mathcal{A}_2$, apresentam as mesmas taxas de falha e um número de avaliações de função ligeiramente diferente ($\mathcal{A}_1$ requer um número de avaliações pouco menor que $\mathcal{A}_2$). A aplicação direta da abordagem descrita acima levaria a conclusão que $\mathcal{A}_1$ é melhor que $\mathcal{A}_2$. No entanto, há casos em que isso pode não ser verdade: se $\mathcal{A}_2$ apresenta um desvio padrão menor que $\mathcal{A}_1$, ele pode ser mais confiável para encontrar rapidamente uma boa solução em uma única execução.

O método de avaliação apresentado neste trabalho estende a abordagem descrita por Takahashi et al. (2003), utilizando comparações intervalares ao invés de comparar apenas a média das funções de mérito. A comparação de intervalos é realizada da seguinte forma:

Comparação de Intervalos

Seja X_1 uma variável aleatória, e $x_1^{q1}, x_1^{q2}, \ldots, x_1^{qi}, \ldots, x_1^{qn}$ os pontos que definem os quantis $q_0, q_{\frac{1}{n-1}}, \ldots, q_{\frac{i-1}{n-1}}, \ldots, q_1$ de X_1, como mostrado na Fig. 4.7 (o ponto x^k define o quantil q_α de X se $P(X < x^k) = \alpha$).

Seja X_2 outra variável aleatória, e $x_2^{q1}, x_2^{q2}, \ldots, x_2^{qi}, \ldots, x_2^{qn}$ os pontos de X_2 que definem os mesmos quantis , $q_0, q_{\frac{1}{n-1}}, \ldots, q_{\frac{i-1}{n-1}}, \ldots, q_1$.

Em um contexto de minimização, diz-se que X_1 "é melhor que" X_2 se:

$$\begin{aligned} x_1^{qi} \leq x_2^{qi} \quad &\forall \quad i \in [1, n] \\ \exists\ i \Leftrightarrow x_1^{qi} < x_2^{qi} \quad &, \quad i \in [1, n] \end{aligned} \tag{4.17}$$

Pode-se notar que quando $n \to \infty$, (4.17) é equivalente à Dominância Estocástica de Primeira Ordem (Pemmaraju and Skiena, 2003), que diz que, uma variável aleatória X_1 domina outra variável aleatória X_2 se, e somente se:

$$\text{cdf}(X_1) \geq \text{cdf}(X_2) \quad \forall\ x_i \in X_1 \cup X_2 \tag{4.18}$$

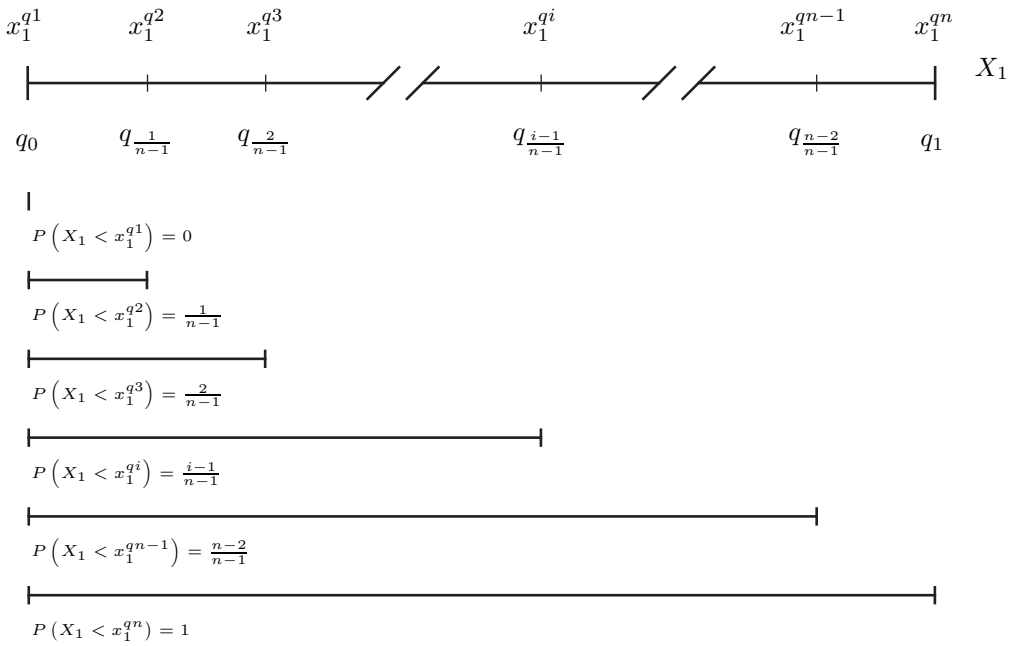


Figura 4.7: Variável aleatória X_1

onde cdf(X) é a função de distribuição cumulativa de X.

Este conceito é ilustrado na Fig. 4.8. Para valores finitos de n, (4.17) pode ser vista como uma aproximação discreta da Dominância Estocástica de Primeira Ordem.

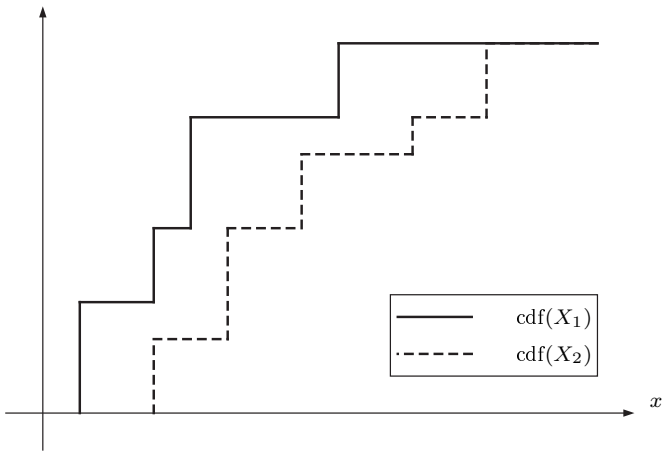


Figura 4.8: Exemplo de dominância estocástica de primeira ordem

O método de comparação de algoritmos proposto aqui é baseado no critério de comparação intervalar descrita acima. Assim como proposto por Takahashi et al. (2003) o número de avaliações de função objetivo é utilizado para estimar o custo computacional do algoritmo. No entanto, o melhor valor de função objetivo é utilizado como alternativa à taxa de falha, para estimar a capacidade de convergência dos algoritmos. Esta abordagem torna possível a comparação de problemas sem um ótimo global conhecido e permite a distinção de algoritmos com taxas de falha semelhantes.

O restante da comparação é idêntico à apresentada por Takahashi et al. (2003): um algoritmo $\mathcal{A}_i$ é considerado ótimo se não existir nenhum outro algoritmo que seja melhor que $\mathcal{A}_i$ em um critério sem ser pior no outro.

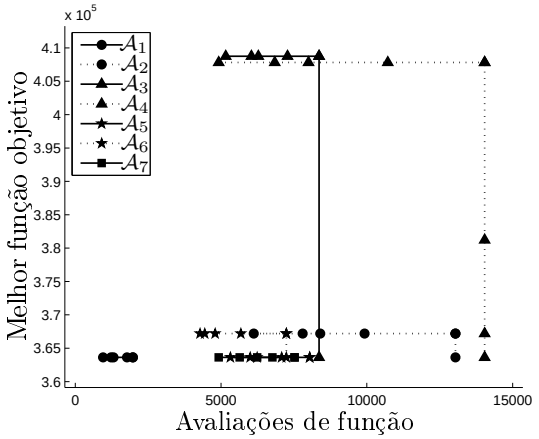
Os algoritmo utilizados na solução do *OCST* e *q-MST*, foram nomeados conforme a Tab. 4.5. Cada algoritmo foi executado 30 vezes para cada uma das instâncias dos dois problemas. Os resultados obtidos para cada uma das funções de mérito foram discretizados em cinco quantis ($q_{0.05}$, $q_{0.25}$, $q_{0.50}$, $q_{0.75}$ e $q_{0.95}$), para comparação intervalar.

Tabela 4.5: Algoritmos

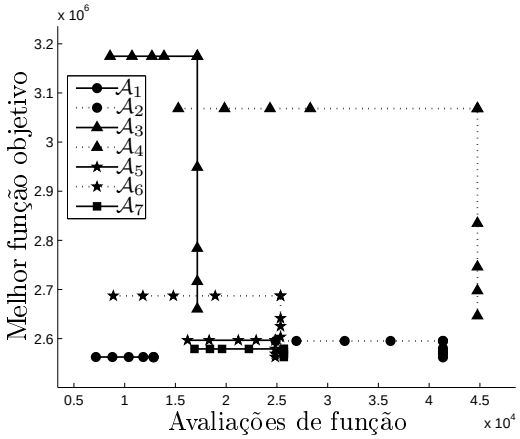
Lab.	Algoritmo
$\mathcal{A}_1$	*GANet*
$\mathcal{A}_2$	GA com Characteristic vector
$\mathcal{A}_3$	GA com Prüfer numbers
$\mathcal{A}_4$	GA com Network random keys
$\mathcal{A}_5$	GA com Edge sets
$\mathcal{A}_6$	GA com Node biased
$\mathcal{A}_7$	*KruskalGA*

4.5.2 *OCST*

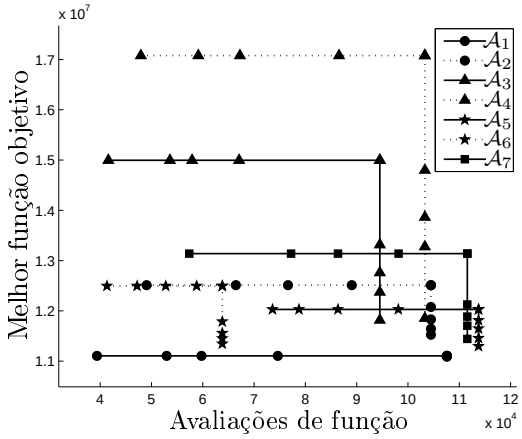
Os resultados obtidos pelos algoritmos para o *OCST* são apresentados na Tab. 4.6. Pela metodologia de comparação intervalar proposta é possível afirmar que $\mathcal{A}_1$ domina todos os outros algoritmos nas três instâncias, uma vez que apresenta melhor capacidade de convergência sem um aumento significativo do esforço computacional. As representações destes intervalos no espaço *avaliações de função* ***vs.*** *melhor função objetivo* são apresentadas nas Figs. 4.9(a) (10 nós), 4.9(b) (25 nós) e 4.9(c) (50 nós).



(a) 10 nós



(b) 25 nós



(c) 50 nós

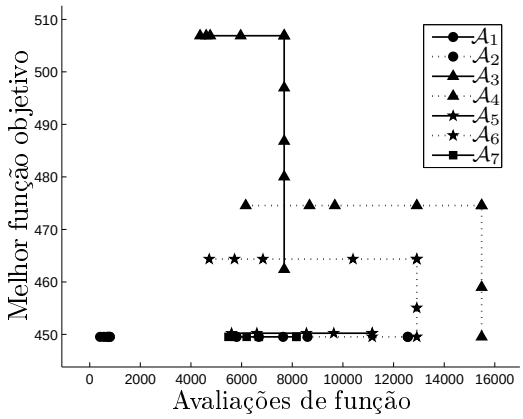
Figura 4.9: *OCST* - Intervalos

Tabela 4.6: *OCST* - Resultados

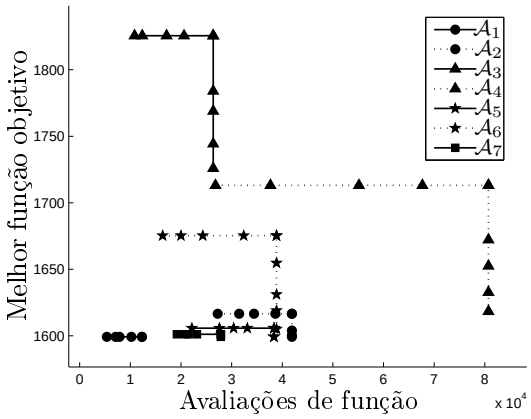
Instância		10 nós		25 nós		50 nós	
alg.	qnt.	val. func.	aval.	val. func.	aval.	val. func.	aval.
$\mathcal{A}_1$	$q_{0.05}$	3,6363e5	950	2,5620e6	7160	1,1089e7	39370
	$q_{0.25}$	3,6363e5	1220	2,5620e6	8800	1,1099e7	52950
	$q_{0.50}$	3,6363e5	1310	2,5620e6	10400	1,1099e7	59740
	$q_{0.75}$	3,6363e5	1780	2,5627e6	11830	1,1102e7	74630
	$q_{0.95}$	3,6363e5	1970	2,5627e6	12850	1,1105e7	107590
$\mathcal{A}_2$	$q_{0.05}$	3,6363e5	6120	2,5620e6	24880	1,1525e7	49040
	$q_{0.25}$	3,6363e5	7800	2,5685e6	26920	1,1643e7	66440
	$q_{0.50}$	3,6363e5	8400	2,5739e6	31680	1,1831e7	76600
	$q_{0.75}$	3,6363e5	9920	2,5804e6	36200	1,2077e7	89040
	$q_{0.95}$	3,6720e5	13040	2,5953e6	41360	1,2511e7	104480
$\mathcal{A}_3$	$q_{0.05}$	3,6363e5	5160	2,6604e6	8560	1,1816e7	41600
	$q_{0.25}$	3,6363e5	6040	2,7165e6	10720	1,2374e7	53600
	$q_{0.50}$	3,6363e5	6280	2,7841e6	12680	1,2761e7	57920
	$q_{0.75}$	3,6363e5	7280	2,9490e6	13880	1,3315e7	67080
	$q_{0.95}$	4,0875e5	8360	3,1746e6	17160	1,4997e7	94520
$\mathcal{A}_4$	$q_{0.05}$	3,6363e5	4920	2,6467e6	15280	1,1854e7	47920
	$q_{0.25}$	3,6363e5	6840	2,6978e6	19840	1,3276e7	59120
	$q_{0.50}$	3,6720e5	8000	2,7461e6	24320	1,3864e7	67240
	$q_{0.75}$	3,8121e5	10720	2,8349e6	28280	1,4800e7	86560
	$q_{0.95}$	4,0782e5	14040	3,0682e6	44760	1,7078e7	103280
$\mathcal{A}_5$	$q_{0.05}$	3,6363e5	5320	2,5627e6	16200	1,1296e7	73600
	$q_{0.25}$	3,6363e5	6000	2,5692e6	18320	1,1458e7	78760
	$q_{0.50}$	3,6363e5	6240	2,5692e6	21200	1,1646e7	86360
	$q_{0.75}$	3,6363e5	7080	2,5790e6	22960	1,1811e7	98120
	$q_{0.95}$	3,6363e5	8040	2,5968e6	24840	1,2028e7	113760
$\mathcal{A}_6$	$q_{0.05}$	3,6363e5	4280	2,5763e6	8880	1,1344e7	41320
	$q_{0.25}$	3,6363e5	4440	2,6035e6	11800	1,1451e7	47200
	$q_{0.50}$	3,6363e5	4800	2,6251e6	14800	1,1556e7	52760
	$q_{0.75}$	3,6363e5	5680	2,6414e6	18920	1,1785e7	58800
	$q_{0.95}$	3,6720e5	7240	2,6871e6	25360	1,2494e7	63800
$\mathcal{A}_7$	$q_{0.05}$	3,6363e5	4920	2,5627e6	16880	1,1444e7	57360
	$q_{0.25}$	3,6363e5	5640	2,5685e6	18400	1,1703e7	77200
	$q_{0.50}$	3,6363e5	6240	2,5692e6	19600	1,1887e7	86360
	$q_{0.75}$	3,6363e5	6760	2,5725e6	22240	1,2127e7	98160
	$q_{0.95}$	3,6363e5	7520	2,5790e6	25680	1,3138e7	111560

4.5.3 *q-MST*

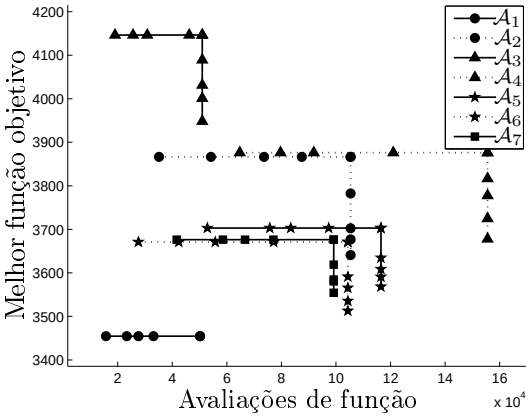
Os resultados obtidos por cada um dos algoritmos para o *q-MST* são apresentados na Tab. 4.7. Assim como *OCST*, o algoritmo $\mathcal{A}_1$ apresenta resultados superiores a todos os outros algoritmos, sendo portanto o único algoritmo Pareto-ótimo. Os intervalos estimados para os algoritmos são apresentados nas Figs. 4.10(a), 4.10(b) e 4.10(c), para 10, 25 e 50 nós respectivamente.



(a) 10 nós



(b) 25 nós



(c) 50 nós

Figura 4.10: *q-MST* - Intervalos

Tabela 4.7: *q-MST* - Resultados

Instância		10 nós		25 nós		50 nós	
alg.	qnt.	val. func.	aval.	val. func.	aval.	val. func.	aval.
$\mathcal{A}_1$	$q_{0.05}$	449,5448	410	1.5992e3	5370	3.4545e3	15700
	$q_{0.25}$	449,5448	560	1.5992e3	7110	3.4545e3	23330
	$q_{0.50}$	449,5448	690	1.5992e3	7880	3.4545e3	27600
	$q_{0.75}$	449,5448	720	1.5992e3	10220	3.4545e3	33120
	$q_{0.95}$	449,5448	790	1.5992e3	12300	3.4545e3	50180
$\mathcal{A}_2$	$q_{0.05}$	449,5448	5800	1.5992e3	27240	3.6410e3	35120
	$q_{0.25}$	449,5448	6680	1.5992e3	31480	3.6767e3	54160
	$q_{0.50}$	449,5448	7640	1.6004e3	34440	3.7024e3	73640
	$q_{0.75}$	449,5448	8600	1.6040e3	38640	3.7824e3	87480
	$q_{0.95}$	449,5448	12560	1.6166e3	41920	3.8664e3	105360
$\mathcal{A}_3$	$q_{0.05}$	462,4037	4360	1.7259e3	10840	3.9482e3	19000
	$q_{0.25}$	480,0168	4600	1.7444e3	12360	4.0011e3	25640
	$q_{0.50}$	486,8105	4760	1.7689e3	17160	4.0313e3	30840
	$q_{0.75}$	496,9731	5960	1.7839e3	20600	4.0891e3	46200
	$q_{0.95}$	506,8800	7680	1.8256e3	26360	4.1462e3	51000
$\mathcal{A}_4$	$q_{0.05}$	449,5448	6160	1.6183e3	26800	3.6778e3	64720
	$q_{0.25}$	449,5448	8680	1.6327e3	37680	3.7248e3	79640
	$q_{0.50}$	449,5448	9680	1.6525e3	55160	3.7774e3	91880
	$q_{0.75}$	458,9808	12920	1.6722e3	67680	3.8166e3	121000
	$q_{0.95}$	474,5539	15480	1.7131e3	80720	3.8762e3	155560
$\mathcal{A}_5$	$q_{0.05}$	449,5448	5600	1.5992e3	22160	3.5681e3	52880
	$q_{0.25}$	449,5448	6600	1.5992e3	27600	3.5908e3	75760
	$q_{0.50}$	449,5448	8560	1.5992e3	30400	3.6087e3	83440
	$q_{0.75}$	449,5448	9640	1.5993e3	33120	3.6344e3	97320
	$q_{0.95}$	450,2335	11160	1.6057e3	38360	3.7030e3	116480
$\mathcal{A}_6$	$q_{0.05}$	449,5448	4720	1.6051e3	16400	3.5126e3	27600
	$q_{0.25}$	449,5448	5720	1.6190e3	20000	3.5353e3	42400
	$q_{0.50}$	449,5448	6840	1.6310e3	24320	3.5657e3	55760
	$q_{0.75}$	455,0606	10400	1.6548e3	32400	3.5911e3	77160
	$q_{0.95}$	464,3534	12920	1.6753e3	38880	3.6709e3	104360
$\mathcal{A}_7$	$q_{0.05}$	449,5448	5480	1.5992e3	19200	3.5542e3	41600
	$q_{0.25}$	449,5448	5760	1.5992e3	20840	3.5804e3	28560
	$q_{0.50}$	449,5448	6200	1.5992e3	21520	3.5846e3	66680
	$q_{0.75}$	449,5448	6680	1.5992e3	23160	3.6190e3	77080
	$q_{0.95}$	449,5448	8160	1.6011e3	27880	3.6763e3	99160

4.6 Testes de Descontinuidade do Espaço de Redes

Os resultados obtidos para o *OCST* e *q-MST* foram utilizados como suporte para estudar os efeitos do uso da *T-norm* no ordenamento das redes no espaço de variáveis. Este ordenamento interfere diretamente na continuidade do espaço, e pode dificultar a otimização. A descontinuidade do espaço de soluções foi comparada para a *T-norm* e a distância Hamming, que é usualmente aplicada em problemas discretos. A seguinte abordagem foi proposta para estimar o ordenamento das soluções e dispersão no espaço:

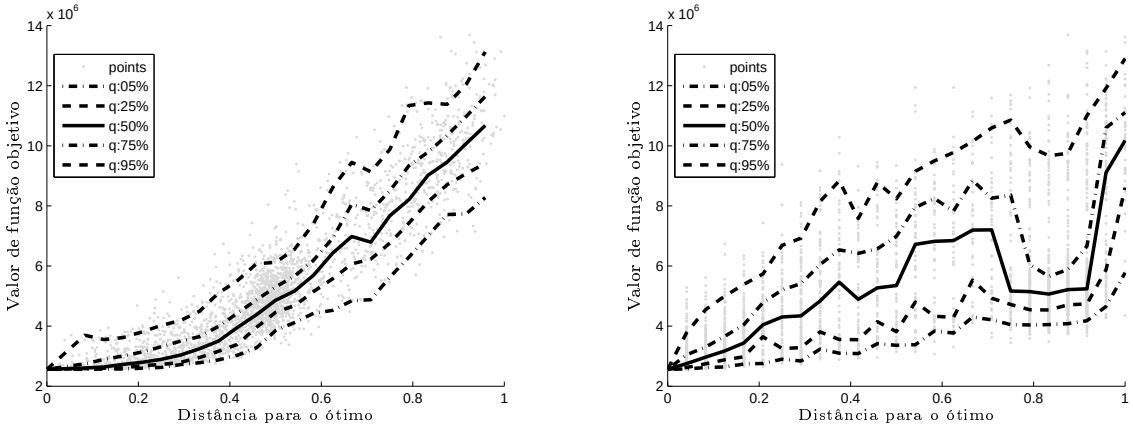
1. Gerar um conjunto de k redes aleatórias seguindo algum padrão, como os apresentados abaixo;
2. Avaliar cada uma das k redes utilizando a função objetivo;
3. Encontrar a distância entre cada rede e o ótimo do problema, $d_s(N_k, N^*)$, utilizando a *T-norm* e a distância Hamming;
4. Ordenar as redes em ordem ascendente de distância para o ótimo;
5. Normalizar a distância para ambas as métricas (entre 0 e 1);
6. Discretizar a distância para o ótimo em n intervalos iguais;
7. Para cada intervalo, encontrar os quantis $q_{0.05}$, $q_{0.25}$, $q_{0.50}$, $q_{0.75}$ e $q_{0.95}$ para estimar a descontinuidade do conjunto de soluções.

O conjunto de dados citado no passo 1 foi gerado seguindo dois padrões distintos:

- Teste 1: Um conjunto de 3.000 redes geradas de forma completamente aleatória, visando estimar a descontinuidade global do espaço;
- Teste 2: Cinco conjuntos de 1.000 redes gerados nas direções definidas entre o ótimo e cinco redes aleatórias, visando estimar a descontinuidade em direções aleatórias no espaço. O conceito de interpolação apresentado na Sec. 4.2.1 foi utilizado como suporte para este teste.

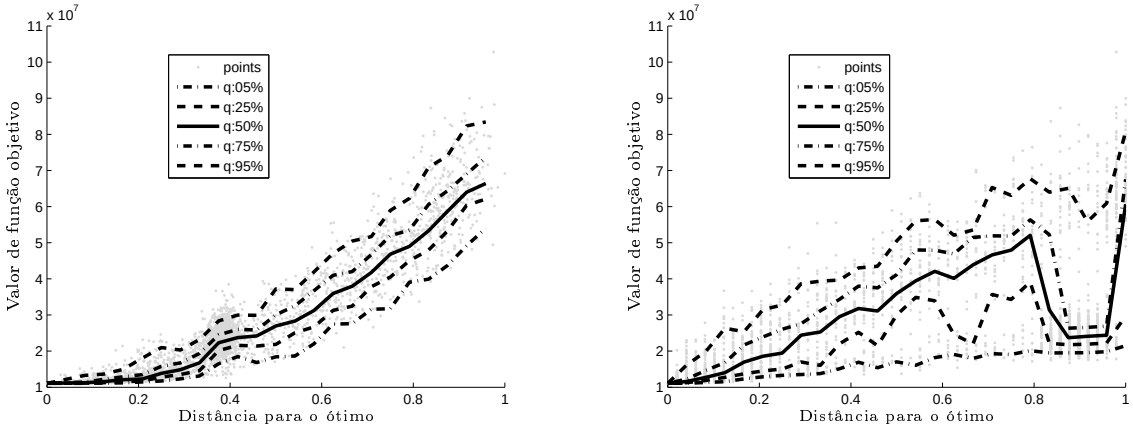
4.6.1 Teste de Descontinuidade 1

A Figs. 4.11 e 4.12 ilustram o ordenamento das soluções proporcionado pela *T-norm* e a distância Hamming, nas instâncias de 25 e 50 nós do *OCST*. Em ambos os casos fica claro que amplitude de variação da função objetivo em cada intervalo é muito maior no conjunto ordenado pela distância Hamming, que no conjunto ordenado pela *T-norm*. Este evento também ocorre de forma semelhante no *q-MST*.



(a) *T-norm* (b) Distância Hamming

Figura 4.11: *OCST* (25 nós) - Teste de descontinuidade 1



(a) *T-norm* (b) Distância Hamming

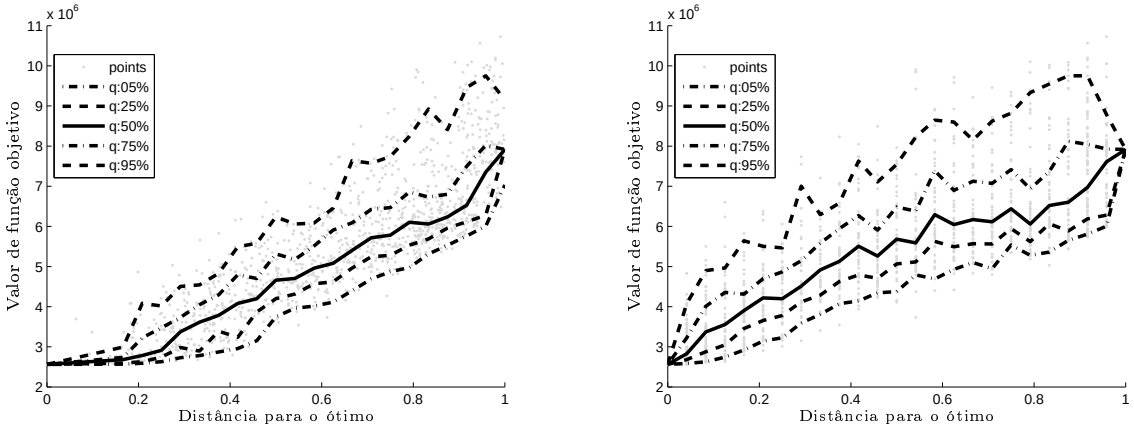
Figura 4.12: *OCST* (50 nós) - Teste de descontinuidade 1

4.6.2 Teste de Descontinuidade 2

Este segundo teste complementa o teste descrito anteriormente: mesmo que o teste 1 apresente boa continuidade, isto não implica em continuidade sobre as possíveis direções do espaço. Como descrito na Sec. 4.6, cinco redes foram geradas aleatoriamente, e, para cada uma dessas redes, foram geradas mais 1.000 redes aleatórias na direção do ótimo. As Figs. 4.13, 4.14, 4.15 e 4.16 ilustram a descontinuidade para duas das direções analisadas, nas instâncias de 25 e 50 nós do *OCST*. Assim como no teste 1, a amplitude da variação do valor de função objetivo para cada intervalo é menor para a *T-norm* que para a distância Hamming. Portanto, pode-se constatar que, para estes casos, a *T-norm* proporciona um ordenamento mais adequado, o que provavelmente explica os resultados obtidos na otimização. Resultados semelhantes também podem ser observados para o *q-MST*.

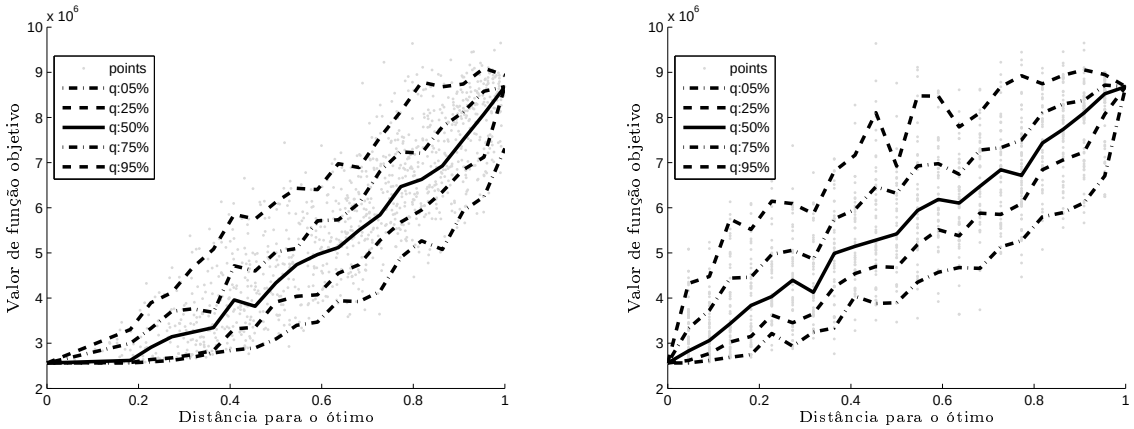
Isso também explica a diferença de performance observada entre o *GANet* e o *KruskalGA*: apesar de ambos os algoritmos realizarem as operações evolutivas no espaço de soluções (ou fenótipos), o KruskalGA utiliza como base a distância de *Hamming* e o *GANet* é construído sobre a *T-norm*, que melhora a relação entre distância para o ótimo *vs.* valor de função objetivo.

O uso dos operadores construídos sobre a *T-norm* elimina um passo crítico no desenvolvimento do algoritmo, que é a escolha da codificação. Como mostrado nas Secs. 4.5.2 e 4.5.3, a escolha da codificação interfere consideravelmente nos resultados obtidos pelo algoritmo evolutivo. Os operadores propostos neste capítulo são desenvolvidos sobre conceitos gerais (independentes da estrutura da rede ou problema tratado), e realizam as operações no espaço de soluções, o que torna os algoritmos derivados dos mesmos robustos quanto à codificação adotada. Esse mesmo aspecto garante generalidade à estes algoritmos.



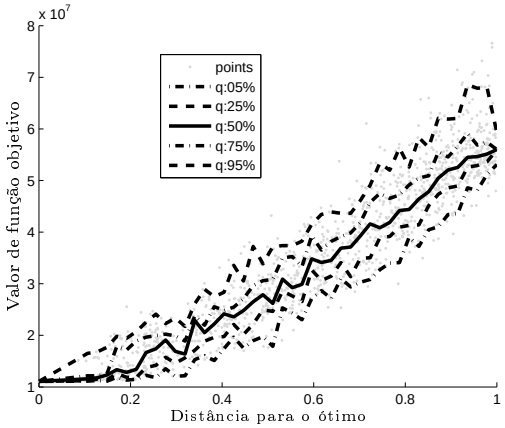
(a) *T-norm* (b) Distância Hamming

Figura 4.13: *OCST* (25 nós) - Teste de descontinuidade 2 - Direção 1

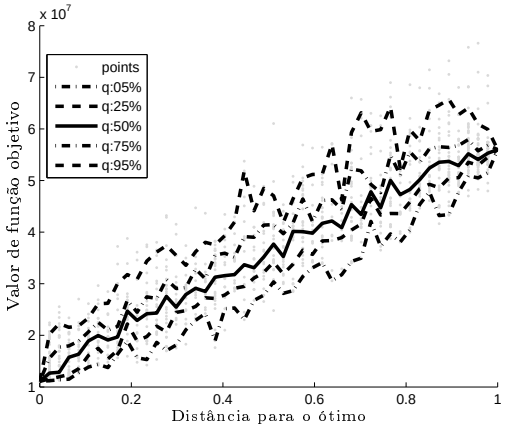


(a) *T-norm* (b) Distância Hamming

Figura 4.14: *OCST* (25 nós) - Teste de descontinuidade 2 - Direção 2

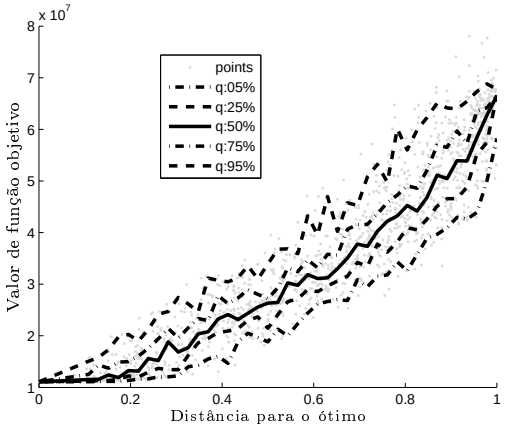


(a) *T-norm*

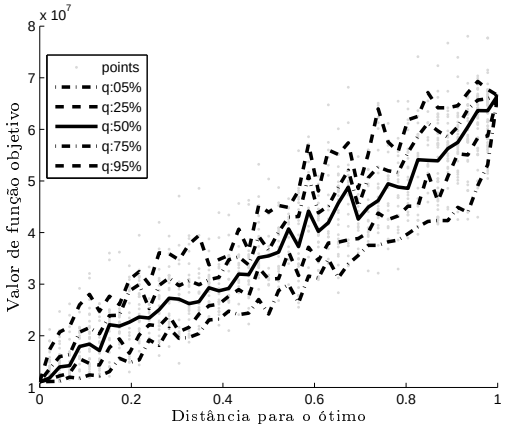


(b) Distância Hamming

Figura 4.15: *OCST* (50 nós) - Teste de descontinuidade 2 - Direção 1



(a) *T-norm*



(b) Distância Hamming

Figura 4.16: *OCST* (50 nós) - Teste de descontinuidade 2 - Direção 2

Capítulo 5

Aplicações no Projeto e Planejamento de Sistemas de Distribuição de Energia Elétrica

5.1 O Sistema de Distribuição de Energia Elétrica

Os sistemas de fornecimento de energia elétrica apresentam uma importância econômica e social muito grande no Brasil, uma vez que atendem 85% da população brasileira, sendo estes responsáveis por mais de 40% de toda a energia consumida no país (Soares, 2001). As redes de distribuição constituem o sub-sistema mais ramificado do sistema de energia elétrica, sendo conseqüentemente responsáveis pela maior parte das perdas ocorridas (Soares et al., 1999; Ćurčić et al., 2001). A redução dessas perdas permite a redução no custo da energia fornecida, dando à concessionária maior flexibilidade na definição de suas políticas de preço. Além disso, estes sistemas estão diretamente ligados ao consumidor, o que implica na necessidade do cumprimento de regulamentações e manutenção da qualidade da energia suprida.

Estes aspectos, associados ao alto custo acarretado pela instalação dessas redes e às limitações dos recursos financeiros disponíveis para as concessionárias, justificam amplamente a utilização de técnicas de otimização no planejamento e projeto destes sistemas.

No entanto, as características do problema não favorecem o uso de algoritmos de

otimização. Por se tratar de um problema de redes, o mesmo apresenta todas as dificuldades destacadas ao longo dos Caps. 1 e 3. Além disso, a natureza do crescimento das cargas no sistema de distribuição, que ocorre em diferentes locais em diferentes instantes de tempo, dificulta ainda mais o problema, introduzindo incertezas no mesmo. Particularmente, se a expansão é executada em pequenos fragmentos do sistema, a maior parte das combinações possíveis de topologias de rede não é considerada, o que pode resultar em um sistema que é mais caro e apresenta mais perdas que um sistema projetado considerando o problema globalmente. Por outro lado a consideração do sistema completo inviabiliza o estudo de todo o conjunto de alternativas possíveis, dado o grande número de possibilidades.

Ao longo deste capítulo são apresentadas aplicações de algoritmos evolucionários em problemas de projeto de redes de distribuição de energia elétrica. São abordadas cinco situações distintas de projeto:

- Projeto de um sistema teste;
- Projeto de um sistema considerando incertezas na evolução de carga;
- Projeto multi-objetivo de dois sistemas de distribuição;
- Posicionamento de subestações associado ao projeto da rede, em duas instâncias mono-objetivo e uma instância multi-objetivo;
- Planejamento dinâmico do *scheduling* da expansão de um sistema.

Os algoritmos empregados nestes casos são construídos utilizando os operadores descritos na Sec. 4.4.1 ou operadores desenvolvidos especificamente para os problema tratados, que são apresentados no Ap. B.

Na seqüencia desta seção é apresentada a formulação do problema de projeto de redes de distribuição de energia elétrica, que será utilizada, de forma direta ou adaptada, em todas as situações de projeto descritas acima.

5.1.1 Formulação do Problema

Três aspectos básicos devem ser levados em conta no projeto de redes de distribuição de energia elétrica (Willis et al., 1996):

- Minimização do investimento de instalações e redimensionamentos no sistema;
- Minimização dos custos de manutenção;
- Minimização das perdas de energia.

Além disso, quatro restrições devem ser consideradas (Willis et al., 1996):

- Atendimento de todos os consumidores;
- Manutenção da estrutura radial da rede (árvore);
- Capacidades de potência das linhas;
- Cumprimento dos níveis de tensão regulamentados.

As duas primeiras restrições equivalem a (1.6) e garantem a estrutura da rede. As outras duas restrições se referem a aspectos técnicos, particulares ao problema.

Os três aspectos relevantes ao projeto citados acima podem ser agregados em uma única função custo, uma vez que ambos representam grandezas monetárias diretas, e diferem apenas no momento em que o dinheiro é investido. As Eqs. (5.1) e (5.2) representam a função objetivo e as restrições para este problema.

$$f^{mc}(N) = \sum_{i=1}^{m} Y_i^N \cdot IC(N_i) + \sum_{t=1}^{at} \left\{ \sum_{i=1}^{m} Y_i^N \left[MC(N_i) + LC_i \right] \right\} \left(1 - int^{rt}\right)^{t-1} \qquad (5.1)$$

$$
\begin{array}{lcl}
c_1 & : & \sum_{i=1}^{m} Y_i^N = |V| - 1 \\
c_2 & : & \sum_{i=1}^{m} Y_i^N \leq |S| - 1 \\
c_3 & : & I_i \leq I_{max}(N_i) \qquad \forall i \in N \\
c_4 & : & 0,92 \leq V_i^n \leq 1,08 \qquad \forall i \in V
\end{array}
\tag{5.2}
$$

onde:

N é a rede avaliada;

m é o número de conexões possíveis;

at é o horizonte de tempo previsto para projeto;

$f^{mc}(N)$ é o custo monetário presente da rede N (em \$);

Y_i^N é 1 se a conexão i está presente na rede N ou 0 caso contrário;

N_i é o tipo de ramo utilizado na conexão i (relacionado com a capacidade);

$IC(N_i) = \ell_i \cdot brc(N_i)$ é o custo total de instalação (ou substituição) do ramo i (em \$);

$MC(N_i) = \ell_i \cdot mnc(N_i)$ é o custo total de manutenção do ramo i (em \$/ano);

$LC_i = 8760 \cdot l^f \cdot en^{tax} \cdot P_i^L$ é o custo total de perdas do ramo i (em \$/ano);

int^{rt} é a taxa de juros anual;

V é o conjunto de nós da rede N;

S é o conjunto de nós induzido pelo conjunto de conexões N;

I_i é a corrente no ramo i;

$I^{max}(N_i)$ é a corrente máxima admissível por um ramo de tipo N_i;

V_i^n é a tensão do nó i;

ℓ_i é o comprimento da conexão i (em km);

$brc(N_i)$ é o custo do ramo de tipo N_i (em \$/km);

$mnc(N_i)$ é o custo de manutenção do ramo de tipo N_i (em \$/km/ano);

l^f é o fator de perda;

en^{tax} é a tarifa de energia (em \$/kWh);

P_i^L é a perda no ramo i (em kW).

A Eq. (5.1) deixa claro a forte relação existente entre as variáveis do problema: alterações nas conexões ou tipos de ramos afetam (de forma não-linear) todo o fluxo de potência da rede, alterando a maior parte das perdas e correntes nos ramos. Este fenômeno é discutido abaixo, em um contexto geral.

Redes com Fluxo Interativo Não-linear

Vários problemas práticos são modelados como redes *multi-branch* em que o fluxo se comporta como uma função não-linear da topologia da rede e dos tipos de conexão da mesma. Instâncias de tais problemas podem aparecer em vários contextos, como: redes de energia elétrica, redes de transporte, redes de comunicação, etc. Nestes problemas, qualquer mudança na topologia ou no tipo de ramo afeta completamente todo o fluxo na rede. Mesmo o fluxo total pode mudar em alguns casos, sem obedecer alguma restrição de conservação de fluxo – este é o caso, por exemplo, do fluxo de energia, que está sujeito a perdas nos ramos. Nestes casos, a função custo se torna uma função não-linear da topologia da rede e do fluxo resultante.

Estes problemas geralmente são $NP-hard$, o que significa que, na maior parte dos casos, não existem algoritmos polinomiais capazes de resolvê-los de forma exata. Com isso, o uso de algoritmos heurísticos se torna a única alternativa para se lidar com grandes instâncias. O projeto de redes de distribuição de energia é um destes problemas, e é utilizado como exemplo de aplicação neste texto.

Similaridades entre o Projeto de Redes de Distribuição, o *OCST* e o *q-MST*

É possível identificar similaridades entre os problemas *OCST* e *q-MST*, apresentados na Sec. 1.3, e o problema de projeto de redes de distribuição de energia elétrica, discutido neste capítulo:

- *OCST*: o projeto de redes de distribuição possui alguma similaridade com o *OCST* considerando demandas de comunicação (definidas pela carga de cada nó)

partindo apenas da raiz (subestação) para os outros nós do problema (consumidores), e conseqüentemente sem demandas entre os nós consumidores e na direção da raiz. Neste caso, o problema *OCST* resultante é uma aproximação do projeto da rede de distribuição, com fluxo constante.

- *q-MST*: a interação quadrática entre as conexões (a alteração de um ramo implica na modificação dos custos decorrentes do mesmo em todos os outros ramos) apresenta uma estrutura similar ao comportamento do fluxo de potência em projeto de redes de distribuição de energia.

Os resultados obtidos pelos algoritmos apresentados na Sec. 4.4 na solução do *OCST* e *q-MST* sugerem a adeqüabilidade destes algoritmos para o projeto de redes de distribuição de energia elétrica, dadas as similaridades entre os problemas.

5.1.2 Formulação Multi-objetivo

Em determinadas situações pode ser necessário considerar a confiabilidade do sistema durante a etapa de projeto. Nestes casos, interrupções no fornecimento de energia podem ser causadoras de conseqüências graves, como prejuízo financeiro da concessionária e clientes, ou acidentes sérios (incluindo mortes), como por exemplo, na interrupção de fornecimento de energia a hospitais ou sistemas de trânsito.

Isso faz com que dois novos aspectos devam ser considerados no planejamento:

- Minimização do número de interrupções;
- Minimização do tempo das interrupções.

Assim como no custo monetário, pode-se agregar ambos os aspectos em uma única função objetivo, como mostrado em (5.3). Essa função estima a confiabilidade do sistema através do custo causado por falhas no sistema.

$$f^{fc}(N) = \sum_{t=1}^{at} \left\{ \sum_{i=1}^{m} Y_i^N \cdot \lambda(N_i) \cdot \ell_i \left[r(N_i) \cdot P_i^A \cdot en^{tax} + fl^{tax} \right] \right\} \left(1 - int^{rt}\right)^{t-1} \quad (5.3)$$

onde:

$f^{fc}(N)$ é o custo de falta da rede N (em \$);

$\lambda(N_i)$ é a taxa de falha do ramo de tipo N_i (em $falhas/km/ano$);

$r(N_i)$ é a duração média por falta do ramo de tipo N_i (em $h/falha$);

P_i^A é a potência ativa no ramo i (em kW);

fl^{tax} é o custo médio por falha.

Apesar das funções objetivo (5.1) e (5.3) representarem custos financeiros, as mesmas não podem ser agregadas em uma única função objetivo, umas vez que representam interesses distintos: é razoável assumir que a concessionária prefira investir uma determinada quantia na instalação e manutenção de sistemas que perder a mesma quantia devido a falhas e interrupções. Isso pode ser justificado pelos prejuízos não mensuráveis decorrentes das falhas, como possíveis autuações, processos jurídicos e desgastes na imagem da empresa.

Trabalhos como (Miranda et al., 1994; Tang, 1996; Ching-Tzong and Guor-Rurng, 2002) incluem a confiabilidade no problema de otimização, porém utilizando uma soma ponderada dos custos monetários e dos custos relacionados à falhas no sistema. Como é discutido na Sec. 2.3, essa abordagem não é recomendada, uma vez que torna impossível o mapeamento de parte do conjunto de Pareto em problemas não-convexos, assim como os tratados ao longo deste capítulo. Os casos multi-objetivo apresentados neste texto são tratados através de algoritmos que possibilitam o mapeamento de todas as soluções do conjunto de Pareto.

5.2 Sistema Teste

[1]O sistema de distribuição de 23 nós proposto por Gómez et al. (2004) foi utilizado como problema teste para análise do *GANet* e *ClonalNet* neste tipo de problema. Dois tipos de ramos estão disponíveis para o projeto (os dados desses condutores podem ser encontrados na referência (Gómez et al., 2004)). O conjunto de conexões possíveis é o mesmo proposto na referência, com 35 variáveis (Fig. 5.1).

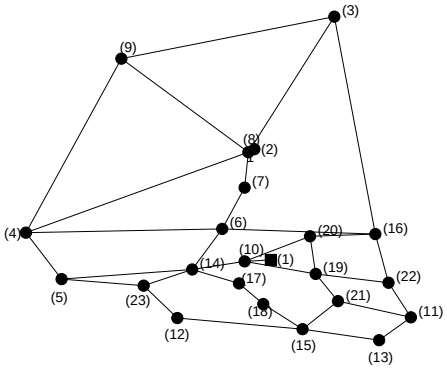


Figura 5.1: Sistema teste - Conexões possíveis

Cada algoritmo foi executado 30 vezes, e os resultados obtidos são apresentados na Tab. 5.1. Ambos algoritmos encontraram a mesma solução em todas execuções (Fig. 5.2). Esta solução possui um custo de $\$171.698,03$ e é superior à melhor solução encontrada em (Gómez et al., 2004).

Tabela 5.1: Sistema teste - Resultados obtidos pelo *GANet* e *ClonalNet*

Algoritmo	Convergência	Avaliações de função
GANet	100%	2.690
ClonalNet	100%	16.909

A diferença entre o número de avaliações de função requeridas pelo *GANet* e pelo *ClonalNet* pode ser justificada pela estrutura de ambos os algoritmos, consideravelmente distinta. A ausência de um processo de seleção estocástico no Clonal reduz a pressão

[1]Os resultados apresentados nesta seção foram retirados das referências (Carrano et al., 2007f,c).

imposta na direção do ótimo. Isso reduz a intensidade da busca em torno da solução e reduz a velocidade de convergência. Por outro lado, essa característica torna o Clonal menos suscetível à convergência prematura, além de permitir o mapeamento de soluções sub-ótimas juntamente com o ótimo do problema. As vantagens do mapeamento de soluções sub-ótimas são discutidas na próxima seção.

Como discutido no Cap. 3, o GA pode ser adaptado para apresentar um comportamento semelhante mas tem, no entanto, uma estrutura menos adequada para este tipo de mapeamento.

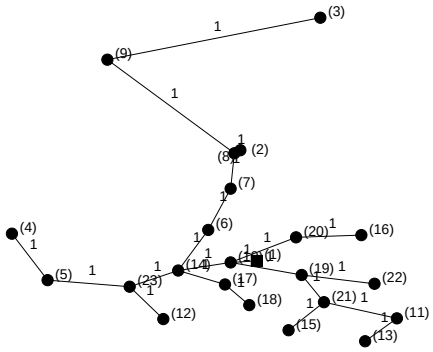


Figura 5.2: Sistema teste - Melhor solução alcançada

5.3 Expansão de Redes de Distribuição Considerando Incertezas na Evolução de Carga

[2]Os sistemas de distribuição de energia elétrica estão sujeitos a evoluções constantes das cargas instaladas, tanto no local da carga quanto no crescimento, o que, em grande parte dos casos, implica na necessidade de expansões ou re-projetos periódicos (Carrano et al., 2005; Carvalho et al., 1998; Miranda et al., 1994).

Um projeto adequado destes sistemas depende não só da consideração da carga atual, mas também da carga que deverá existir dentro de um horizonte de tempo, o que insere o problema em um contexto de incertezas. Redes projetadas subestimando o crescimento da carga tendem a se tornar insuficientes em pouco tempo, requerendo novas expansões que conduzem a um custo total superior a instalação inicial do sistema completo. Por outro lado, o uso de cargas super-estimadas na etapa de projeto geralmente leva a soluções que não fazem uma alocação adequada dos recursos disponíveis, requerendo investimentos mais altos que os necessários e deixando equipamentos trabalhando com grande parte de sua capacidade ociosa. Isso faz com que as companhias de energia utilizem técnicas de previsão de carga, visando reduzir a chance de ocorrência destas situações desfavoráveis (Miranda et al., 1994; Carvalho et al., 1998). A maior parte das abordagens voltadas a este problema realiza o projeto do sistema tendo em conta um único cenário futuro, tido como o mais provável, ao invés de considerar um conjunto de cenários possíveis, capazes de representar a incerteza relacionada à previsão de carga (Ramírez-Rosado and Bernal-Agustín, 1998; Gómez et al., 2004; Parada et al., 2004).

De fato, a variação da carga em cada nó do sistema é um processo estocástico que se acumula ao longo do tempo, levando a um conjunto de variáveis estocásticas com uma dada distribuição de probabilidade para cada um dos nós, sendo que a variância dessas distribuições cresce juntamente com o horizonte de tempo (Willis and Northcode-Green, 1983; Morsi et al., 1994). Conseqüentemente, um conjunto discreto representativo de

[2]Os resultados apresentados nesta seção foram retirados da referência (Carrano et al., 2007c).

possíveis cenários de carga para cada nó levaria a um conjunto exponencial de cenários possíveis para o sistema como um todo. Além disso, fatores podem afetar o custo da energia na rede, que tende a ser diferente para cada nó do sistema: a localização final dos consumidores por exemplo, pode evoluir seguindo padrões diferentes dos previstos, levando a diferentes custos e diferentes perdas no circuito secundário.

O crescimento da cardinalidade do conjunto de cenários possíveis torna impraticável qualquer tentativa de se projetar a rede tendo em conta todos os cenários de possíveis, mesmo em instâncias pequenas.

A abordagem proposta nesta seção utiliza o *ClonalNet* para a solução deste problema. Este algoritmo oferece como resultado o ótimo do problema juntamente com outras soluções sub-ótimas, que formam o conjunto de soluções candidatas. Assume-se que a distribuição de probabilidade da evolução da carga de cada nó é conhecida. A busca pela rede ótima é executada considerando o crescimento médio de cada nó e uma única taxa para a energia. O conjunto de soluções candidatas é então submetido a uma análise de sensibilidade, que considera vários cenários possíveis (do crescimento de carga e da taxa de energia dos nós), seguindo as distribuições de probabilidade inicialmente propostas. A solução final é escolhida tendo em conta os resultados obtidos na análise de sensibilidade. O estudo de caso apresentado revela um padrão interessante: a solução ótima para o cenário médio tende a ser mais sensível à variações nas condições do sistema que outras soluções, não sendo portanto a solução escolhida. Isso portanto justifica o custo computacional requerido pelo *ClonalNet* para o mapeamento de sub-ótimos, uma vez que estes representam alternativas capazes de lidar com condições não consideradas explicitamente durante o processo de otimização.

5.3.1 Análise de Sensibilidade Multi-objetivo

A análise de sensibilidade proposta é executada para mudanças nas condições de operação da rede de distribuição, considerando cada solução sub-ótima como uma solução candidata. Algumas razões justificam essa metodologia:

- O cenário médio, apesar de provavelmente incorreto quando comparado ao cenário real, é o mais provável, sendo portanto a escolha mais razoável para otimização;
- O mapeamento de soluções sub-ótimas define um conjunto de alternativas de projeto. É possível que algumas redes que apresentam boa performance para o cenário médio se revelem pouco robustas, apresentando uma perda considerável de desempenho para variações do cenário médio. Outras redes que apresentam performance similar para o cenário médio podem ainda manter um desempenho razoável para perturbações deste cenário.

A análise de sensibilidade proposta apresenta a seguinte estrutura:

1. Gerar um conjunto de cenários (simulação de Monte Carlo), considerando perturbações no crescimento de carga e tarifa de energia de cada nó, dentro do horizonte de tempo de projeto. Estas variáveis devem ser geradas seguindo uma distribuição de probabilidade conjunta, que é assumida como conhecida;
2. Avaliar todas as soluções candidatas em todos os cenários, de acordo com 4 funções de mérito:
   - Custo da solução para o cenário médio;
   - Taxa de infactibilidade da solução;
   - Custo médio da solução para os cenários factíveis;
   - Custo médio de falta da solução para os cenários factíveis.
3. Extrair o conjunto de soluções eficientes, baseado em uma análise de dominância multi-objetivo das soluções para os quatro critérios citados acima;
4. Descartar as soluções eficientes que apresentam altas taxas de infactibilidade;
5. Selecionar uma solução dentre as restantes, com base nas funções de mérito e na opinião de um especialista.

Os aspectos mais importantes deste procedimento são descritos na seqüência.

Modelagem de Incertezas

Dois conjuntos de parâmetros são modelados como variáveis incertas:

- Crescimento de carga para cada um dos nós;
- Tarifa de energia para cada um dos nós.

A principal fonte de incertezas que afeta o projeto da rede é o crescimento da carga para cada nó. A modelagem da tarifa de energia de cada nó como incerta permite levar em conta um fator de incerteza importante: o custo por kWh fornecido varia entre os nós, devido a diferentes custos ocorridos no sistema secundário que precedem os mesmos. Isto pode ser aplicado para levar em conta incertezas na distribuição geográfica das cargas futuras, que serão adicionadas a cada nó. Essa variação causa diferentes custos de instalação de cabos e afeta as perdas previstas para o sistema secundário.

A rede que deve ser projetada possui $2n + 1$ variáveis incertas (carga, tarifa de energia para cada nó e taxa de juros). Seja $V_i \in \mathbb{R}^{2n+1}$ o vetor de variáveis incertas no ano i. O vetor V_{i+1} é calculado por:

$$V_{i+1} = V_i + \Delta_i \tag{5.4}$$

onde Δ_i é um vetor de variáveis aleatórias que representa o incremento de cada variável do problema, de um ano para o próximo. Para representar as dependências entre as variáveis, o vetor Δ_i é calculado por:

$$\Delta_i = D \cdot \delta_i \tag{5.5}$$

onde δ_i é um vetor de variáveis aleatórias independentes e D é uma matriz que descreve a dependência entre as variáveis. Na simulação de Monte Carlo, o vetor δ_i é gerado seguindo uma distribuição de probabilidade conhecida. δ_i é então utilizado para encontrar Δ_i e conseqüentemente V_{i+1}.

No estudo de caso apresentado nesta seção, as variáveis δ_i foram modeladas como processos Gaussianos com distribuição conhecida (qualquer modelo estocástico pode ser utilizado, sem necessidade de adaptações no processo proposto). O efeito da incerteza acumulada em uma variável δ_i, após cinco anos, é ilustrado na Fig. 5.3.

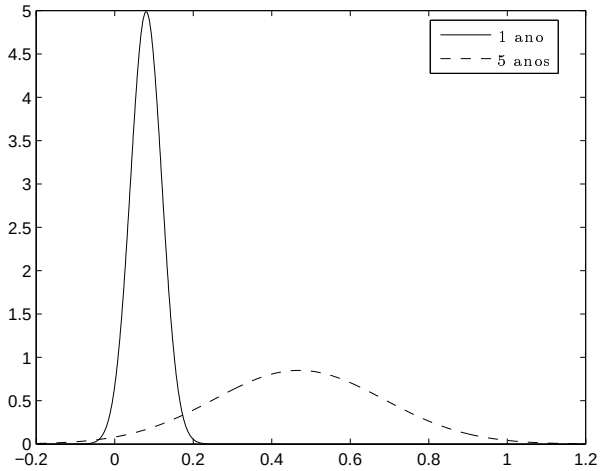


Figura 5.3: Incerteza acumulada em uma variável Gaussiana

Funções de Mérito

Os resultados da simulação de Monte Carlo são utilizados para avaliar a performance de cada solução, considerando quatro critérios:

- Custo da rede para o cenário médio (f^{m1}) - este critério analisa o custo da solução para as condições nominais de projeto (cenário mais provável);

- Taxa de infactibilidade da rede (f^{m2}) - este critério analisa o número de vezes em que a rede se torna infactível nos cenários analisados, que é correlacionado com a probabilidade de infactibilidade da rede dentro do horizonte de tempo considerado;

- Custo médio da rede para os cenários factíveis (f^{m3}) - este critério analisa o custo médio da rede nos cenários em que a mesma é factível. Esta função é, em geral,

diferente de f^{m1}, e representa outra medida para o custo incorporando o efeito das incertezas;

- Custo médio de falta da rede (f^{m4}) - este critério analisa os custos médios causados por eventuais falhas da rede.

Estes critérios apresentam a seguinte formulação matemática:

$$\begin{aligned} f^{m1} &= CN_{nm}^{N} \\ f^{m2} &= \frac{1}{N_s}\sum_{i=1}^{N_s} FE_i^N \\ f^{m3} &= \frac{1}{N_{fs}^N}\sum_{i=1}^{N_{fs}^N} CN_i^N \\ f^{m4} &= \frac{1}{N_{fs}^N}\sum_{i=1}^{N_{fs}^N} FN_i^N \end{aligned} \tag{5.6}$$

onde:

CN_{nm}^N é o custo da rede N para as condições nominais, calculado utilizando (5.1);

N_s é o número de cenários analisados;

FE_i^N é 1 se a rede N é infactível para o cenário i ou 0 caso contrário;

N_{fs}^N é o número de cenários em que a rede N é factível;

CN_i^N é o custo da rede N para o cenário i;

FN_i^N é o custo de falta da rede N para o cenário i, calculado utilizando (5.3).

Algoritmo Utilizado

A única adaptação realizada no algoritmo *ClonalNet* para solução do problema proposto foi a adição de um população externa ao mesmo. Esta população permite um aumento do mapeamento de soluções. Todas soluções avaliadas pelo algoritmo são inseridas nessa população, que periodicamente é analisada para eliminação de redundância. Uma rede é considerada redundante se existe outra rede, próxima a ela, com melhor valor de função objetivo, ou:

Definição 5.1 *Uma rede B é considerada redundante se existe alguma rede A na população externa tal que:*

$$\begin{aligned} d_s(A,B) &< \varepsilon \\ f^{mc}(A) &< f^{mc}(B) \end{aligned}$$

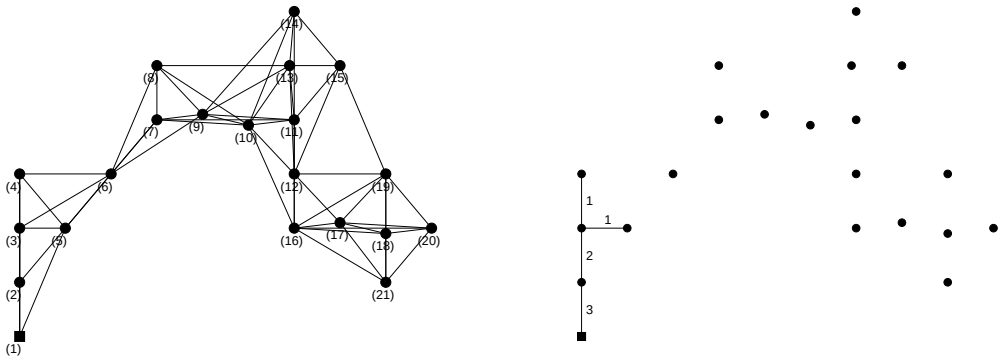
onde ε é um escalar que define o raio de redundância da população.

5.3.2 Resultados

O algoritmo proposto foi aplicado na otimização de um sistema de 21 nós, considerando incertezas na evolução da carga e tarifa de energia.

O tempo de projeto considerado é de 10 anos, estando 7 tipos de condutores disponíveis para projeto. Os principais dados relacionados a este caso podem ser encontrados na referência (Carrano et al., 2007d).

A *Controlled-Greedy Encoding* foi aplicada para redução das variáveis do problema, resultando em 62 conexões possíveis (Fig. 5.4(a)). Neste caso, existe um sistema inicial, que deve ser considerado durante o processo de otimização – ele é apresentado na Fig. 5.4(b).

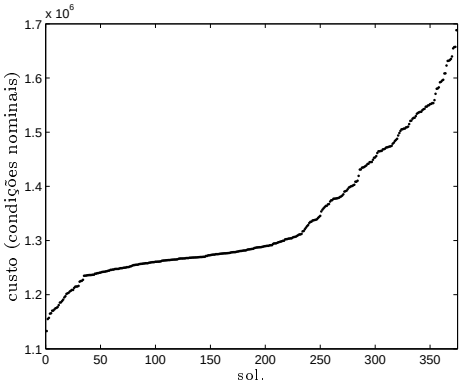


(a) Conexões possíveis (b) Sistema inicial

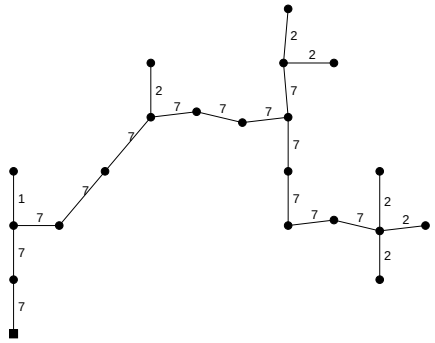
Figura 5.4: Sistema de 21 nós - Problema

O algoritmo mapeou 374 soluções, como apresentado na Fig. 5.5(a). A Fig. 5.5(b) mostra a melhor solução obtida para o cenário médio (esta solução será referida como

$1a$, e tem um custo de $\$1.132.799,55$).



(a) Soluções mapeadas pelo *ClonalNet*



(b) Melhor solução obtida para as condições nominais

Figura 5.5: Sistema de 21 nós - Resultados

Análise de Sensibilidade Multi-objetivo

As 374 soluções mapeadas pelo *ClonalNet* foram submetidos a uma Simulação de Monte Carlo, com a avaliação de 2.000 cenários possíveis. Os resultados obtidos pela Simulação de Monte Carlo foram utilizados para avaliar cada solução nos critérios apresentados em (5.6). Um conjunto foi construído com as soluções não-dominadas. Os parâmetros utilizados nas distribuições de probabilidade são apresentados na Tab. 5.2. O crescimento médio das cargas e das tarifas de energia foi considerado igual para todos os nós, o que adotado para efeito de simplificação.

O conjunto de Pareto encontrado é composto por 41 das 374 soluções mapeadas. A Tab. 5.3 mostra a performance das soluções obtidas para cada uma das funções de mérito.

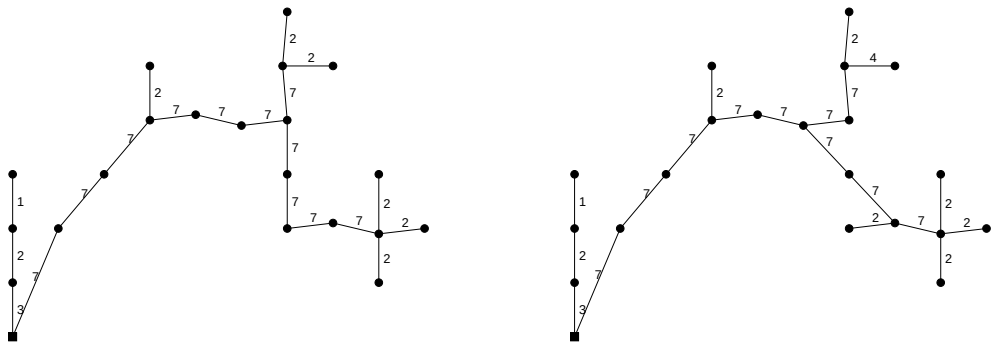
Tabela 5.2: Parâmetros das distribuições de probabilidades

Taxa de cresc. de carga:	**01 ano**	**10 anos**
média:	0,050	0,629
desvio padrão:	0,025	0,280
Var. da tarifa de energia:	**01 ano**	**10 anos**
média:	0,000	0,000
desvio padrão:	0,050	0,629

Análise dos Resultados

Como discutido na Sec. 5.3.1, soluções com alta taxa de infactibilidade não são robustas o suficiente para atender com à maior parte dos cenários possíveis, o que justifica o seu descarte. É importante notar que a rede com mínimo custo para o cenário médio ($1a$) tem uma taxa de infactibilidade maior que 70%, e deve ser descartada.

Soluções com taxa de infactibilidade menor que 20% foram consideradas aceitáveis. Isso reduz o conjunto de soluções candidatas para apenas quatro: $3a$, $17a$, $67a$ e $301a$. Uma vez que esse número de redes é pequeno, elas podem ser facilmente analisadas por um projetista experiente. O autor acredita que as soluções $3a$ e $17a$ (Figs. 5.6(a) e 5.6(b) respectivamente) representam boas alternativas para este sistema. Estas soluções têm um custo nominal pouco superior a $1a$ (cerca de 2% e 5% respectivamente) e superam $1a$ em todos os outros critérios.



(a) Solução $3a$ (b) Solução $17a$

Figura 5.6: Sistema de 21 nós - Alternativas viáveis

Tabela 5.3: *ClonalNet* - Soluções não-dominadas

Sol.	f^{m1}	f^{m2}	f^{m3}	f^{m4}
$1a$	1132799,55	0,7286	1244216,18	1831968,35
$2a$	1155002,59	0,6992	1211798,82	1726380,49
$3a$	1157128,93	0,0035	1226846,20	1736293,99
$4a$	1164948,03	0,6967	1221791,14	1718656,74
$5a$	1165814,14	0,6952	1222324,53	1691850,80
$6a$	1170425,18	0,6856	1224558,65	1641068,99
$10a$	1175613,79	0,6856	1229825,58	1587254,26
$17a$	1194965,91	0,0035	1261480,42	1689429,56
$19a$	1201459,76	0,7291	1263077,08	1551298,81
$25a$	1208758,56	0,7423	1273264,87	1549835,29
$34a$	1227517,01	0,7403	1291989,34	1550149,19
$50a$	1240780,79	0,7787	1310098,86	1298258,54
$51a$	1241676,00	0,7787	1310947,16	1298215,04
$65a$	1247486,58	0,7681	1314482,80	1209464,15
$66a$	1247595,64	0,8324	1300151,79	1379186,02
$67a$	1248171,33	0,0025	1311512,42	1683388,93
$69a$	1248872,96	0,7661	1315515,42	1176593,07
$73a$	1250052,95	0,7651	1316742,28	1176784,06
$104a$	1261099,73	0,2699	1338725,00	1585951,92
$117a$	1264890,36	0,7023	1321124,30	1558351,26
$132a$	1268127,01	0,6992	1323606,86	1500580,70
$134a$	1268529,06	0,7651	1335090,79	1176736,84
$139a$	1269184,29	0,5575	1337490,17	1567388,26
$141a$	1269425,84	0,6992	1325327,51	1448208,43
$151a$	1273493,81	0,7078	1331363,83	1438844,68
$206a$	1291271,35	0,8770	1354175,85	1133528,79
$210a$	1294811,74	0,8704	1356947,20	1046831,37
$232a$	1311185,53	0,7590	1366745,50	1316450,89
$233a$	1312146,27	0,7590	1367752,68	1296170,80
$237a$	1323571,48	0,7635	1381889,55	1183375,30
$238a$	1326310,50	0,7332	1385793,22	1406853,41
$241a$	1334298,56	0,7119	1390992,27	1359490,32
$250a$	1345556,52	0,2587	1417873,43	1150903,27
$252a$	1356879,30	0,2734	1431309,07	1139807,09
$253a$	1359283,93	0,2724	1432832,30	1139571,50
$262a$	1377050,62	0,2527	1454658,83	1674244,40
$265a$	1377741,42	0,3605	1455127,69	1115444,35
$266a$	1378220,24	0,2527	1455891,12	1652436,58
$293a$	1441345,88	0,2162	1511650,22	1454593,24
$301a$	1455851,51	0,1808	1520748,97	1511240,00
$309a$	1469515,97	0,8223	1531425,87	818023,63

Comparação com o *GANet*

O *GANet*, discutido na Sec. 4.4.2, foi aplicado ao mesmo problema, para estabelecer uma comparação com o *ClonalNet*. A população final do GA foi submetida ao mecanismo de supressão, reduzindo o conjunto de soluções candidatas a 13. Este conjunto foi submetido à mesma análise de sensibilidade multi-objetivo discutida anteriormente. O conjunto de Pareto encontrado é composto por 7 soluções, como mostrado na Tab. 5.4[3].

É importante notar que a melhor solução obtida pelo GA para o cenário médio coincide com a melhor solução obtida pelo *ClonalNet* para estas condições. No conjunto de soluções mapeadas pelo GA, apenas $12b$ apresenta taxa de infactibilidade inferior à 20%. Esta solução apresenta uma performance inferior a de $3a$ e $17a$, para as funções de mérito consideradas. Portanto, pode-se constatar que o *ClonalNet* é mais adequado a este tipo de problema, e seu maior custo computacional é ponderado pela capacidade do mesmo em mapear outras alternativas de projeto.

Tabela 5.4: *GANet* - Soluções não-dominadas

Sol.	f^{m1}	f^{m2}	f^{m3}	f^{m4}
$1b$	1132799,55	0,7280	1244439,22	18328346,32
$2b$	1155002,59	0,6985	1211967,69	17269902,72
$3b$	1170425,18	0,6850	1224697,18	16416035,96
$6b$	1177197,19	0,7015	1234459,51	16098496,83
$10b$	1317687,58	0,7635	1373407,11	12962783,36
$11b$	1337903,39	0,2965	1412298,59	19579966,26
$12b$	1377400,40	0,0040	1453732,57	18880288,24

[3]O índice b indica as soluções obtidas pelo *GANet*.

5.4 Projeto Multi-objetivo de Redes de Distribuição

[4]Como discutido na Sec. 5.1, em várias situações se torna importante considerar a confiabilidade como um objetivo do problema de projeto de sistemas de distribuição de energia elétrica. Neste caso, o problema se torna multi-objetivo, sendo portanto necessário o uso de ferramentas adequadas para solução do mesmo.

Nesta seção é apresentado um GA capaz de resolver o problema de projeto multi-objetivo de redes de distribuição de energia elétrica e os resultados obtidos mediante a aplicação dele a dois problemas, com 21 nós e 100 nós.

5.4.1 Algoritmo Genético Específico

O GA específico desenvolvido é baseado no *NSGA-II*, proposto por Deb et al. (2002) e discutido na Sec. 3.2.1. Os operadores de mutação e cruzamento foram adaptados para tornar o algoritmo apto para este problema. Este algoritmo é referido ao longo deste trabalho como *NSGA-PS*.

Operadores de Cruzamento e Mutação

A inclusão da confiabilidade como objetivo de projeto pode levar a inclusão de alguns ramos redundantes na rede. Estes ramos, chamados de ramos de reserva, ficam inicialmente inativos, e são ativados em caso de falhas. Isso geralmente possibilita o isolamento da falha, reduzindo, ou até eliminando seu impacto. Os operadores utilizados devem ser capazes de levar em conta essa redundância, mas ainda assim mantendo a factibilidade (e radialidade) da rede principal.

Os operadores construídos para o algoritmo proposto exploram as características do problema de redes de distribuição de energia. Os mesmos garantem, por construção, a manutenção da estrutura da rede, além de considerar os ramos de reserva. Foram desenvolvidos dois operadores de cruzamento, sete operadores de mutação, e dois ope-

[4]Os resultados apresentados nesta seção foram retirados da referência (Carrano et al., 2006b).

radores determinísticos, responsáveis por tentar melhorar os dois pontos extremos do conjunto de Pareto. A descrição completa da estrutura destes operadores é apresentada na Sec. B.3 do Ap. B. A escolha de qual operador é executado (tanto para cruzamento quanto para mutação) é realizada através de uma variável aleatória uniforme.

5.4.2 Resultados

O GA específico foi aplicado na solução de duas instâncias do problema: 21 nós e 100 nós. Para ambos os casos, o tempo de projeto considerado foi de 1 ano, sendo disponibilizado para projeto 9 tipos de condutores, 7 condutores descobertos e 2 condutores cobertos, que possuem maior confiabilidade. Os dados detalhados dessas simulações podem ser encontrados em (Carrano et al., 2006a).

Sistema de 21 Nós

As Figs. 5.4(b) e 5.4(a) apresentam o sistema inicial e o conjunto de conexões possíveis para o sistema de 21 nós tratado. Este conjunto de conexões possíveis foi gerado com a *Controlled-Greedy Encoding*, e é composto por 62 variáveis. Foi imposto um limite máximo de 7 ramos de reserva para o problema, visando aumentar a confiabilidade das redes sem aumentos excessivos dos custos.

Como resultado da aplicação do GA, foram mapeadas 137 soluções Pareto-ótimas (Fig. 5.7). A rede A (Fig. 5.8(a)) é a rede mais confiável encontrada[5]. Como esperado, a mesma utiliza todos os sete ramos de reserva. As redes B e C (Figs. 5.8(b) e 5.8(c)) representam uma solução intermediária e a rede de mínimo custo respectivamente. Deve-se notar que B tem um custo monetário 26.4% maior que C e 50.4% menor que A. O custo de falha de B é 523.4% maior que o custo de falha da rede A e 94.3% menor que o custo de falha da rede C. As redes A e B tiveram todos os condutores pré-existentes substituídos por condutores com menor índice de falhas. Já a rede C teve apenas dois condutores substituídos por condutores de maior capacidade, para reduzir

[5]Os ramos de reserva são representados por linhas tracejadas.

os custos relacionados às perdas e cumprir com as novas demandas de carga.

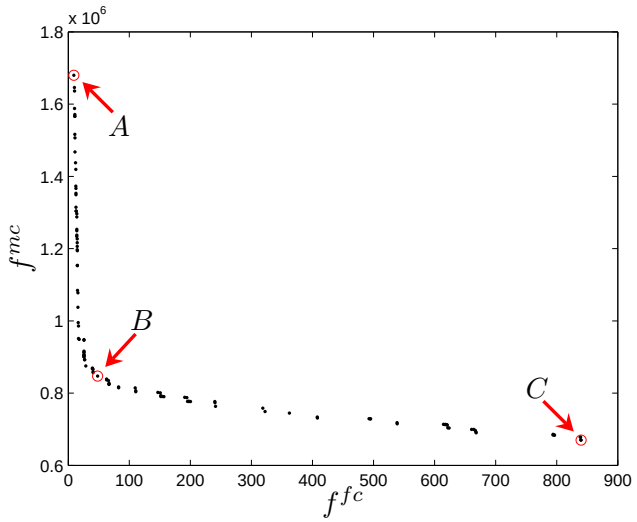
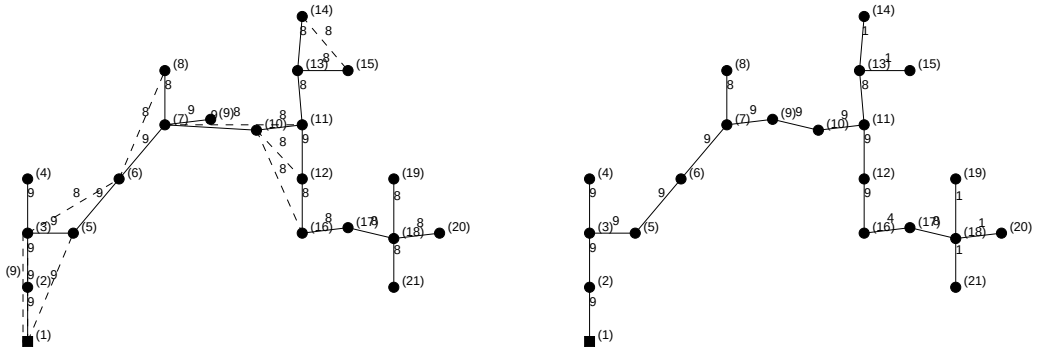


Figura 5.7: Sistema de 21 nós - Conjunto de Pareto

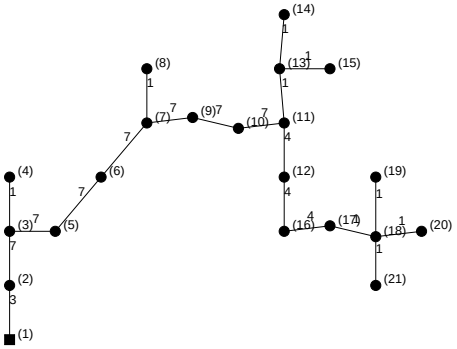
Sistema de 100 Nós

O conjunto de conexões possíveis para o sistema de 100 nós tratado foi determinado utilizando a *Controlled-Greedy Encoding*, o que resultou em um espaço com 397 variáveis (Fig. 5.9). Neste caso não é considerado nenhum sistema inicial para redimensionamento. Os dados referentes a este caso também podem ser encontrados na referência (Carrano et al., 2006a).

Foram mapeadas 132 soluções Pareto-ótimas, conforme mostrado na Fig. 5.10. A Fig. 5.11 apresenta a solução A do conjunto de Pareto, que é uma rede equilibrada dentre as encontradas.



(a) Rede A (b) Rede B



(c) Rede C

Figura 5.8: Sistema de 21 nós - Algumas redes do conjunto de Pareto

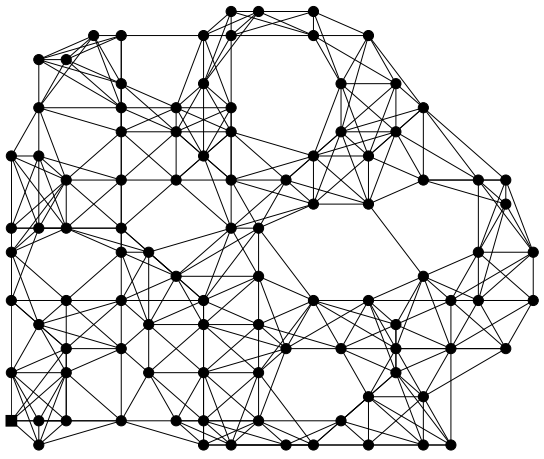


Figura 5.9: Sistema de 100 nós - Conexões possíveis

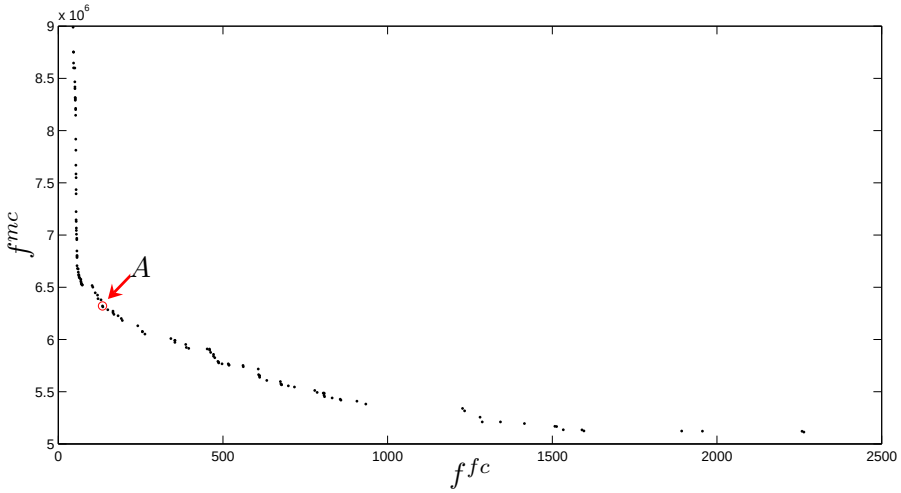


Figura 5.10: Sistema de 100 nós - Conjunto de Pareto

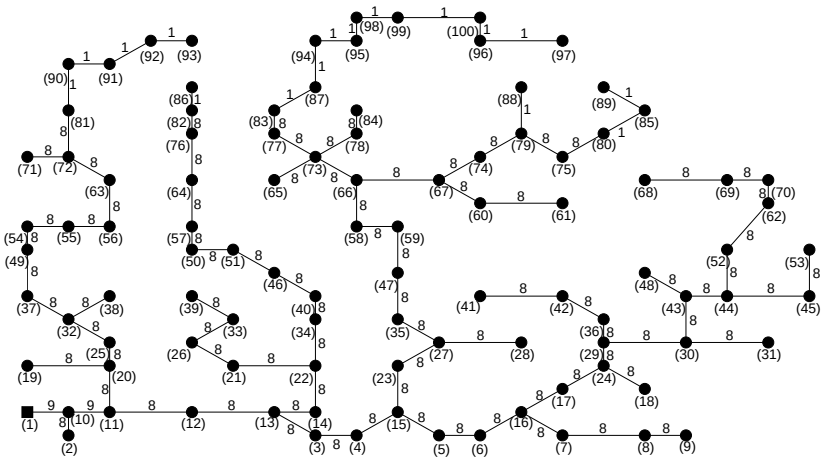


Figura 5.11: Sistema de 100 nós - Rede A

5.5 Localização de Subestações Associado ao Projeto da Topologia de Redes de Distribuição - Caso Mono-objetivo

[6]Como foi discutido ao longo deste capítulo, o sistema de distribuição de energia passa por modificações constantes, com o surgimento de novos centros consumidores, modificações dos padrões de carga existentes, mudanças na distribuição geográfica das cargas, etc. Isso faz com que, em alguns casos, as subestações (SS) existentes deixem de ser capazes de suprir a demanda de energia de uma determinada região. Cidades que experimentam altos índices de crescimento constituem um bom exemplo destes casos. Apesar desse fenômeno ocorrer com maior freqüência em países em desenvolvimento, as flutuações econômicas, intrínsecas do processo de globalização, também provocam deslocamentos demográficos em países desenvolvidos.

Os sistemas elétricos que operam em regiões onde essas alterações ocorrem devem ser capazes de suportar o eventual crescimento de carga, e ainda ponderar o compromisso entre a minimização do custo presente de instalação e minimização da necessidade de um re-projeto futuro do sistema. Nesses casos, novas subestações devem ser instaladas e o sistema de distribuição deve ser re-projetado, ou completamente substituído em alguns casos críticos, para suprir a nova demanda de carga.

Para esta situação de projeto, a expansão do sistema involve dois sub-problemas:

- Localização da subestação;
- Projeto da topologia da rede.

Estes problemas não devem ser tratados separadamente, uma vez que eles apresentam um forte interação: a mudança da solução de um, geralmente implica na alteração da solução do outro.

[6]Os resultados apresentados nesta seção foram retirados da referência (Carrano et al., 2005).

A localização da subestação é um problema de otimização de variáveis contínuas (as coordenadas geográficas da SS). Já o projeto da topologia da rede, como discutido no Cap. 1, é um problema combinatório, e portanto, necessariamente discreto. A iteração entre os problemas é clara: a mudança na localização da SS afeta o custo dos condutores ligados diretamente a ela, os quais têm seu custo alterado. Afeta também, indiretamente, as perdas em toda a rede. ReVelle and Eiselt (2005) discute a classe geral de problemas que envolvem a otimização de variáveis contínuas e discretas de forma conjunta.

Nesta seção é apresentado um algoritmo evolucionário híbrido, chamado de *GA-BFGS*, que combina uma versão mono-objetivo do *NSGA-PS*, apresentado na Sec. 5.4, e um algoritmo quasi-Newton BFGS. O algoritmo proposto é capaz de tratar de forma eficiente ambos os sub-problemas citados, considerando o acoplamento entre os mesmos. Por fim são apresentados os resultados obtidos por este algoritmo na solução de dois problemas: um sistema real de 8 nós e um sistema fictício de 50 nós.

5.5.1 Formulação do Problema

A formulação apresentada em 5.1.1 deve ser adaptada para a solução deste novo problema. Uma vez que duas novas variáveis são consideradas para otimização (coordenadas x e y da SS) é necessário ter em conta as restrições que tratam das mesmas (Eq. (5.7)).

$$\begin{aligned} c_5 &: x_{MIN} \leq x_{SS} \leq x_{MAX} \\ c_6 &: y_{MIN} \leq y_{SS} \leq y_{MAX} \\ c_7 &: (x_{SS}; y_{SS}) \neq (x_{IMP}; y_{IMP}) \end{aligned} \tag{5.7}$$

onde:

$(x_{SS}; y_{SS})$ são as coordenadas da SS;

$(x_{IMP}; y_{IMP})$ são as coordenadas de acidentes geográficas;

x_{MIN}, x_{MAX} , y_{MIN} e y_{MAX}) são os limites para as coordenadas x_{SS} e y_{SS}.

A formulação apresentada na Sec. 5.1.1, quando adaptada à este problema, pode ser definida por:

$$\begin{bmatrix} N^* \\ (x^*_{SS}; y^*_{SS}) \end{bmatrix} = \arg \min_{N,(x_{SS};y_{SS})} f^{mc}(N, x_{SS}, y_{SS})$$

$$\text{sujeito a:} \begin{cases} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{cases}$$

5.5.2 Algoritmo Proposto

Cada uma das partes do algoritmo híbrido (GA e BFGS) possui uma finalidade específica. O GA é responsável por encontrar a topologia ótima, considerando fixas as coordenadas da SS. O BFGS busca a melhor posição para a SS. A busca unidimensional, que é executada dentro do algoritmo quasi-Newton, considera a topologia da rede fixa. Ao fim da busca unidimensional, dentro do *loop* principal do algoritmo BFGS, a topologia ótima é encontrada para a localização da SS correspondente. Isto garante o acoplamento entre os problemas. A estrutura completa do algoritmo é apresentada abaixo.

Algoritmo GA-BFGS

1: $H_0 \leftarrow I$;

2: $k \leftarrow 0$;

3: $CT \leftarrow \text{TopOtima}(x_0)$;

4: $\nabla f^{mc}(CT, x_0) \leftarrow \text{gradiente}(CT, x_0)$;

5: **while** $\|x_{k+1} - x_k\| < \varepsilon$ **do**

6: $\alpha^* \leftarrow \arg\min_{\alpha} f^{mc}(CT, x_k - \alpha \cdot H_k \cdot \nabla f^{mc})$;

7: $x_{k+1} \leftarrow x_k - \alpha^* \cdot H_k \cdot \nabla f^{mc}$;

8: $\nabla f^{mc}(CT, x_{k+1}) \leftarrow \text{gradiente}(CT, x_{k+1})$;

9: $r_k \leftarrow x_{k+1} - x_k$;

10: $v_k \leftarrow \nabla f^{mc}(CT, x_{k+1}) - \nabla f^{mc}(CT, x_k)$;

11: $H_{k+1} \leftarrow H_k + \left(1 + \frac{r'_k \cdot H_k \cdot r_k}{r'_k \cdot v_k}\right) \cdot \frac{v_k \cdot v'_k}{v'_k \cdot v_k} - \frac{v_k \cdot r'_k \cdot H_k + H_k \cdot r_k \cdot v'_k}{r'_k \cdot v_k}$;

12: $NT \leftarrow TopOtima(x_{k+1})$;

13: **if** $f^{mc}(NT, x_{k+1}, ntop) < f^{mc}(CT, x_{k+1})$ **then**

14: $CT \leftarrow NT$;

15: **end if**

16: $k \leftarrow k + 1$;

17: **end while**

Onde:

x_0 é a posição inicial da SS;

I é a matriz identidade;

TopOtima(x_k) retorna a topologia ótima para a posição x_k, usando o GA;

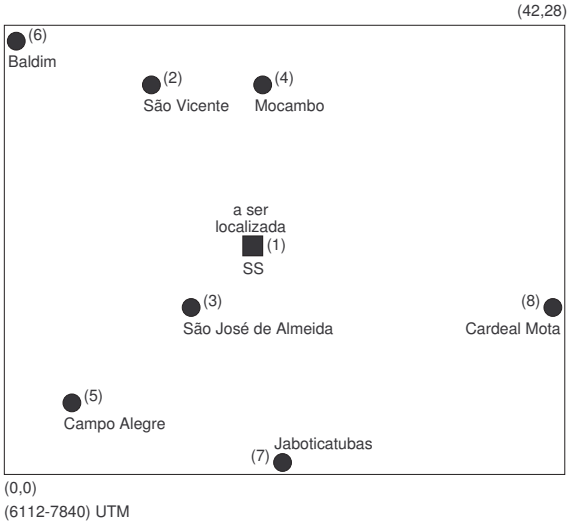
gradiente(CT, x_k) retorna o gradiente de f^{mc} para as coordenadas x_k e topologia CT.

5.5.3 Resultados

Sistema Real

O algoritmo desenvolvido foi aplicado no projeto da expansão de um sistema real, enfrentado pela Companhia de Energia Elétrica de Minas Gerais (CEMIG). As Figs. 5.12(a) e 5.12(b) apresentam a configuração dos nós e a foto carta deste sistema. A região, que possui baixa densidade populacional, vem sofrendo um grande crescimento tanto na atividade econômica quanto na população, tornando o sistema elétrico insta-

lado insuficiente. Um estudo rigoroso apontou a necessidade de uma nova SS na região, e a substituição de todos os cabos instalados.



(a) Configuração do nós

(b) Foto carta

Figura 5.12: Caso real

Todas as simulações foram realizadas considerando o centro de carga como a posição inicial da SS. Foram analisados horizontes de tempo de 1 à 20 anos, visando estimar o impacto do tempo de projeto na solução final. Dois condutores são disponibilizados para projeto. Os dados referentes à este caso podem ser encontrados na referência (Carrano et al., 2005).

É interessante perceber que a posição da SS muda com o horizonte de tempo considerado. Esse fenômeno acontece devido a mudança na relação custos fixos / custos

variáveis da rede. Para horizontes de tempo curtos, os custos fixos são sensivelmente maiores que os variáveis, se tornando predominantes no processo de otimização. O aumento do tempo de projeto implica em uma redução na importância dos custos fixos em relação aos custos variáveis. A Fig. 5.13 mostra esse fenômeno, indicando os pontos de troca de topologia.

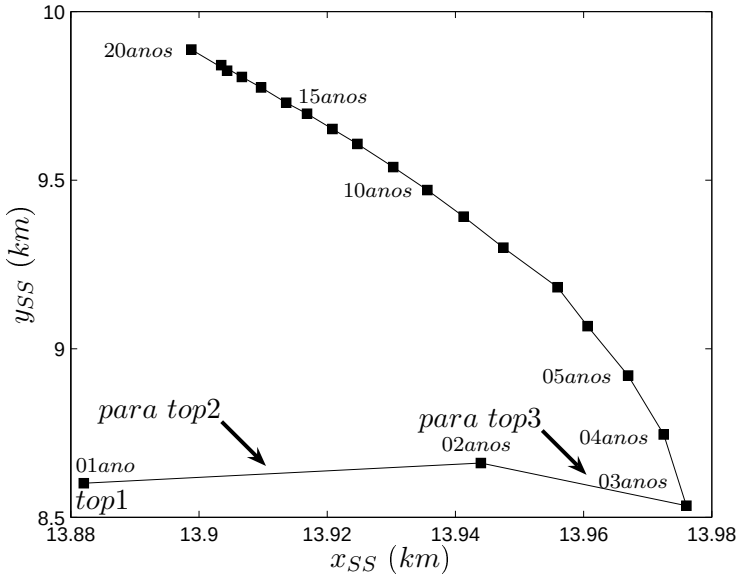
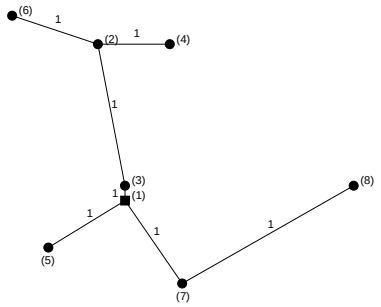


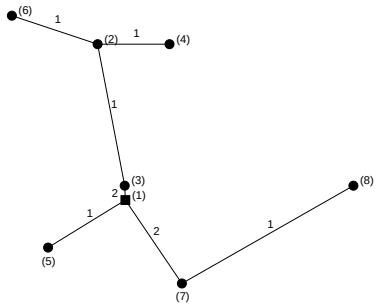
Figura 5.13: Sistema real - Posição da SS para diferentes horizontes de tempo

As Figs. 5.14(a), 5.14(b) e 5.14(c) apresentam as topologias indicadas na Fig. 5.13. Os *labels para top*2 e *para top*3 indicam os pontos onde ocorrem trocas de topologia (entre 1-2 e 2-3 anos respectivamente). Pode-se notar que entre 3 e 20 anos a topologia permanece a mesma, ocorrendo apenas variações na posição da SS.

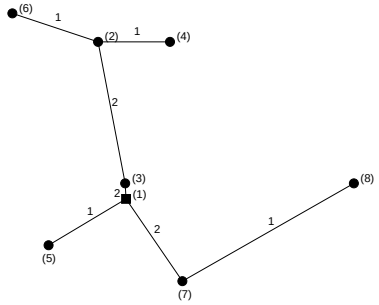
A Fig. 5.15 mostra o custo acumulado das redes resultantes (configuração ótima da rede e posição da SS) para alguns horizontes de tempo considerados. Pode-se ver que cada rede é de fato menos onerosa que as outras para o horizonte de tempo em que ela foi projetada: a rede de 01 ano tem menor custo acumulado em um ano, a rede de 02 anos tem menor custo acumulado em dois anos, e assim por diante. Redes com configurações similares (mesma topologia e localização da SS próxima) tem projeções



(a) Topologia 1 (*top*1)



(b) Topologia 2 (*top*1)



(c) Topologia 3 (*top*3)

Figura 5.14: Sistema real - Topologias

de custo muito similares, como era esperado. O custo acumulado de redes projetadas para horizontes curtos cresce rapidamente. Em redes projetadas para 10 e 20 anos a taxa de incremento é praticamente a mesma, o que se deve à influência da taxa de juros.

O custo inicial da rede de 20 anos é mais que 5% maior que a o custo da rede projetada para 1 ano. Este tipo de análise pode ser usada para ajudar o projetista na escolha da melhor rede a ser implantada, tendo em vista os recursos financeiros disponíveis.

Por fim, é importante destacar que as soluções obtidas pelo algoritmo proposto foram muito melhores que as soluções obtidas por engenheiros utilizando as técnicas tradicionais. A solução a ser implantada na prática deve ser a topologia 3, com a localização da SS situada entre a encontrada para 5 anos e a encontrada para 10 anos. Esta decisão é recomendada, uma vez que a região se encontra ainda em desenvolvimento, e

espera-se que, para um tempo de projeto mais longo, a nova configuração de carga faça necessário um re-projeto do sistema.

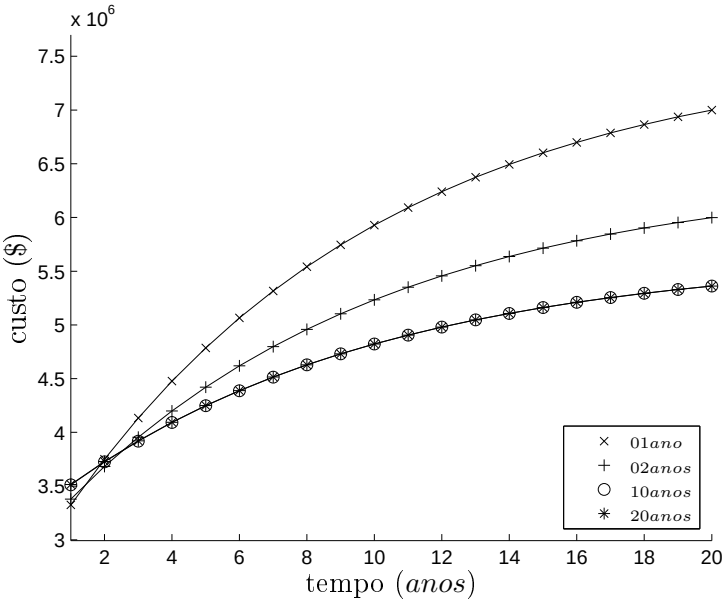


Figura 5.15: Sistema real - Custo acumulado, ao longo do anos, das redes ótimas para diferentes horizontes de tempo.

Sistema Fictício

O algoritmo proposto também foi aplicado a um problema fictício de 50 nós, visando avaliar seu desempenho em problemas de maior porte. Foram considerados horizontes de tempo entre 1 e 15 anos, sendo disponibilizados dois tipos de condutores para projeto. A posição inicial da subestação foi mais uma vez definida para o centro de carga do problema. Os dados referentes à este caso podem ser encontrados na referência (Carrano et al., 2005).

A Fig. 5.16 mostra a posição da SS para cada horizonte de tempo considerado, e as Figs. 5.17(a) e 5.17(b) ilustram as topologias indicadas. O ponto marcado com *para top*2 na Fig. 5.16 indica uma troca de topologia, que ocorre entre 4 e 5 anos. Nota-se que a posição da SS muda consideravelmente entre 4 e 5 anos, o que provavelmente se deve à alteração da topologia.

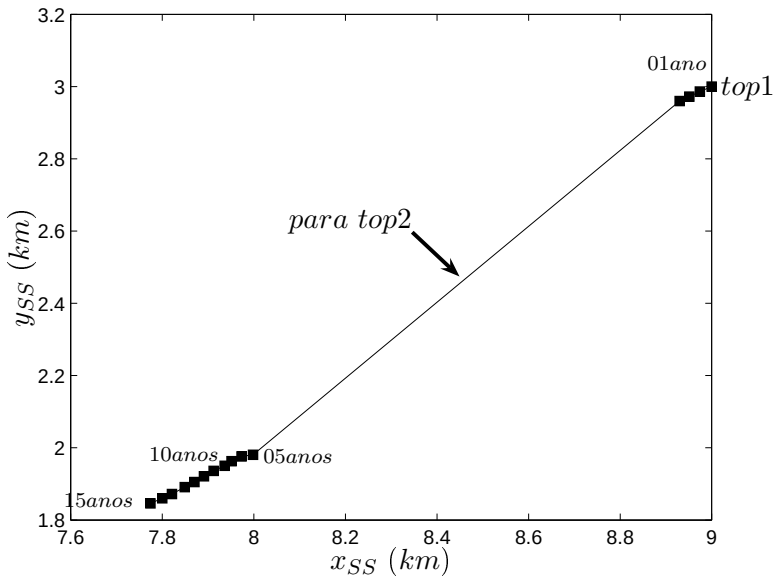
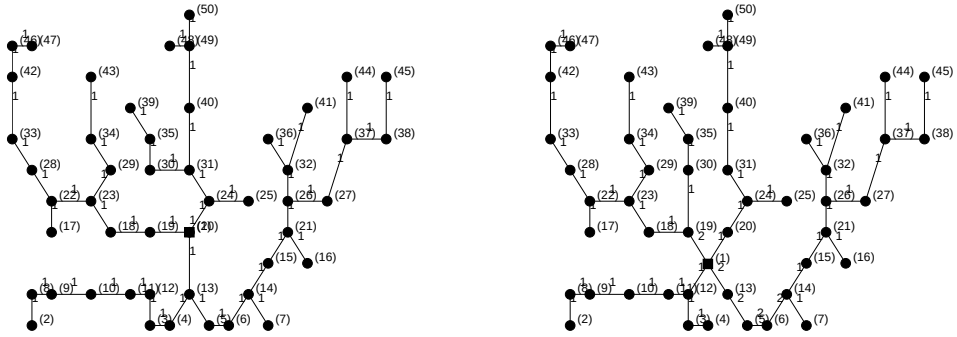


Figura 5.16: Sistema fictício - Posição da SS para diferentes horizontes de tempo

A Fig. 5.18 mostra o custo acumulado das redes ótimas encontradas para alguns horizontes de tempo considerados. Deve-se notar que o custo inicial da rede de 10 anos (que apresenta um custo inicial quase idêntico ao da rede de 15 anos) é mais que 3% maior que o da rede projetada para 1 ano.



(a) Topologia 1 ($top1$) (b) Topologia 2 ($top2$)

Figura 5.17: Sistema fictício - Topologias

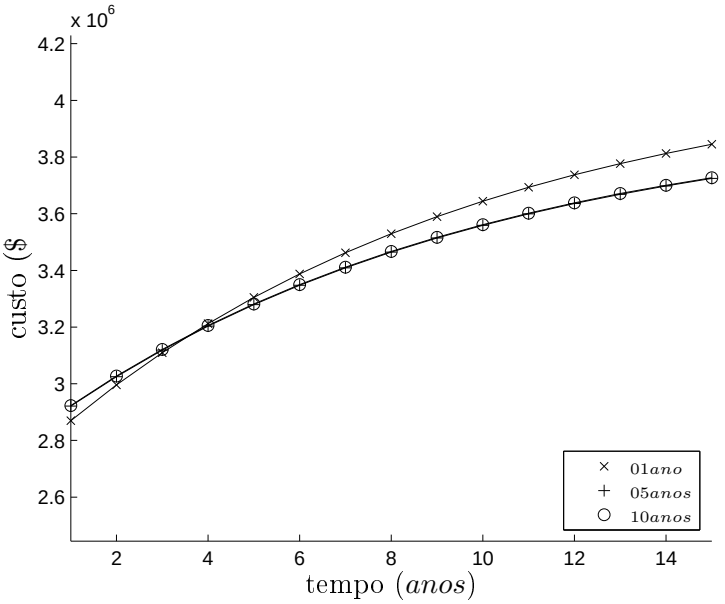


Figura 5.18: Sistema fictício - Custo acumulado, ao longo do anos, das redes ótimas para diferentes horizontes de tempo.

5.6 Localização de Subestações Associado ao Projeto da Topologia de Redes de Distribuição - Caso Multi-objetivo

[7]O problema apresentado na seção anterior pode ser estendido para um abordagem multi-objetivo (considerando a confiabilidade como segundo critério de projeto), desde que o algoritmo *GA-BFGS* seja adaptado. Ao longo desta seção é discutida a estrutura do problema multi-objetivo de localização de subestações associado ao projeto da topologia de redes (referido em (Carrano et al., 2007e) como *Multi-objective Joint Facility Location and Network Design*, ou *MJFLND*). É proposta uma extensão do algoritmo *GA-BFGS*, que combina um método baseado em direções de busca para posicionamento da SS e um GA para projeto da topologia. Ambas as rotinas foram adaptadas para gerar o conjunto de Pareto em suas variáveis. Uma heurística é aplicada para realizar a execução iterativa destas rotinas, visando considerar o acoplamento entre os problemas e permitir a obtenção de todo o conjunto de Pareto. Por fim, é apresentado um critério de convergência baseado na estabilização das ilhas do conjunto de Pareto, que caracterizam o problema proposto.

São apresentados os resultados obtidos pelo algoritmo no mesmo sistema real, de oito nós, apresentados na seção anterior.

5.6.1 Formulação do Problema

O problema *MJFLND* apresenta a seguinte formulação:

$$\begin{bmatrix} \mathcal{N}^* \\ (\mathcal{X}^*_{SS}; \mathcal{Y}^*_{SS}) \end{bmatrix} = \arg \min_{N,(x_{SS};y_{SS})} \begin{Bmatrix} f^{mc}(N, x_{SS}, y_{SS}) \\ f^{fc}(N, x_{SS}, y_{SS}) \end{Bmatrix}$$

[7]Os resultados apresentados nesta seção foram retirados da referência (Carrano et al., 2007e).

$$\text{sujeito a:} \begin{cases} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{cases}$$

onde:

$(\mathcal{X}^*_{SS}; \mathcal{Y}^*_{SS})$ é o conjunto de posições não-dominadas das topologias de $\mathcal{N}^*$.

Assim como no caso mono-objetivo, o conjunto de soluções deste problema é composto por dois conjuntos de variáveis de naturezas distintas: posição da SS (contínua) e topologia da rede (discreta). Isso faz com que o *MJFLND* possa ser decomposto em dois sub-problemas:

- Encontrar o conjunto de topologias não-dominadas (problema multi-objetivo discreto);
- Encontrar o conjunto de posições não-dominadas para cada topologia (problema multi-objetivo contínuo).

As características gerais dos conjuntos de Pareto usualmente encontrados em problemas contínuos e discretos são discutidas nos Caps. 3 e 2. Neste caso, é razoável assumir que a fronteira Pareto ($\mathcal{Y}^*$) seja composta de conjuntos contínuos disjuntos no espaço de objetivos: o sub-problema discreto dá origem a pontos isolados, enquanto que, o sub-problema contínuo leva a superfícies contínuas associadas a esses pontos. A Fig. 5.19 mostra um exemplo de fronteira Pareto idealizada para este problema.

Cada parte contínua da fronteira Pareto é gerada para cada topologia específica e com a posição da SS variando continuamente. É importante notar que o deslocamento da SS em uma topologia específica pode eventualmente gerar soluções que não são eficientes, sendo dominadas por outras soluções associadas a outras topologias. Neste

caso, a fronteira Pareto "pula" para a superfície contínua associada a outra topologia. Este fenômeno pode ser visto na Fig. 5.19, nos pontos onde há mudança de topologia.

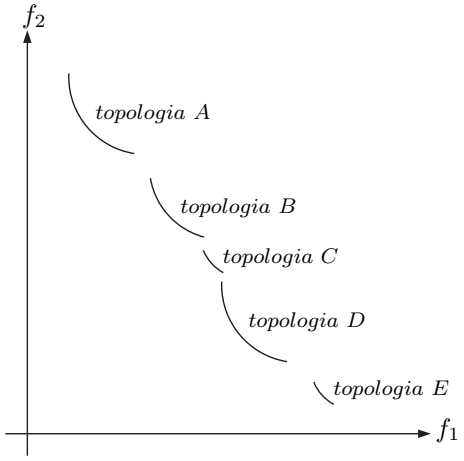


Figura 5.19: Exemplo de fronteira Pareto para o problema *MJFLND*

5.6.2 Algoritmo Conceitual

Como discutido no caso mono-objetivo, os sub-problemas que compõem o *MJFLND* não podem ser tratados separadamente, uma vez que apresentam uma forte interação. Por outro lado, a diferença da natureza dos problemas justifica o uso de métodos de otimização distintos de otimização. Isso faz com que seja necessária a elaboração de estratégias capazes de tratar este acoplamento de forma iterativa, tendo ainda em conta o contexto multi-objetivo. A estratégia proposta neste trabalho é apresentada abaixo:

1. Escolher uma posição inicial para a SS; definir o contador $i \leftarrow 1$;

2. Encontrar o conjunto de Pareto discreto das topologias, mantendo a posição da SS fixa. Isto resulta em um conjunto de topologias $\mathcal{T}_i$, com a j-ésima topologia do conjunto denotada por $\mathcal{T}_i(j)$. ;

3. Para cada topologia não-dominada em $\mathcal{T}_i$ encontrar conjuntos locais de posições eficientes da SS, sendo o k-ésimo ponto do j-ésimo conjunto local denotado por

$\mathcal{L}_i(j,k)$. Ordenar os pontos dos conjuntos locais utilizando f^{mc}, de forma que $\mathcal{L}_i(j,1)$ é associada ao menor valor de f^{mc} dentre os pontos do conjunto e $\mathcal{L}_i(j,\alpha_j)$ é associada ao maior valor de f^{mc};

4. Executar uma busca local por topologias eficientes para os pontos extremos de cada conjunto, $\mathcal{L}_i(j,1)$ e $\mathcal{L}_i(j,\alpha_j)$, considerando a posição da SS fixa;

5. Realizar uma análise de dominância considerando o conjunto de Pareto obtido até o momento e as novas soluções, para atualizar o conjunto de Pareto atual;

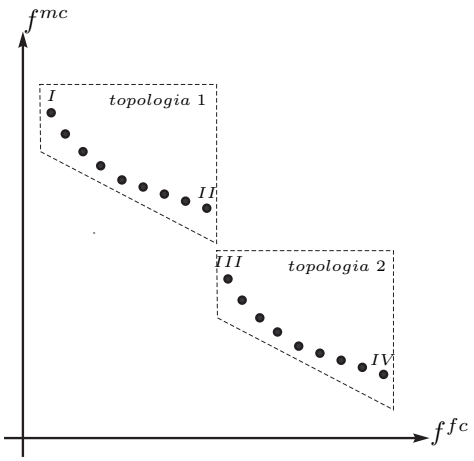
6. Se o conjunto de Pareto não sofreu alterações após os passos 4 e 5, então parar o algoritmo. Caso contrário, fazer $i \leftarrow i+1$, definir o novo conjunto de topologias como $\mathcal{T}_i$ e retornar ao passo 3.

O passo 2 estima uma aproximação inicial do conjunto de Pareto, composto por pontos disjuntos. Este é o único ponto em que uma busca global é executada pelo algoritmo. As outras buscas por topologias ótimas são executadas utilizando algoritmos locais, computacionalmente mais baratos.

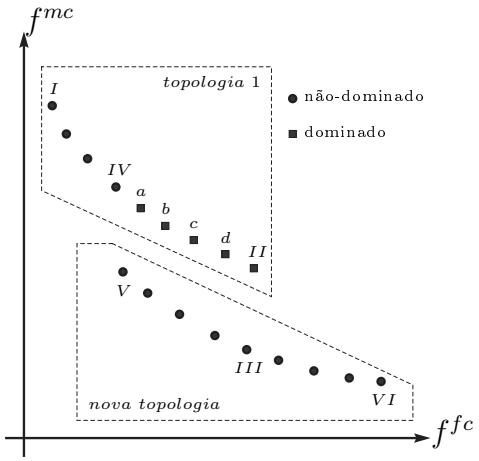
O passo 3 é responsável por encontrar as superfícies contínuas associadas a cada topologia da fronteira Pareto.

No passo 4, apenas os pontos extremos de cada topologia são analisados (a Fig. 5.20(a) ilustra esse processo). Esta é uma aproximação aceitável, uma vez que estas são as soluções com maiores chances de encontrar novas topologias para suas posições. Quando uma nova topologia é encontrada, ela pode dominar algumas soluções de outras topologias, mudando os pontos extremos de ambas (Fig. 5.20(b)).

Se os pontos extremos de uma certa topologia não mudam em duas iterações consecutivas, esta topologia é considerada estável. Topologias estáveis não são analisadas nas iterações subseqüentes, a não ser que sejam afetadas pela análise de dominância, devido a mudanças ocorridas em topologias vizinhas. O critério de convergência é alcançado quando todas topologias se tornam estáveis.



(a) Neste exemplo, apenas os pontos I e II são analisados para a *topologia* 1, e os pontos III e IV são analisados para a *topologia* 2.



(b) Suponha que o algoritmo local encontrou uma nova topologia, obtendo o ponto III à partir da posição II (*topologia* 1). As posições V à VI foram encontrados para a nova topologia, à partir de III. Estas soluções dominam as soluções a, b, c, d e II. Neste caso, os pontos extremos da *topologia* 1 se tornam I e IV e os pontos extremos da *nova topologia* se tornam V e VI.

Figura 5.20: Soluções analisadas pelo algoritmo no *MJFLND*

5.6.3 O Algoritmo *GA-BFGS* Multi-objetivo

O algoritmo conceitual apresentado acima foi utilizado como base para construção de um algoritmo capaz de realizar o posicionamento multi-objetivo de subestações associado ao projeto da topologia da rede. Este algoritmo, que é referido como *GA-BFGS* multi-objetivo (ou *MO GA-BFGS*) é composto por quatro módulos isolados:

- *GA-BFGS*: o algoritmo GA-BFGS mono-objetivo, apresentado na Sec. 5.5, é utilizado para encontrar uma posição inicial para a SS e a rede de mínimo custo para esta posição;

- *NSGA-PS* global: o *NSGA-PS*, discutido na Sec. 5.4, é aplicado para estimar uma amostragem inicial das topologias Pareto-ótimas. Este algoritmo permite a busca em todo o espaço de soluções do problema.

- *NSGA-PS* local: o mesmo *NSGA-PS* é utilizado para busca local de topologias. A versão local deste algoritmo trabalha sobre uma população local, com menos indivíduos, visando reduzir seu custo computacional.

- quasi-Newton *BFGS* multi-objetivo: O algoritmo *BFGS* apresentado na Sec. 2.2.1 foi utilizado para estimar as superfícies contínuas associadas à cada uma das topologias eficientes. A abordagem híbrida, que combina Pλ e Pϵ-restrito, foi associada ao mesmo, para torná-lo capaz de mapear o conjunto de Pareto neste problema.

Estes módulos são executados de forma conjunta, seguindo o esquema apresentado na Fig. 5.21. As principais características do algoritmo conceitual, como análise de convergência, estabilização local e global e pontos analisados, são válidas para este algoritmo.

5.6.4 Resultados

O algoritmo proposto foi aplicado na solução do sistema real de 8 nós, apresentado na Fig. 5.12. O tempo de projeto considerado neste caso foi de cinco anos, e o número máximo de ramos de reserva foi limitado em 5. Quatro tipos de cabos podem ser usados no projeto: os mesmos dois utilizados no caso mono-objetivo e outros dois cabos cobertos, com maior índice de confiabilidade. Os dados completos do sistema tratado podem ser encontrados nas referências (Carrano et al., 2005, 2007e).

A Fig. 5.22(a) apresenta a fronteira Pareto que foi obtida em uma execução do algoritmo proposto. Foram mapeadas 4.145 soluções, correspondentes a 57 topologias distintas.

Pode-se notar que algumas partes das fronteira Pareto são praticamente contínuas, como por exemplo a região *a* (a Fig. 5.22(b) mostra um *zoom* desta região). Esta região apresenta várias posições Pareto-ótimas da SS, obtidas pelo algoritmo *BFGS* multi-objetivo para a topologia de mínimo custo.

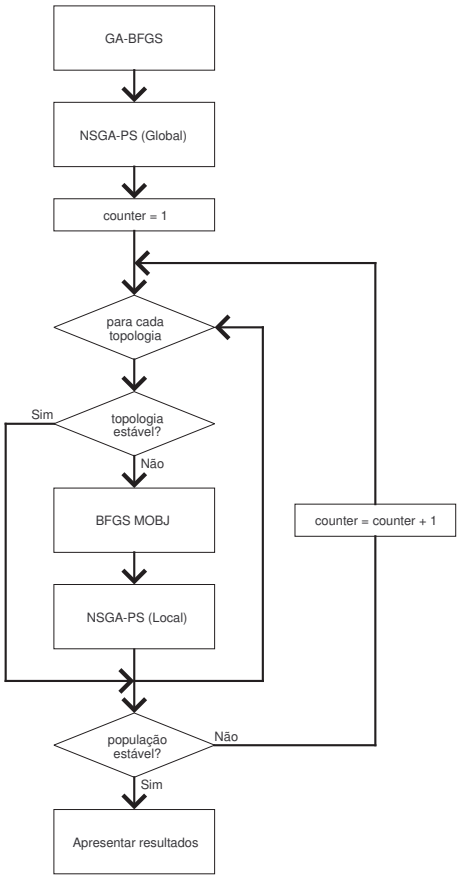
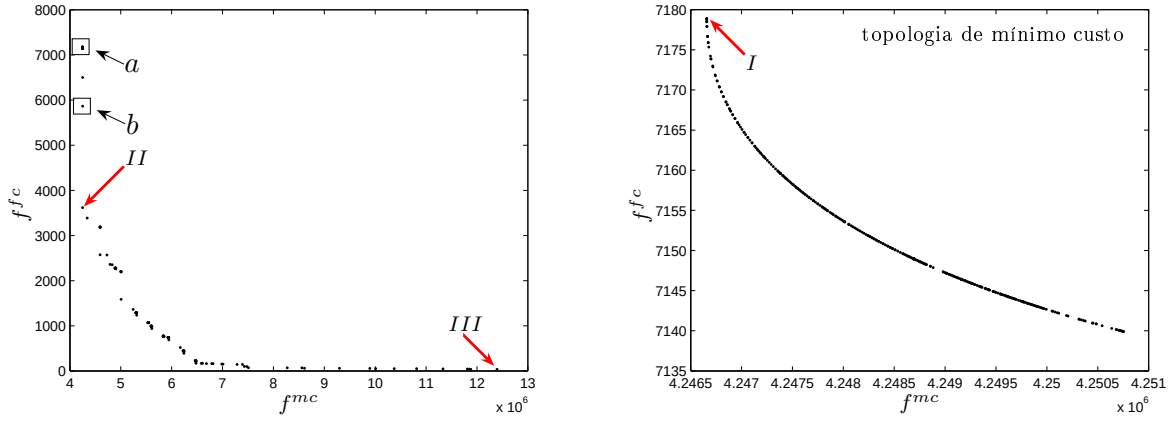


Figura 5.21: Algoritmo GA-BFGS multi-objetivo

Por outro lado, esta "parte contínua" não ocorre para algumas topologias, como é o caso da região *b*. Isto é esperado em alguns casos, uma vez que, para algumas topologias específicas, os custos monetários e os custos de falta da rede são estritamente dependentes do comprimento dos cabos. Portanto, para estas topologias, ambas as funções possuem o mesmo ótimo, resultando em um único ponto não dominado. Em outras topologias, as perdas apresentam um efeito considerável no custo da rede, deslocando a SS da "posição de mínimo comprimento dos cabos" e resultando em múltiplos pontos no conjunto de Pareto.

É importante notar que pequenas mudanças na rede (que usualmente implicam em pequenas variações do custo monetário) podem resultar em grandes incrementos na confiabilidade do sistema. Por exemplo, a solução II (Fig. 5.23(b)) tem um custo

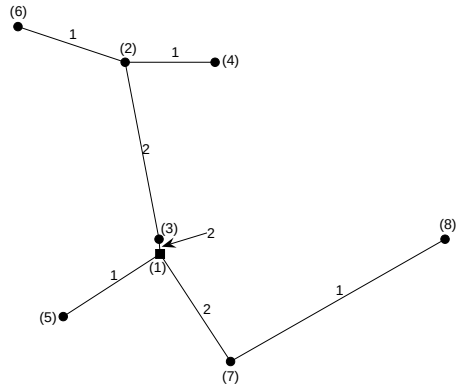
monetário apenas 0.1% maior que a solução de mínimo custo (solução I, Fig. 5.23(a)) e um custo de falta 50% menor. A rede de mínimo custo de falta (solução III) é ilustrada na Fig. 5.23(c).



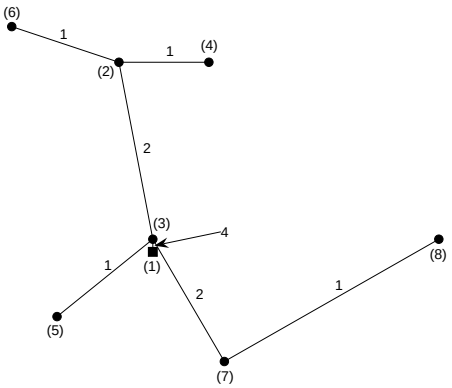
(a) Sistema real - Fronteira Pareto

(b) Zoom da região a

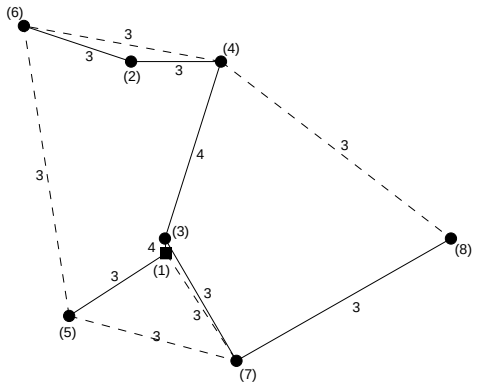
Figura 5.22: Sistema real - Fronteira Pareto



(a) Rede com mínimo custo monetário (*sol. I*)



(b) Rede alternativa (*sol. II*)



(c) Rede com mínimo custo de falta (*sol. III*)

Figura 5.23: Sistema real - Algumas soluções do conjunto de Pareto

5.7 *Scheduling* da Expansão de Sistemas de Distribuição Utilizando o *Dynamic Programming Genetic Algorithm* (*DP-GA*)

[8]As alterações constantes a que o sistema de distribuição está sujeito usualmente implicam na necessidade de sucessivas expansões (Carvalho et al., 1998; Vaziri et al., 2004; Levitin and Lisnianski, 1999), o que, rigorosamente falando, fazem deste problema um problema de programação dinâmica (do inglês *Dynamic Programming* ou *DP*) (Bellman, 1957; Bertsekas, 1995). Isto significa que existe uma seqüencia de ações de projeto (a expansão em cada passo) que são inter-dependentes: o projeto da expansão em um estágio afeta a definição do problema para os estágios subseqüentes.

A complexidade computacional associada a estes problemas é geralmente muito alta, uma vez que o espaço de variáveis é definido pelo número de variáveis de decisão em cada estágio multiplicado pelo número de estágios. Estas dificuldades provavelmente explicam a escassez de trabalhos que tratam o planejamento da expansão de sistemas de distribuição utilizando programação dinâmica. Exemplos da aplicação de programação dinâmica na solução de problemas relacionados podem ser encontrados nas referências Vaziri et al. (2004); Yehia et al. (2002); Monteiro et al. (2005)

Nesta seção, o problema de projeto da expansão de sistemas de distribuição é modelado usando os conceitos de programação dinâmica, com o sistema sendo expandido por passos incrementais no tempo. Conceitualmente, esta abordagem permite encontrar a política de expansão ótima, uma vez que o espaço de variáveis do problema de programação dinâmica inclui este ótimo. Um algoritmo genético, com operadores específicos de mutação e cruzamento (aqui chamado de *DP-GA*), é proposto como meta-heurística para solução deste problema. São apresentados os resultados obtidos pelo algoritmo na expansão de um sistema de 100 nós.

[8]Os resultados apresentados nesta seção foram retirados da referência (Carrano et al., 2007a).

5.7.1 *Scheduling* da Expansão de Sistemas de Distribuição

Uma rede de distribuição de energia elétrica pode ser representada como um sistema dinâmico, onde as variáveis do sistema são suas conexões. O sistema deve ser descrito como um sistema dinâmico linear e invariante no tempo, como mostrado em (5.8):

$$x[k] = x[k-1] + u[k] \quad \forall \quad k = 1, \ldots, N \tag{5.8}$$

onde:

k é o índice do estágio (tempo discreto);

N é o número total de estágios em que o tempo foi discretizado;

$x[k]$ é a rede no estágio k;

$u[k]$ é o incremento da rede no estágio k (novas instalações e redimensionamentos).

O *scheduling* da expansão da rede de distribuição é interpretado, neste caso, como a evolução do sistema dentro de um horizonte finito de tempo. O problema de programação dinâmica depende da definição de 4 aspectos:

- Sistema de distribuição inicial (rede inicial);
- Horizonte de tempo de projeto;
- Número de estágios considerados;
- Previsão de carga para cada estágio dentro do horizonte de tempo de projeto.

A metodologia proposta parte da determinação da "rede alvo": a rede que é ótima para o perfil de carga que é esperado para o fim do tempo de projeto. Os algoritmos apresentados nas Secs. 4.4, 5.4, 5.5 e 5.6 são exemplos de ferramentas que podem ser utilizadas para determinação dessa rede. O algoritmo proposto é então aplicado para determinação do *scheduling* de expansão, que leva o sistema inicial até a rede alvo determinada.

Seja $g_k(x[k], u[k])$ o custo da rede no estágio k (incluindo o custo dos incrementos na rede, $u[k]$). O custo total da evolução do sistema dinâmico, em uma formulação geral, é:

$$J = \sum_{k=1}^{N} g_k(x[k], u[k]) \tag{5.9}$$

No caso específico deste trabalho, a função objetivo apresenta a seguinte formulação

$$f^{dpds}(U) = \sum_{k=1}^{N} [\text{inst cost}(u[k]) + \text{oper cost}(x[k])] \cdot (1 - int^{rt})^{k-1} \tag{5.10}$$

onde:

f^{dpds} é o custo presente do sistema;

inst cost$(u[k])$ são os custos de instalação e redimensionamento no estágio k;

oper cost$(x[k])$ são os custos de manutenção e perdas no estágio k;

int_{rt} é a taxa de juros[9].

A estrutura do problema de programação dinâmica implica na consideração de duas restrições:

$$\begin{aligned} c_8 &: x[k] = x[k-1] + u[k] \quad \forall \quad k = 1, \ldots, N \\ c_9 &: x[0] + \sum_{k=1}^{N} u[k] = x^* \end{aligned} \tag{5.11}$$

onde:

$x[0]$ é a rede inicial;

x^* é a rede alvo.

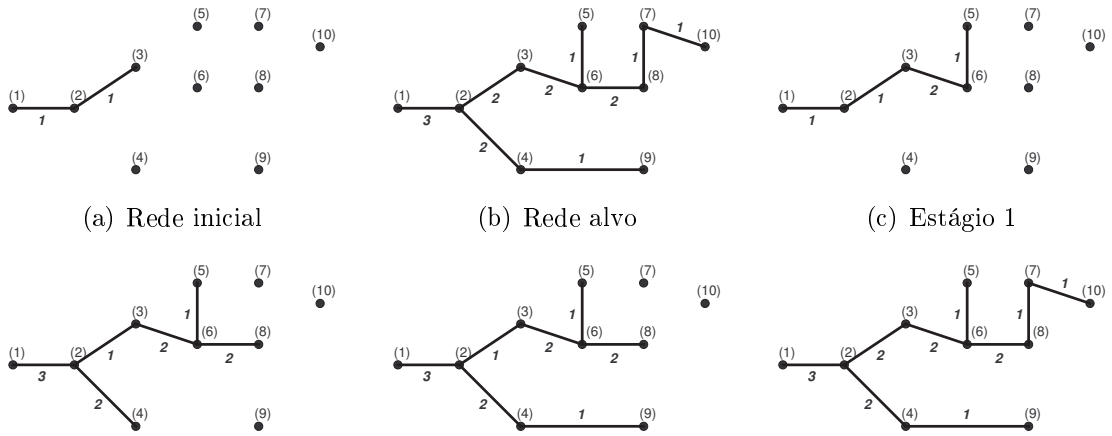
Estas duas restrições, juntamente com as quatro restrições apresentadas na Eq. (5.2) e a função objetivo (5.10) definem a formulação do problema de programação dinâmica, como apresentado abaixo.

$$U^* = \arg\min_{U} f^{dpds}(U)$$

[9] Os custos fixos e variáveis são determinados utilizando os mesmo procedimentos apresentados na Sec. 5.1.1.

$$\text{sujeito a:} \begin{cases} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_8 \\ c_9 \end{cases}$$

Essa formulação é ilustrada na Fig. 5.24. As Figs. 5.24(a) e 5.24(b) representam o sistema inicial e o sistema alvo respectivamente. Neste exemplo, o número de estágios foi definido em 4. As Figs. 5.24(c), 5.24(d), 5.24(e) e 5.24(f) mostram uma possível expansão para este caso.



(a) Rede inicial (b) Rede alvo (c) Estágio 1

(d) Estágio 2 (e) Estágio 3 (f) Estágio 4

Figura 5.24: Exemplo de *scheduling* da expansão de uma rede

Também deve-se levar em conta que as cargas do sistema também constituem um sistema dinâmico:

$$P_L[k] = P_L[k-1] \cdot \Phi[k] + N_L[k] \tag{5.12}$$

onde:

$P_L[k]$ são as cargas no estágio k;

$N_L[k]$ são as novas cargas conectadas ao sistema no estágio k;

$\Phi[k]$ é a taxa de crescimento esperada para as cargas existentes no estágio k.

5.7.2 *Dynamic Programming Genetic Algorithm - DP-GA*

Abordagens exatas para problemas de programação dinâmica são conhecidas por apresentar alto custo computacional, o que geralmente torna inviável sua aplicação em problemas de grande porte. Uma abordagem alternativa para o problema de programação dinâmica é o uso de aproximações que recursivamente estimam a solução do problema (Lincoln and Rantzer, 2006; Farias and Roy, 2003; Bertsekas, 1995).

Atualmente, o uso de meta-heurísticas, como algoritmos que combinam métodos evolucionários / programação dinâmica, vem sendo proposto como uma abordagem viável (Mak and Lam, 2005; Lakshminarasimman and Subramanian, 2006; Tse et al., 2007). Entretanto, a aplicação direta destes algoritmos no planejamento da expansão de redes não é necessariamente eficiente, uma vez que, como discutido na Sec. 3.4, eles apresentam sérias dificuldades na manutenção da factibilidade das soluções.

Na seqüência desta seção, é apresenta a codificação que foi elaborada para este problema. Os operadores de cruzamento, mutação e correção foram desenvolvidos tendo como base esta codificação. Eles garantem a factibilidade de todas as soluções da população, eliminando o problema da factibilidade, recorrente nos algoritmos tradicionais. Uma descrição detalhada da estrutura dos mesmos é apresentada na Sec. B.4 do Ap. B.

Uma vez que o DP-GA é um método de busca heurístico, não existe garantia da obtenção do ótimo neste método. No entanto, é esperado que o mesmo seja capaz de encontrar boas soluções em um tempo computacional razoável, mesmo para problemas de grande porte.

Codificação das Soluções

No *DP-GA*, cada solução candidata é representada pela matriz U, de dimensão $N \times m$ (N é o número de estágios e m é o número de conexões consideradas). Cada célula $u[i;j]$ representa a ação (nova instalação, redimensionamento e remoção do ramo) que deve ser executada na conexão i durante o estágio j. Portanto, $u[5;4] = 2$ significa que a conexão 4 recebe uma expansão de dimensão 2 no estágio 5. Esta expansão

pode ser de dois tipos: a instalação de um ramo de índice 2, caso não exista nenhum condutor na conexão 4; ou o redimensionamento da conexão pré-existente, aumentando sua capacidade em 2 índices.

A Fig. 5.25 mostra essa codificação, para o exemplo de expansão apresentado na Fig. 5.24.

	[$U[i,1]$	$U[i,2]$	$U[i,3]$	$U[i,4]$	$U[i,5]$	$U[i,6]$	$U[i,7]$	$U[i,8]$	$U[i,9]$]
do nó	1	2	2	3	4	5	6	7	7
para o nó	2	3	4	6	9	6	8	8	10

$$\begin{bmatrix} U[1] \\ U[2] \\ U[3] \\ U[4] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figura 5.25: Representação da expansão apresentada na Fig. 5.24

5.7.3 Métodos Não-dinâmicos

É possível desenvolver métodos não-dinâmicos capazes de lidar com o *scheduling* da expansão de sistemas de distribuição. Duas abordagens são descritas na seqüência.

Abordagem Incremental (Inc)

1. Discretizar o tempo de projeto em N estágios;
2. Para cada estágio k:
   (a) Atualizar as cargas (Eq. (5.12));
   (b) Obter $x[k]$ conectando os novos nós (se necessário) e redimensionando os ramos que se tornaram insuficientes.

Algoritmo Genético Não-dinâmico (ndGA)

1. Discretizar o tempo de projeto em N estágios;

2. Para cada estágio k:

   (a) Atualizar as cargas (Eq. (5.12));

   (b) Obter $x[k]$ através da execução de uma GA estático (específico para otimização de redes), considerando os custos imediatos como objetivo de projeto.

Ambas as abordagens descritas acima são capazes de planejar a expansão do sistema em múltiplos estágios. Entretanto, o projeto é realizado para cada estágio separadamente, em detrimento do contexto geral do problema. É ainda importante destacar que os três métodos discutidos podem levar a diferentes sistemas finais, o que implica em diferentes custos.

5.7.4 Resultados

DP-GA *vs* Abordagens Não-dinâmicas

Os três métodos descritos foram aplicados para *scheduling* da expansão de um sistema de 100 nós. Os seguintes parâmetros foram definidos para o sistema:

Tempo de projeto: 10 anos;

Nós pré-existentes: 70 nós;

Freqüência de adição de novos nós: anual;

Crescimento de carga anual esperado: 5.00% por nó existente;

Taxa de juros anual: 10.00%.

O sistema possui conexões pré-existentes, como apresentado na Fig. 5.26. A rede alvo do DP-GA foi encontrada utilizando o *GANet*. A representação gráfica das soluções encontradas pelas abordagens incremental, algoritmo genético não-dinâmico e *DP-GA* são apresentadas no Ap. C.

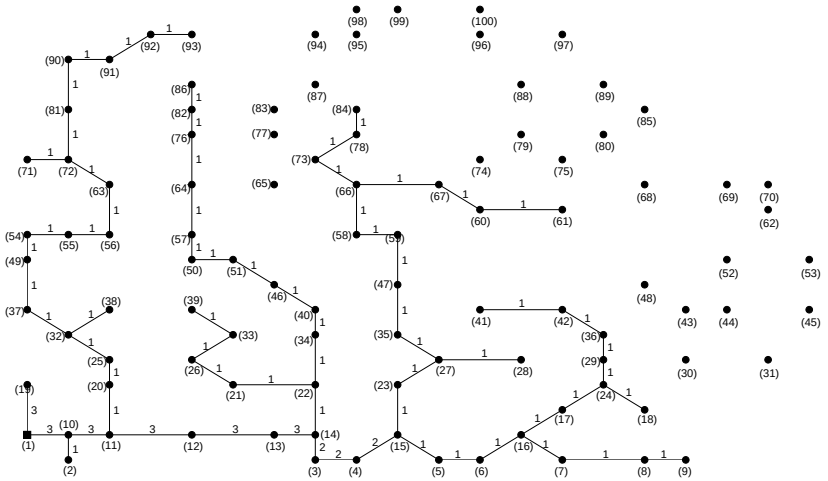


Figura 5.26: Sistema de 100 nós - Rede inicial ($x[0]$)

O custo monetário presente das três abordagens, para cada ano do tempo de projeto, é apresentado na Fig. 5.27. O *DP-GA* apresenta um custo inicial maior, o que era esperado, uma vez que as outras abordagens realizam o projeto considerando apenas as condições imediatas do sistema. Após alguns estágios, os custos acumulados das outras abordagens se tornam mais altos que os do *DP-GA*, que considera as condições globais do sistema durante todo o processo de otimização. Apesar da abordagem incremental ter obtido o *scheduling* mais caro, esta é a mais rápida, uma vez que não requer a execução de nenhum algoritmo de otimização. Por outro lado, o algoritmo genético não-dinâmico é consideravelmente mais lento que o *DP-GA*. Isto pode ser explicado pela necessidade da execução do algoritmo de otimização N vezes (o GA deve ser executado para cada estágio) ao invés de 2 execuções no *DP-GA* (primeiramente, o *GANet* é executado para determinação da rede alvo, e então o *DP-GA* é utilizado para determinar o *scheduling* da expansão)[10].

Pode-se concluir que o uso do *DP-GA* é amplamente justificável, uma vez que ele obteve um sistema que é cerca de 15% mais barato que o obtido pelo GA não-dinâmico

[10]Para o problema tratado, ambos os algoritmos possuem custos computacionais similares.

e 35% mais barato que a abordagem incremental.

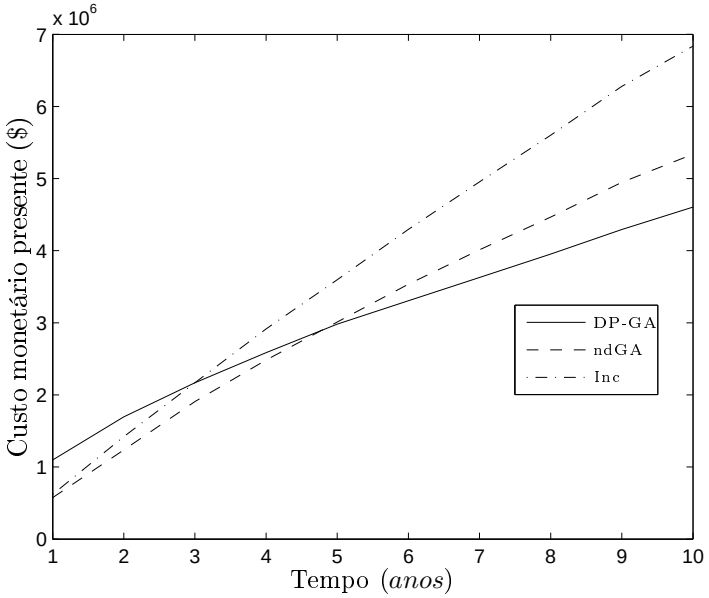


Figura 5.27: Sistema de 100 nós - Comparação das abordagens

Efeitos da Discretização do Tempo na Qualidade da Solução

Nove níveis de discretização do tempo de projeto foram testados, 1 (que equivale a uma abordagem estática), 2, 5, 10, 20, 30, 40, 60 e 120 estágios, visando analisar o impacto deste parâmetro na qualidade final da solução obtida pelo *DP-GA*.

A Fig. 5.28 mostra os resultados obtidos para estes 9 casos. Como esperado, o custo monetário presente se tornou praticamente estável após 10 estágios (o custo exato de cada caso pode ser encontrado na Tab. 5.5). Também ocorre um decaimento considerável do custo quando se passa da abordagem estática para as abordagens dinâmicas, o que sugere que a mesma deve ser utilizada em todas as situações, mesmo que para um número pequeno de estágios.

A consistência dos resultados obtidos, que apresentam custos similares para diferentes números de estágios, parece confirmar a hipótese que o algoritmo encontrou a solução ótima para o problema. Por fim, deve-se ter em conta que um algoritmo

de programação dinâmica exato não seria capaz de resolver este problema em tempo computacional razoável.

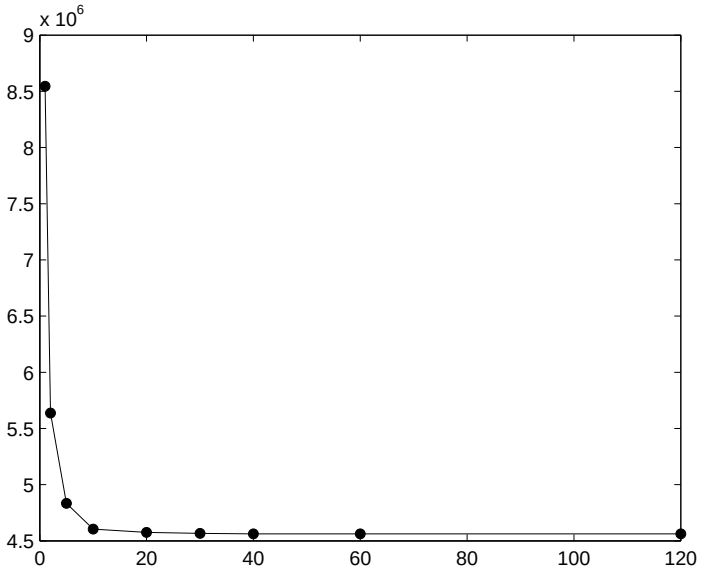


Figura 5.28: Sistema de 100 nós - Efeitos da discretização do tempo de projeto

Tabela 5.5: Sistema de 100 nós - Custo presente para os casos analisados

Estágios:	**01**	**02**	**05**	**10**	**20**
Custo ($):	8,545,547.18	5,637,242.35	4,834,213.87	4,604,416.15	4,575,654.04
Redução:	-	34.0%	14.3%	04.8%	00.6%

Estágios:	**30**	**40**	**60**	**120**
Custo ($):	4,566,171.55	4,561,447.70	4,561,447.70	4,561,447.70
Redução:	00.2%	00.1%	00.0%	00.0%

Capítulo 6

Conclusões e Propostas de Continuidade

6.1 Conclusões

6.1.1 Projeto de Redes

A uso de técnicas de otimização no projeto de redes é amplamente justificado pela importância destes sistemas e os altos custos geralmente demandados para instalação dos mesmos. Estas estruturas são utilizadas na modelagem de vários problemas práticos, como sistemas de energia, telefonia, abstecimento de água, redes de trânsito, etc. O desenvolvimento de novas técnicas de otimização, por outro lado, se justifica pela complexidade destes problemas, que são combinatórios e freqüentemente também não-lineares, o que limita a aplicação dos métodos hoje disponíveis.

Nos Caps. 1, 2 e 3 foram apresentadas as principais características do projeto de redes, incluindo a formulação geral destes problemas, representação das variáveis, problemas clássicos e dificuldades associadas. Nos Caps. 2 e 3 foram discutidos algoritmos de otimização aplicados direta ou indiretamente no projeto de redes. Os principais vantagens e desvantagens destes métodos são discutidas nos referidos capítulos, incluindo suas limitações no tratamento de redes.

6.1.2 Algoritmos Evolucionários Baseados em Inspirações Contínuas

Os algoritmos evolucionários tradicionais, usualmente propostos para otimização contínua, apresentam sérias deficiências no tratamento de redes, o que implica na necessidade de adaptações dos algoritmos. A adaptação dos operadores se mostra como uma alternativa plausível, potencialmente capaz de contornar estas limitações.

No Cap. 4 foi apresentado um conjunto de operadores baseados em extensões de conceitos contínuos. Estes operadores foram utilizados na construção de dois algoritmos evolucionários: um algoritmo genético, chamado de *GANet*, e um algoritmo de seleção clonal, chamado de *ClonalNet*. A generalidade destes algoritmos é garantida pelos operadores que servem de suporte para construção dos mesmos: como os operadores são baseados em conceitos abstratos gerais, sua aplicação não fica restrita a nenhum problema de redes específico.

Estes algoritmos apresentaram bons resultados em dois problemas clássicos reconhecidamente difíceis, e em duas aplicações no projeto de sistemas de distribuição de energia: um sistema teste e o projeto destes sistemas considerando incertezas na evolução de carga. Cada um dos algoritmos apresenta características favoráveis a contextos específicos: o *GANet* apresentou uma convergência rápida para o ótimo dos problemas, enquanto que o *ClonalNet* apresentou um mapeamento mais abrangente de soluções sub-ótimas, mesmo tendo mostrado convergência mais lenta. Isso faz com que nenhuma das ferramentas suplante a outra, sendo portanto ambas as alternativas viáveis para o projeto de redes.

6.1.3 Aplicações no Projeto de Sistemas de Distribuição de Energia Elétrica

O desenvolvimento de operadores baseados em características específicas de cada problema tratado também constitui uma solução para o problema da ineficiência apre-

sentada pelos algoritmos evolucionários básicos. Estes operadores tendem a melhorar consideravelmente a eficiência dos algoritmos, uma vez que levam em consideração aspectos relevantes do problema. No entanto, a aplicação destes operadores em contextos diferentes daqueles para os quais foram originalmente desenvolvidos se torna pouco plausível.

No Cap. 5 foram apresentados algoritmos específicos, orientados para três situações distintas, no contexto do projeto de redes de distribuição de energia:

- Projeto multi-objetivo, considerando custo e confiabilidade do sistema como critérios de projeto;
- Posicionamento de subestações associado ao projeto da rede, para instâncias mono e multi-objetivo;
- *Scheduling* da expansão do sistema.

Mais uma vez, os resultados obtidos para estas situações de projeto justificam o uso dos algoritmos propostos. Em todos os casos os resultados obtidos se mostraram melhores que os possíveis de serem obtidos com técnicas tradicionais de projeto, e o ganho econômico proporcionado pelos algoritmos foi significativo. O uso de técnicas deste tipo permite uma flexibilização das políticas de preço das concessionárias de energia, proporcionando uma maior competitividade em mercados abertos, ou uma maior margem de lucro em mercados fechados.

6.2 Propostas de Continuidade

As seguintes atividades são propostas como possíveis continuações desta tese de doutorado:

Aplicação dos Algoritmos Propostos na Solução de Outros Problemas de Redes

Propõe-se a aplicação dos algoritmos generalizados propostos, *GANet* e *ClonalNET*, a contextos diferentes dos tratados neste trabalho. Dentre os problemas previstos estão:

- Redimensionamento de redes de água;
- Projeto de redes de gás;
- Projeto de redes de comunicação, incluindo telecomunicações e redes de informática.

Redução da Complexidade Computacional do Algoritmo *GA-BFGS* Multi-objetivo

O alto número de execuções de algoritmos de otimização requerido pelo algoritmo *GA-BFGS* Multi-objetivo, proposto neste trabalho, faz do mesmo oneroso do ponto de vista computacional. Isto faz com que seja necessário um longo tempo de execução do algoritmo para obtenção do conjunto de Pareto completo para instâncias do problema. Propõe-se o estudo das especificidades do problema tratado, visando uma aceleração do método. Com base nessas especificidades, espera-se encontrar métodos potencialmente mais eficientes para o problema, além de inserir aproximações que não comprometam de forma determinante o resultado obtido.

Desenvolvimento de Métodos Capazes de Realizar o Planejamento "Completo" da Expansão de Sistemas de Distribuição de Energia Elétrica

Ao longo deste trabalho foram tratadas isoladamente três situações distintas de expansão do sistema de distribuição:

- Planejamento da expansão do sistema considerando incertezas na evolução de carga;
- Posicionamento de novas subestações associado ao projeto da topologia da rede;
- *Scheduling* da expansão do sistema de distribuição.

Estes casos representam condições relevantes de projeto, que podem ocorrer simultaneamente em alguns casos. Nestas situações, a mera execução seqüencial dos métodos propostos não garante a obtenção do melhor planejamento de expansão, uma vez que estes problemas apresentam uma forte interação entre si: diferentes posições da subestação e/ou diferentes contextos de incerteza implicam em diferentes topologias, que levam a diferentes *schedulings*. Espera-se propor métodos capazes de realizar este planejamento completo em casos que múltiplas situações de projeto devem ser consideradas.

6.3 Produção Bibliográfica Durante o Doutoramento

Periódicos

- E. G. Carrano, R. H. C. Takahashi, E. P. Cardoso, R. R. Saldanha & O. M. Neto. Optimal substation location and energy distribution network design using a hybrid GA-BFGS algorithm, *IEE Proceedings on Generation, Transmission and Distribution* **152**:919-926, 2005.

- E. G. Carrano, L. A. E. Soares, R. H. C. Takahashi, R. R. Saldanha & O. M. Neto. Electric distribution multiobjective network design usign a problem-specific genetic algorithm, *IEEE Transactions on Power Delivery* **21**:995-1005, 2006.

- E. G. Carrano, F. G. Guimaraes, R. H. C. Takahashi, O. M. Neto & F. Campelo. Electric distribution network expansion under load-evolution uncertainty using an immune system inspired algorithm, *IEEE Transactions on Power Systems* **22**:851-861, 2007.

- E. G. Carrano, R. H. C. Takahashi, C. M. Fonseca & O. M. Neto. Non-linear network topology optimization - An embedding vector space approach, *IEEE Transactions on Evolutionary Computation.* **Artigo submetido.**

- E. G. Carrano, R. T. N. Cardoso, R. H. C. Takahashi, C. M. Fonseca & Oriane M. Neto Power distribution network expansion scheduling using the dynamic programming genetic algorithm (DP-GA), *IEE Proceedings on Generation, Transmission and Distribution.* **Artigo submetido.**

Conferências

- E. G. Carrano, O. M. Neto & R. H. C. Takahashi. Algoritmos genéticos aplicados ao projeto e planejamento de expansão de redes de distribuição de energia elétrica, *Congresso Brasileiro de Automática*, Gramado, Brasil, 2005.

- V. L. Silva, A. R. da Cruz, E. G. Carrano & R. H. C. Takahashi. On nonlinear fitness functions for ranking-based selection, *IEEE World Congress on Computational Intelligence*, Vancouver, Canadá, 2006.

- E. G. Carrano, C. M. Fonseca & R. H. C. Takahashi. Power distribution network expansion scheduling using genetic algorithms, *Global Educational Technology Symposium*, Faro, Portugal, 2006. **Resumo expandido.**

- E. G. Carrano, R. H. C. Takahashi, C. M. Fonseca & O. M. Neto. Bi-objective combined facility location and network design, *Lecture Notes in Computer Science, Proceedings of EMO'2007* **4403**:486-500, Sendai, Japão, 2007.

- E. G. Carrano, C. M. Fonseca, R. H. C. Takahashi, L. C. A. Pimenta & O. M. Neto. A preliminary comparison of tree encoding schemes for evolutionary algorithms, *IEEE International Conference on System, Man and Cybernetics*, Montreal, Canadá, 2007. **Artigo aceito para publicação.**

Referências Bibliográficas

Abuali, F. N., Schoenefeld, D. A., and Wainwright, R. L. (1994). Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees. In *Proc. ACM Symposium on Applied Computing*, pages 242–246, Phoenix, USA.

Aguirre, H. E., Tanaka, K., Sugimura, T., and Oshita, S. (2001). Halftone image generation with improved multiobjective genetic algorithm. In *Proc. International Conference on Evolutionary Multi-Criterion Optimization*, pages 501–515, Zurich, Switzerland.

Aherne, F. J., Thacker, N. A., and Rockett, P. I. (1997). Automatic parameter selection for object recognition using a parallel multiobjective genetic algorithm. In *Proc. International Conference on Computer Analysis of Images and Patterns*, pages 559–566, Kiel, Germany.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1st edition.

Anderson, M. B. (1996). Using pareto genetic algorithms for preliminary subsonic wing design. In *Proc. AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium*, Bellevue, Washington.

Anderson, M. B., Burkhalter, J. E., and Jenkins, R. M. (2000). Missile aerodynamic shape optimization using genetic algorithms. *Journal of Spacecraft and Rockets*, 37:663–669.

Anton, H. and Rorres, C. (2000). *Elementary Linear Algebra - Applications Version*. John Willey & Sons, New York, 8th edition.

Arora, S., Lund, C., Montwani, R., Sundan, M., and Szegedy, M. (1998). Proof verification and the hardness of approximation problems. Technical report, University Trier, Trier, Germany. Colloquium on Computational Complexity Report TR98-008.

Assad, A. and Xu, W. (1992). The quadratic minimum spanning tree problem. *Naval Research Logistics*, 39:399–417.

Awadh, B., Sepehri, N., and Hawaleshka, O. (1995). A computer-aided process planning model based on genetic algorithms. *Computer & Operational Research*, 22:651–652.

Baker, B. M. and Ayechew, M. A. (2003). A genetic algorithm for vehicle routing problem. *Computer & Operational Research*, 30:787–800.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proc. International Conference on Genetic Algorithms*, pages 14–21, Massachusetts, United States.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. (1991). *Linear Programming and Network Flows*. Wiley, 2nd edition.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (1993). *Nonlinear Programming - Theory and Algorithms*. John Willey & Sons.

Bell, J. E. and McMullen, P. R. (2004). Ant colony optimization techiniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18:41–48.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

Billera, L. J., Holmes, S. P., and Vogtmann, K. (2001). Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27:733–767.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. MacMillanillan.

Campelo, F., Guimaraes, F. G., Igarashi, H., and Ramirez, J. A. (2005). A clonal selection algorithm for optimization in electromagnetics. *IEEE Transactions on Magnetics*, 41:1736–1739.

Campelo, F., Guimaraes, F. G., Saldanha, R. R., Igarashi, H., Noguchi, S., Lowther, D. A., and Ramírez, J. A. (2004). A novel multiobjective immune algorithm using nondominated sorting. In *Proc. International IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, Graz, Austria.

Cardon, A., Galinho, T., and Vacher, J. (1999). Using genetic algorithm in job-shop scheduling problem to constraints negociators' agents. In *Proc. Evolutionary Algorithms in Engineering and Computer Science*, pages 20–27, Jyväskylä, Finland.

Carrano, E. G., Cardoso, R. T. N., Takahashi, R. H. C., Fonseca, C. M., and Neto, O. M. (2007a). Power distribution network expansion scheduling using the dynamic programming genetic algorithm (DP-GA). *IEE Proceedings on Generation, Transmission and Distribution*. Paper submitted.

Carrano, E. G., Fonseca, C. M., Takahashi, R. H. C., Pimenta, L. C. A., and Neto, O. M. (2007b). A preliminary comparison of tree encoding schemes for evolutionary algorithms. In *Proc. IEEE International Conference on System, Man and Cybernetics*, Vancouver, Canada.

Carrano, E. G., Guimaraes, F. G., Takahashi, R. H. C., Neto, O. M., and Campelo, F. (2007c). Electric distribution network expansion under load-evolution uncertainty using an immune system inspired algorithm. *IEEE Transactions on Power Systems*, 22:851–861.

Carrano, E. G., Guimaraes, F. G., Takahashi, R. H. C., Neto, O. M., and Campelo, F. (2007d). Electric distribution network expansion under load evolution uncertainty using immune system inspired algorithms: simulation data. Technical report, Universidade Federal de Minas Gerais. Available in: `http://www.mat.ufmg.br/~taka/techrep/aisnet01.pdf`.

Carrano, E. G., Neto, O. M., and Takahashi, R. H. C. (2004). Algoritmos genéticos aplicados ao projeto e planejamento de expansão de redes de distribuição de energia elétrica. In *Proc. Congresso Brasileiro de Automática*, Gramado, Brazil.

Carrano, E. G., Soares, L. A. E., Takahashi, R. H. C., Saldanha, R. R., and Neto, O. M. (2006a). Electric distribution multiobjective network design usign a problem-specific genetic algorithm: simulation data. Technical report, Universidade Federal de Minas Gerais, Dept. de Matemática. Available in: `http://www.mat.ufmg.br/~taka/techrep/agnet01.pdf`.

Carrano, E. G., Soares, L. A. E., Takahashi, R. H. C., Saldanha, R. R., and Neto, O. M. (2006b). Electric distribution multiobjective network design using a problem-specific genetic algorithm. *IEEE Transactions on Power Delivery*, 21:995–1005.

Carrano, E. G., Takahashi, R. H. C., Cardoso, E. P., Saldanha, R. R., and Neto, O. M. (2005). Optimal substation location and energy distribution network design using a hybrid GA-BFGS algorithm. *IEE Proceedings on Generation, Transmission and Distribution*, 152:919–926.

Carrano, E. G., Takahashi, R. H. C., Fonseca, C. M., and Neto, O. M. (2007e). Bi-objective combined facility location and network design. *Lecture Notes in Computer Science*, 4403:486–500.

Carrano, E. G., Takahashi, R. H. C., Fonseca, C. M., and Neto, O. M. (2007f). Non-linear network topology optimization - An embedding vector space approach. *IEEE Transactions on Evolutionary Computation*. Paper submitted.

Carvalho, P. M. S., Ferreira, L. A. F. M., Lobo, F. G., and Barruncho, L. M. F. (1998). Optimal distribution network expansion planning under uncertainty by evolutionary

decision convergence. *International Jounal of Electric Power and Energy Systems*, 20(2):125–129.

Cayley, A. (1889). A theorem on trees. *Quart. Journal of Mathematics*, 23:376–378.

Chakroborty, P. (2003). Genetic algorithms for optimal urban transit network design. *Computer-Aided Civil and Infrastructure Engineering*, 18:184–200.

Chankong, V. and Haimes, Y. Y. (1983). *Multiobjective Decision Making: Theory and Methodology*. North-Holland Elsevier.

Ching-Tzong, S. and Guor-Rurng, L. (2002). Reliability design of distribution systems using modified genetic algorithms. *Electrical Power and System Research*, 60:201–206.

Christodoulakis, M., Iliopoulos, C. S., Park, K., and Sim, J. S. (2005). Implementing approximate regularities. *Mathematical and Computer Modelling*, 42:855–866.

Chung, T. S., Li, K. K., Chen, G. J., Xie, J. D., and Tang, G. D. (2003). Multi-objective transmission network system planning by hybrid GA approach with fuzzy decision analysis. *International Jounal of Electric Power and Energy Systems*, 25:187–192.

Coello, C. A. C. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32:109–143.

Cormem, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, 2nd edition.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proc. Genetic and Evolutionary Computation Conference*, pages 283–290, San Francisco, USA.

Corne, D. W., Knowles, J. D., and Oates, M. J. (2000). The pareto envelope-based selection algorithm for multiobjective optimization. In *Proc. Parallel Problem Solving from Nature Conference*, pages 839–848, Paris, France.

Cossi, A., Romero, R., and Mantovani, J. (2005). Planning of secundary distribution circuits through evolutionary algorithms. *IEEE Transactions on Power Delivery*, 20:205–213.

Ćurčić, S., Strbac, G., and Zhang, X. P. (2001). Effect of losses in design of distribution circuits. *IEE Proceedings on Generation, Transmission and Distribution*, 148:343–349.

Dasgupta, D. (1999). *Artificial Immune Systems and their Applications*. Springer-Verlag, Berlin, Germany.

Dasgupta, D. and Gonzalez, F. (2002). An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, 6:281–291. Special Issue on Artificial Immune Systems.

Dasgupta, D., Ji, Z., and Gonzalez, F. (2003). Artificial immune system (AIS) research in the last five years. In *Proc. IEEE International Congress on Evolutionary Computation*, pages 123–130, Canberra, Australia.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry Algorithms and Applications*. Springer-Verlag, 2nd edition.

de Castro, L. N. and Timis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, Berlin, Germany.

de Castro, L. N. and Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6:239–251.

Deb, K., Pratap, A., Agrawal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271.

Dorigo, M. (1992). *Optimization, learning and aatural algorithms*. PhD thesis, Politecnico di Milano. in Italian.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on System, Man and Cybernetics*, 26:1–13.

Duan, G. and Yu, Y. (2002). Problem-specific genetic algorithms for power transmission system planning. *Electrical Power and System Research*, 61:41–50.

Ehrgott, M. (2000). *Multicriteria Optimization*. Springer.

Eppstein, D. (1996). Spanning trees and spanners. Technical Report 96-16, University of California, Dept. of Information and Computer Science. Available in: `http://www.ics.uci.edu/~eppstein/pubs/Epp-TR-96-16.pdf`.

Even, S. (1989). *Graph Algorithms*. Computer Science Press, Rockville, Maryland.

Farias, D. P. and Roy, B. V. (2003). The linear programming approach to approximate dynamic programming. *Operations Research*, 51:850–865.

Feng, C. M. and Lin, J. J. (1999). Using a genetic algorithm to generate alternative sketch maps for urban planning. *Computer Environment and Urban Systems*, 23:91–108.

Ferentinos, K. P., Arvanitis, K. G., and Sigrimos, N. (2002). Heuristic optimization methods for motion planning of autonomous agricultural vehicles. *Journal of Global Optimization*, 23:155–170.

Fonseca, C. M. (1995). *Multiobjective genetic algorithms with applications to control engineering problems*. PhD thesis, University of Sheffield, Sheffield, UK.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proc. International Conference on Genetic Algorithms*, San Mateo, USA.

Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 9:1–16.

Forrest, S., Javornik, B., Smith, R. E., and Perelson, A. S. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1:191–211.

Galton, A. (1999). Two topologies for discrete space. In *Proc. Leeds Spatial Reasoning Workshop*, Leeds, UK.

Galton, A. (2000). Continuous motion in discrete spaces. In *Proc. International Conference on Principles of Knowledge Representation and Reasoning*, San Francisco, USA.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, USA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Gómez, J. F., Khodr, H. M., De Oliveira, P. M., Ocque, L., Yusta, J. M., Villasana, R., and Urdaneta, A. J. (2004). Ant colony system algorithm for the planning of primary distribution circuits. *IEEE Transactions on Power Systems*, 19:996–1004.

Grötschel, M. and Holland, O. (1991). Solution of large scale travelling salesman problems. *Mathematical Programming*, 52:141–202.

Güneş, M. and Bouazizi, U. S. I. (2002). ARA - The ant colony based routing algotihm for MANETs. In *Proc. International Conference on Parallel Processing Workshops*, pages 79–85, Vancouver, Canada.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technology Journal*, 26:147–160.

Hamming, R. W. (1980). *Coding and Information Theory*. Prentice Hall.

Hancock, P. J. B. (1994). An empirical comparison of selection methods in evolutionary algorithms. In *Proc. AISB Workshop on Evolutionary Computation*, pages 80–94.

Hartmann, J. W., Coverstone-Carroll, V. L., and Williams, S. N. (1998). Optimal interplanetary spacecraft trajectories via a pareto genetic algorithm. *The Journal of the Astronautical Sciences*, 46:267–282.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Horn, J. and Nafpliotis, N. (1993). Multiobjective optimization using the niched-pareto genetic algorithm. Technical report, University of Illinois, Genetic Algorithms Laboratory, Illinois, USA.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proc. IEEE International Congress on Evolutionary Computation*, Orlando, USA.

Hu, T. C. (1974). Optimum communication spanning trees. *SIAM Journal of Computing*, 3:188–195.

Huang, S. J. (2002). Application of immune-based optimization method for fault-section estimation in a distribution system. *IEEE Transactions on Power Delivery*, 17:779–784.

Johnson, J. M. and Ramat-Semii, Y. (1997). Genetic algorithms in engineering eletromagnetics. *IEEE Antennas Propagation Magazine*, 39:7–25.

Keko, H., Skok, M., and Skrlec, D. (2003). Artificial immune systems in solving routing problems. In *Proc. IEEE EUROCON: Computer as a tool*, pages 62–66, Ljubljana, Slovenia.

Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. In *Proc. IEEE International Conference on Evolutionary Computation*, Indianapolis, USA.

Knowles, J. and Corne, D. (1999). The pareto archived evoution strategy: a new baseline for pareto multiobjective optimisation. In *Proc. IEEE International Congress on Evolutionary Computation*, pages 98–105, Washington, USA.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling sallesman problem. *Proceedings of American Mathematics Society*, 7:48–50.

Lafage, C. and Lang, J. (2005). Propositional distances and compact preference representation. *European Journal of Operational Research*, 160:741–761.

Lakshminarasimman, L. and Subramanian, S. (2006). Short-term scheduling of hydrothermal power system with cascaded reservoirs by using modified differential evolution. *IEE Proceedings on Generation, Transmission and Distribution*, 153:693–700.

Levitin, G. and Lisnianski, A. (1999). Optimal multistage modernization of power system subject to reliability and capacity requirements. *Electrical Power and System Research*, 50:183–190.

Lima, E. L. (1995). *Curso de Análise*. Projeto Euclides - IMPA, Rio de Janeiro.

Lin, L. and Gen, M. (2006). Node-based genetic algorithm for communication spanning tree problem. *IEICE Transactions on Communications*, 89-B:1091–1098.

Lincoln, B. and Rantzer, A. (2006). Relaxing dynamic programming. *IEEE Transactions on Automatic Control*, 51:1249–1260.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods*. John Willey & Sons.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley.

Luh, G. C., Chueh, C. H., and Liu, W. W. (2003). MOIA: Multi-objective immune algorithm. *Engineering Optimzation*, 35:143–164.

Luh, G. C. and Liu, W. W. (2004). Reactive immune network based mobile robot navigation. *Lecture Notes in Computer Science*, 3239:119–132.

Mak, T. S. T. and Lam, K. P. (2005). Equivalence-set genes partitioning using an evolucionary-dp approach. *IEEE Transactions on Nanobiosciense*, 4:295–300.

Man, K. F., Tang, K. S., and Kwong, S. (1996). Genetic algorithms: Concepts and applications. *IEEE Transactions on Industrial Electronics*, 43:519–534.

Michelan, R. and Von Zuben, F. J. (2002). Decentralized control system for autonomous navigation based on an evolved artificial immune network. In *Proc. IEEE International Congress on Evolutionary Computation*, pages 1021–1026, Hawaii, USA.

Miranda, V., Ranito, J. V., and Proença, L. M. (1994). Genetic algorithms in optimal multistage distribution network planning. *IEEE Transactions on Power Systems*, 9:1927–1933.

Monteiro, C., Ramirez-Rosado, I. J., Miranda, V., Zorzano-Santamaria, P. J., Garcia-Garrido, E., and Fernandez-Jimenez, L. A. (2005). Gis spatial analysis applied to electric line routing optimization. *IEEE Transactions on Power Delivery*, 20:934–942.

Morsi, D. M., Abbasy, N. H., and Abul Ella, M. S. (1994). A hybrid expert system assisting decision making for distribution system load forecasting. In *Proc. Mediterranean Electrotechnical Conference*, pages 893–896, Antalya.

Narsingh, D. (1984). *Graph Theory with Applications to Engineering and Computer Science*. Prentice Hall, New Delhi, India.

Palmer, C. C. and Kershenbaum, A. (1994). Representing trees in genetic algorithms. In *Proc. IEEE International Conference on Evolutionary Computation*, pages 379–384, Bristol, USA.

Parada, V., Ferland, J. A., and Daniels, M. A. K. (2004). Optimization of electrical distribution feeders using simulated annealing. *IEEE Transactions on Power Delivery*, 19:1135–1141.

Pemmaraju, S. and Skiena, S. (2003). *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press.

Pierre, S. (1993). Application of artificial intelligence techniques to computer network design. *Engineering Applications in Artificial Intelligence*, 6:465–472.

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401.

Prüfer, H. (1918). Neuer Beweis eines Satzes über Permutationen. *Arch. Math. Phys.*, 27:742–744.

Raidl, G. R. and Julstrom, B. A. (2003). Edge-sets: An effective evolutionary encoding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7:225–239.

Ramírez-Rosado, I. and Bernal-Agustín, J. (1998). Genetic algorithms applied to the design of large power distribution systems. *IEEE Transactions on Power Systems*, 13:696–702.

Ramos, R. M., Saldanha, R. R., Takahashi, R. H. C., and Moreira, F. J. S. (2003). The real-biased multiobjective genetic algorithm and its application to the design of wire antennas. *IEEE Transactions on Magnetics*, 39:1329–1332.

ReVelle, C. S. and Eiselt, H. A. (2005). Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165:1–19.

Rothlauf, F. (2005). *Representations for Genetic and Evolutionary Algorithms*. Springer, Berlim, 2nd edition.

Routhlauf, F., Goldberg, D., and Heinzl, A. (2002). Network random key - A tree network representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10:75–97.

Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. International Conference on Genetic Algorithms*, pages 99–100, New Jersey, USA.

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University, Nashville, USA.

Shamos, M. I. and Hoey, D. (1975). Closest-point problems. In *Proc. IEEE Symposium Foundations of Computer Science*, pages 151–162.

Silva, V. L., da Cruz, A. R., Carrano, E. G., and Takahashi, R. H. C. (2006). On nonlinear fitness functions for ranking-based selection. In *To appear in IEEE World Congress on Computational Intelligence*, Vancouver, Canada.

Smith, D. K. and Walters, G. A. (2000). An evolutionary approach for finding optimal trees in undirected networks. *European Journal of Operational Research*, 120:593–602.

Soak, S., Corne, D. W., and Ahn, B. (2006). The edge-window-decoder representation for tree-based problems. *IEEE Transactions on Evolutionary Computation*, 10:124–144.

Soares, G. L. (1997). Algoritmos genéticos: estudo, novas técnicas e aplicações. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil.

Soares, L. A. E. (2001). Uma abordagem multi-critério no planejamento ótimo de sistemas de distribuição de energia. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil.

Soares, S., Neto, A. M., Cardoso, A., Viana, F., Javaroni, M., Pereira, M., and Brito, W. (1999). Metododologia para determinação, análise e otimização de perdas técnicas em sistemas de distribuição. In *Proc. Encontro Luso-Afro-Brasileiro de Planejamento e Exploração de Redes de Energia*, Rio de Janeiro, Brazil.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248.

Storn, R. and Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In *Proc. IEEE International Conference on Evolutionary Computation*, Washington, USA.

Takahashi, R. H. C. (2004). Otimização Escalar e Vetorial - Notas de Aula. Technical report, Universidade Federal de Minas Gerais, Dept. de Matemática. Available in: `http://www.mat.ufmg.br/~taka/otev04.pdf`.

Takahashi, R. H. C., Palhares, R. M., Dutra, D. A., and Gonçalves, L. P. S. (2004). Estimation of pareto sets in the mixed h_2/h_∞ control problem. *International Journal of System Science*, 35:55–67.

Takahashi, R. H. C., Vasconcelos, J. A., Ramirez, J. A., and Krahenbuhl, L. (2003). A multiobjective methodology for evaluating genetic operators. *IEEE Transactions on Magnetics*, 39:1321–1324.

Tang, Y. (1996). Power distribution system planning with reliability modeling and optimization. *IEEE Transactions on Power Systems*, 11:181–189.

Tanomaru, J. (1995). Motivação, fundamentos e aplicações de algoritmos genéticos. In *Proc. Congresso Brasileiro de Redes Neurais*, Curitiba, Brazil.

Tse, S., Liang, Y., Leung, K., Lee, K., and Mok, T. S. (2007). A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE Transactions on System, Man and Cybernetics - Part B*, 37:84–91.

Vanderbei, R. J. (2001). *Linear Programming: Foundations and Extensions*. Springer, 2nd edition.

Vaziri, M., Tomsovic, K., and Bose, A. (2004). A directed graph formulation for the multistage distribution expansion problem. *IEEE Transactions on Power Delivery*, 19:1335–1341.

Willis, H. L. and Northcode-Green, J. E. D. (1983). Spatial electric load forecasting: A tutorial review. *Proceedings of IEEE*, 71:232–253.

Willis, H. L., Tram, H., Engel, M. V., and Finley, L. (1996). Selecting and applying distribution optimization methods. *IEEE Computer Applications in Power Magazine*, 9:12–17.

Wilson, R. J. (1996). *Introduction to Graph Theory*. Prentice Hall, 4th edition.

Yehia, M. A., Matar, E. E., Hobeila, N. Y., and Avedikian, A. D. (2002). A heuristic algorithm for electric distribution networks optimal feeder configuration using geographic information system. *IEEE Transactions on Power Systems*, 17:1232–1237.

Zecchin, A. C., Maier, H. R., Simpson, A. R., Roberts, A. J., Berrisford, M. J., and Leonard, M. (2003). Max-Min ant system applied to water distibution system optimization. In *Proc. Congress on Modelling and Simulation*, pages 795–800, Queensland, Australia.

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. *Lecture Notes in Computer Science*, 3242:832–842.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. Technical report, Swiss Federal Institute of Technology (ETH), Computer Engineering and Network Laboratory. Available in: `http://www.tik.ee.ethz.ch/sop/publicationListFiles/zlt2001a.pdf`.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the Strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271.

Apêndice A

Definições Relevantes

Neste apêndice são apresentadas algumas definições necessárias para o entendimento dos algoritmos de otimização apresentados. Estas definições foram extraídas de (Takahashi, 2004).

A.1 Conjuntos

Definição A.1 (Conjunto Aberto) *Seja $\mathcal{X}$ um conjunto contido em um espaço normado. O conjunto $\mathcal{X}$ é dito ser aberto se:*

$$x_0 \in \mathcal{X} \Rightarrow \{\exists \epsilon | x \in \mathcal{X}\} \quad \forall \quad \|x - x_0\| < \epsilon \tag{A.1}$$

Definição A.2 (Complemento de Conjunto) *Seja $\mathcal{X} \subset \mathcal{Q}$ um conjunto contido em outro conjunto $\mathcal{Q}$. O conjunto $\mathcal{Y}$ é o complemento de $\mathcal{X}$ em relação a $\mathcal{Q}$ se:*

$$\mathcal{X} \cup \mathcal{Y} = \mathcal{Q} \tag{A.2}$$

$$\mathcal{X} \cap \mathcal{Y} = \emptyset \tag{A.3}$$

Definição A.3 (Conjunto Fechado) *Seja $\mathcal{X}$ um conjunto contido em um espaço topológico. O conjunto $\mathcal{X}$ é dito fechado se seu complemento em relação ao espaço for aberto.*

Definição A.4 (Conjunto Compacto) *Seja o conjunto $\mathcal{Q} \subset \mathbb{R}^n$. Esse conjunto é dito compacto se for fechado e se dados dois pontos $x_1, x_2 \in \mathcal{Q}$, eles se encontram a uma distância finita um do outro:*

$$\|x_1 - x_2\| = \delta < \infty \tag{A.4}$$

Definição A.5 (Conjunto Conexo) *Seja o conjunto $\mathcal{Q} \subset \mathbb{R}^n$. Esse conjunto é conexo se, dados quaisquer dois pontos $x_1, x_2 \in \mathcal{Q}$, existe uma curva finita q tal que $x_1 \in q$, $x_2 \in q$, e $q \subset \mathcal{Q}$.*

Definição A.6 (Conjunto Convexo) *Seja um conjunto $\mathcal{X}$, contido em um espaço vetorial, e seja a função $x(x_1, x_2, \alpha)$ definida por:*

$$x(x_1, x_2, \alpha) = \alpha \cdot x_1 + (1 - \alpha) \cdot x_2 \tag{A.5}$$

para x_1 e x_2 elementos do espaço vetorial e α um escalar, onde $0 \leq \alpha \leq 1$. Se se verificar:

$$\{x_1, x_2 \in \mathcal{X}; 0 \leq \alpha \leq 1\} \Rightarrow x(x_1, x_2, \alpha) \in \mathcal{X} \tag{A.6}$$

então o conjunto $\mathcal{X}$ é dito ser convexo.

Definição A.7 (Casca Convexa) *A casca convexa $\mathcal{C}(S)$ de um conjunto S é definida como o menor conjunto convexo que contém S, ou seja:*

i. $\mathcal{C}(S)$ *é um conjunto convexo*

ii. $x \in S \Rightarrow x \in \mathcal{C}(s)$

iii. $\nexists D$ *convexo tal que:*

$$\begin{aligned} D &\supset S \\ D &\subset \mathcal{C}(S) \\ D &\neq \mathcal{C}(s) \end{aligned}$$

A.2 Superfícies de Nível

Definição A.8 (Superfície de Nível) *Seja* $f(\cdot) : C \subset \mathbb{R}^n \mapsto \mathbb{R}$. *A superfície de nível* $S(f, \alpha)$, *associada ao nível* α, *é definida como:*

$$S(f, \alpha) = \{x \in C | f(x) = \alpha\} \tag{A.7}$$

Definição A.9 (Região Sub-Nível) *Seja* $f(\cdot) : C \subset \mathbb{R}^n \mapsto \mathbb{R}$. *A região sub-nível* $R(f, \alpha)$, *associada ao nível* α, *é definida como:*

$$R(f, \alpha) = \{x \in C | f(x) \leq \alpha\} \tag{A.8}$$

Apêndice B

Operadores Desenvolvidos

B.1 Operadores - *KruskalGA*

B.1.1 Cruzamento

KruskalCrossover

1. Selecionar duas árvores da população, p_1 e p_2;
2. Encontrar G_p que é a união das duas árvores:
   - $G_p = p_1 \bigcup p_2$;
3. Gerar um conjunto aleatório de pesos para as arestas de G_p, $W_1^{G_p}$;
4. Encontrar s_1 utilizando o algoritmo de Kruskal sobre o grafo G_p para o conjunto de pesos $W_1^{G_p}$;
5. Gerar outro conjunto aleatório de pesos para as arestas de G_p, $W_2^{G_p}$;
6. Encontrar s_2 utilizando o algoritmo de Kruskal sobre o grafo G_p para o conjunto de pesos $W_2^{G_p}$.

B.1.2 Mutação

KruskalMutation

1. Selecionar um árvore da população, p_1;
2. Encontrar o conjunto complementar de arestas de p_1, $\bar{p_1}$;
3. Gerar um conjunto aleatório de pesos para as arestas de p_1, $W_1^{p_1}$;
4. Escolher aleatoriamente uma das arestas de $\bar{p_1}$, $e^{\bar{p_1}}$;
5. Criar s_1 inserindo a aresta $e^{\bar{p_1}}$;
6. Completar a árvore s_1 utilizando o algoritmo de Kruskal sobre o grafo p_1 para o conjunto de pesos $W_1^{p_1}$.

B.2 Operadores - *GANet* e *ClonalNet*

B.2.1 Cruzamento

DirecCrOv

1. Selecionar duas redes da população, p_1 e p_2;
2. Gerar um fator de distância f_1 (entre 0 e 1);
3. Encontrar s_1 a partir de p_1, na direção de p_2, tal que:
   - $d_s(s_1, p_1) \approx f_1 \cdot d_s(p_1, p_2)$;
   - $d_s(s_1, p_2) \approx (1 - f_1) \cdot d_s(p_1, p_2)$;
4. Gerar um fator de distância f_2 (entre 0 e 1);
5. Encontrar s_2 a partir de p_2, na direção de p_1, tal que:
   - $d_s(s_2, p_1) \approx (1 - f_2) \cdot d_s(p_1, p_2)$;

- $d_s(s_2, p_2) \approx f_2 \cdot d_s(p_1, p_2)$.

GoldSecCrOv

1. Selecionar duas redes da população, p_1 e p_2;
2. Encontrar s_1 a partir de p_1, na direção de p_2, utilizando um algoritmo análogo a seção áurea, implementado para redes;
3. Gerar um fator de distância f_1 (entre 0 e 1);
4. Encontrar s_1 a partir de p_1, na direção de p_2, tal que:
   - $d_s(s_1, p_1) \approx f_1 \cdot d_s(p_1, p_2)$;
   - $d_s(s_1, p_2) \approx (1 - f_1) \cdot d_s(p_1, p_2)$;

B.2.2 Mutação

AnyDstMut

1. Selecionar uma rede da população, p_1;
2. Encontrar o raio de mutação:
   - $r_m = \text{rand} \cdot r_b$[1];
3. Encontrar s_1 a partir de p_1, tal que:
   - $d_s(s_1, p_1) \approx r_b$;

LocSeaMut

1. Selecionar uma rede da população, p_1;
2. Copiar p_1 para s_1;

[1] r_b é o raio de mutação base.

3. Para cada uma de k execuções:

   (a) Encontrar o raio de mutação:

   - $r_m = \text{rand} \cdot r_b$;

   (b) Encontrar s_t a partir de p_1, tal que:

   - $d_s(s_t, p_1) \approx r_b$;

   (c) Se s_t é melhor que s_1, copiar Copiar s_t para s_1.

B.3 Operadores - *NSGA-PS*

B.3.1 Cruzamento

SmartCrossover1

1. Selecionar duas redes da população, p_1 e p_2;
2. Remover os ramos de reserva de p_1 e p_2;
3. Identificar os ramos comuns nessas redes ($p_1 \cap p_2$);
4. Gerar dois grafos com os ramos restantes:
   - $t_1 = p_1 - (p_1 \cap p_2)$;
   - $t_2 = p_2 - (p_1 \cap p_2)$;
5. Criar as duas redes filhas:
   - $s_1 = t_1$;
   - $s_2 = t_2$;
6. Completar s_1 e s_2 utilizando os ramos de $p_1 \cap p_2$:
   - Completar s_1 utilizando os ramos de menor capacidade;

   - Completar s_2 utilizando os ramos de menor comprimento;

7. Inserir os ramos de reserva de p_2 em s_1 e os ramos de reserva de p_1 em s_2.

SmartCrossover2

1. Selecionar duas redes da população, p_1 e p_2;
2. Remover os ramos de reserva de p_1 e p_2;
3. Identificar os ramos comuns nessas redes $(p_1 \cap p_2)$;
4. Gerar dois grafos com os ramos restantes:
   - $t_1 = p_1 - (p_1 \cap p_2)$;
   - $t_2 = p_2 - (p_1 \cap p_2)$;
5. Criar as duas redes filhas:
   - $s_1 = t_1$;
   - $s_2 = t_2$;
6. Completar s_1 e s_2 utilizando os ramos de $p_1 \cap p_2$:
   - Completar s_1 utilizando os ramos de maior capacidade;
   - Completar s_2 utilizando os ramos de menor comprimento;
7. Inserir os ramos de reserva de p_2 em s_1 e os ramos de reserva de p_1 em s_2.

B.3.2 Mutação

SmartMutation1

1. Selecionar uma rede da população, p_1;
2. Copiar p_1 para s_1;

3. Encontrar $\bar{p_1}$;
4. Ordenar os ramos de $\bar{p_1}$ em ordem crescente de custos em um vetor V;
5. Inserir o primeiro ramo de V em s_1 e identificar os extremos do ramo, n_a e n_b;
6. Avaliar todos os ramos conectados a n_a e n_b em p_1, e o excluir o de maior comprimento em s_1;
7. Caso as restrições de factibilidade e radialidade não sejam obedecidas, desfazer as ações realizadas nos passos 5 e 6. Considerar o próximo ramo de V e repetir o processo.

SmartMutation2

1. Selecionar uma rede da população, p_1;
2. Copiar p_1 para s_1;
3. Encontrar $\bar{p_1}$;
4. Ordenar os ramos de p_1 em ordem decrescente de custos em um vetor V;
5. Excluir o primeiro ramo de V em s_1 e identificar os extremos do ramo, n_a e n_b;
6. Avaliar todos os ramos conectados a n_a e n_b em $\bar{p_1}$, e inserir o de menor comprimento em s_1;
7. Caso as restrições de factibilidade e radialidade não sejam obedecidas, desfazer as ações realizadas nos passos 5 e 6. Considerar o próximo ramo de V e repetir o processo.

SmartMutation3

1. Selecionar uma rede da população, p_1;

2. Copiar p_1 para s_1;

3. Caso exista mais de um ramo de reserva em s_1, excluir o ramo de reserva de maior comprimento.

SmartMutation4

1. Selecionar uma rede da população, p_1;

2. Copiar p_1 para s_1;

3. Caso algum nó de s_1 não possua alternativa de alimentação se um de seus ramos falhar, então criar um ramo de reserva para este nó.

SimpleMutation1

1. Selecionar uma rede da população, p_1;

2. Copiar p_1 para s_1;

3. Encontrar o ramo de maior impedância em s_1, r_a;

4. Substituir r_a por um condutor de menor impedância.

SimpleMutation2

1. Selecionar uma rede da população, p_1;

2. Copiar p_1 para s_1;

3. Encontrar o ramo de maior taxa de falha em s_1, r_a;

4. Substituir r_a por um condutor com menor taxa de falha.

SimpleMutation3

1. Selecionar uma rede da população, p_1;
2. Copiar p_1 para s_1;
3. Calcular o fluxo de potência de s_1, e substituir todos os condutores pelos de menor capacidade admissível.

B.3.3 Operadores Determinísticos

DetMutation1 - Custos

1. A cada k gerações, selecionar a rede com menor custo monetário da população, p_1;
2. Copiar p_1 para s_1;
3. Substituir cada ramo pelo ramo de índice ótimo para a topologia s_1;
4. Caso s_1 tenha menor custo monetário que p_1, então s_1 substitui p_1.

DetMutation2 - Confiabilidade

1. A cada k gerações, selecionar a rede com menor custo de falta da população, p_1;
2. Copiar p_1 para s_1;
3. Substituir cada ramo por um ramo com menor taxa de falha;
4. Caso s_1 tenha menor custo de falta que p_1, então s_1 substitui p_1.

B.4 Operadores - *DP-GA*

B.4.1 Operadores de Correção

Fix1

1. Para um indivíduo da população, p_1;
2. Verificar se p_1 obedece a restrição dinâmica (c_8);
3. Caso negativo, inserir ou remover ações em p_1 para que o mesmo cumpra com a restrições dinâmica.

Fix2

1. Para um indivíduo da população, p_1;
2. Verificar se, para cada estágio, todos os nós com carga estão ligados à rede;
3. Caso negativo, antecipar ações de p_1 para o instante de tempo adequado, fazendo com que não existam cargas ativas desconectadas do sistema em nenhum instante de tempo.

Fix3

1. Para um indivíduo da população, p_1;
2. Verificar se, para cada estágio, a rede é radial;
3. Caso negativo, atrasar ações de p_1 para o instante de tempo adequado, fazendo com que a rede ao fim de cada estágio seja radial.

B.4.2 Cruzamento

DPCrossover1

1. Selecionar dois indivíduos da população, p_1 e p_2;

2. Copiar p_1 para s_1 e p_2 para s_2;

3. Escolher um conjunto de k estágios aleatórios, $\mathcal{S}$;

4. Trocar as ações ocorridas nestes k estágios entre s_1 e s_2:

   - $s_1[\mathcal{S}; 1, \ldots, m] = p_2[\mathcal{S}; 1, \ldots, m]$;
   - $s_2[\mathcal{S}; 1, \ldots, m] = p_1[\mathcal{S}; 1, \ldots, m]$;

5. Corrigir s_1 e/ou s_2 utilizando o **Fix1**, caso seja necessário;

6. Corrigir s_1 e/ou s_2 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

DPCrossover2

1. Selecionar dois indivíduos da população, p_1 e p_2;

2. Copiar p_1 para s_1 e p_2 para s_2;

3. Escolher um conjunto de l conexões aleatórias, $\mathcal{C}$;

4. Trocar as ações ocorridas para estas l conexões entre s_1 e s_2:

   - $s_1[1, \ldots, N; \mathcal{C}] = p_2[1, \ldots, N; \mathcal{C}]$;
   - $s_2[1, \ldots, N; \mathcal{C}] = p_1[1, \ldots, N; \mathcal{C}]$;

5. Corrigir s_1 e/ou s_2 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

B.4.3 Mutação

DPMutation1

1. Selecionar um indivíduo da população, p_1;

2. Copiar p_1 para s_1;

3. Escolher uma conexão c aleatoriamente;

4. Escolher um estágio k_1 dentre os estágios em que:
   - $p_1[k_1, c] \neq 0$;
5. Escolher outro estágio k_2 aleatoriamente;
6. Alterar o momento da ação $p_1[k_1, c]$:
   - $s_1[k_2; c] = s_1[k_2; c] + p_1[k_1; c]$;
   - $s_1[k_1; c] = 0$;
7. Corrigir s_1 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

DPMutation2

1. Selecionar um indivíduo da população, p_1;
2. Copiar p_1 para s_1;
3. Escolher um estágio k_1 dentre os estágios em que:
   - $p_1[k_1, c] \neq 0$ para ao menos uma conexão $c = 1, \ldots, m$;
4. Escolher outro estágio k_2 aleatoriamente;
5. Mover as ações de k_1 para k_2 e vice-versa:
   - $s_1[k_1; 1, \ldots, m] = p_1[k_2; 1, \ldots, m]$;
   - $s_1[k_2; 1, \ldots, m] = p_1[k_1; 1, \ldots, m]$;
6. Corrigir s_1 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

DPMutation3

1. Selecionar um indivíduo da população, p_1;
2. Copiar p_1 para s_1;
3. Escolher uma conexão c dentre as conexões em que:
   - $\sum_{i=1}^{N} p_1(i,c) \geq 2$;
   - $|\{p_1[1,\ldots,N;c] > 0\}| \geq 2$;
4. Escolher um estágio k_1 dentre os estágios em que:
   - $p_1[k_1,c] \geq 2$;
5. Escolher um estágio k_2 aleatoriamente;
6. Atualizar s_1 tal que:
   - Gerar a entre 1 e $s_1[k_1;c]$;
   - $s_1[k_1;c] = s_1[k_1;c] - a$;
   - $s_1[k_2;c] = s_1[k_2;c] + a$;
7. Se $s_1[k_1;c] > 0$ então voltar ao passo 5;
8. Corrigir s_1 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

DPMutation4

1. Selecionar um indivíduo da população, p_1;
2. Copiar p_1 para s_1;
3. Escolher uma conexão c dentre as conexões em que:
   - $\sum_{i=1}^{N} P(i,c) \geq 2$;

- $|\{p_1(1,c), p_1(2,c), \ldots, p_1(N,c)\}| \geq 2$;

4. Escolher um estágio k_1 dentre os estágios em que:

   - $p_1(k_1,c) \geq 0$;

5. Escolher um estágio k_2 dentre os estágios em que:

   - $p_1(k_2,c) \geq 0$;

6. Escolher um estágio k_3 aleatoriamente;

7. Atualizar s_1 tal que:

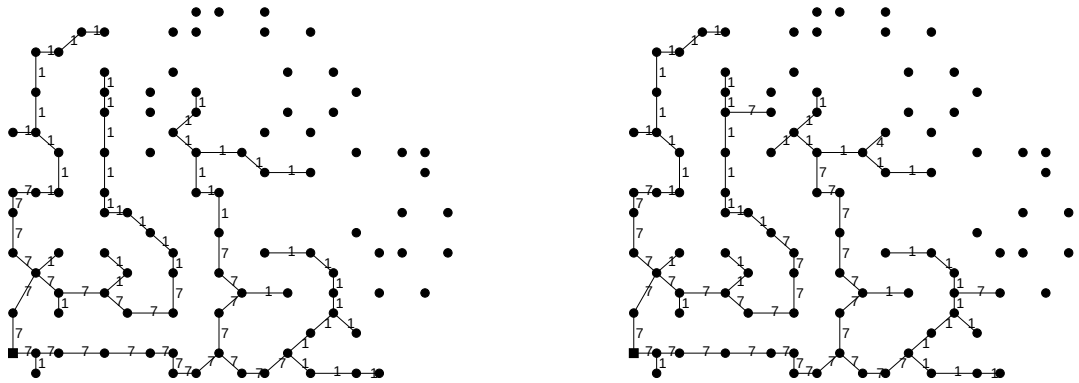
   - $s_1[k_1;c] = 0$;
   - $s_1[k_2;c] = 0$;
   - $s_1[k_3;c] = p_1[k_1;c] + p_1[k_2;c]$;

8. Corrigir s_1 utilizando o **Fix2** e o **Fix3**, caso seja necessário.

Apêndice C

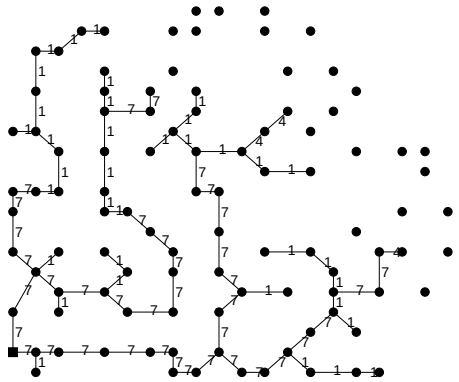
Scheduling da Expansão do Sistema de 100 Nós - Soluções Obtidas

C.1 *Scheduling* Obtido pelo *DP-GA*

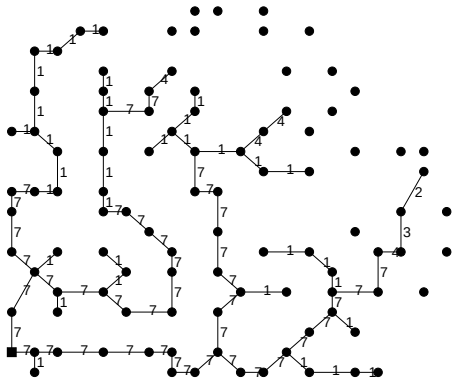


(a) Estágio 01

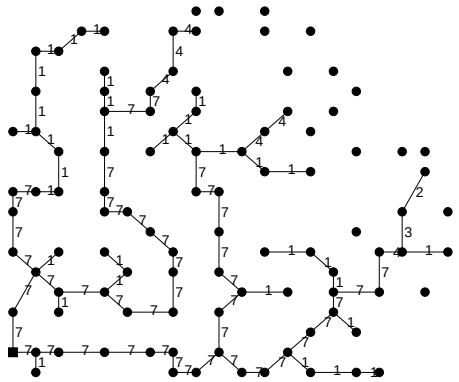
(b) Estágio 02



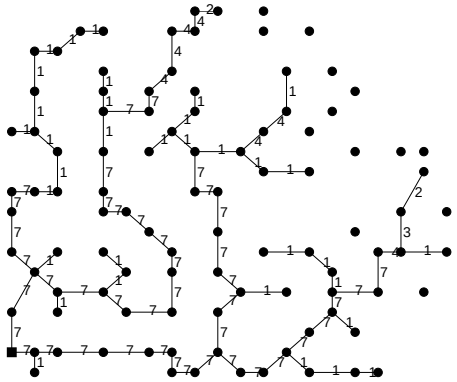
(c) Estágio 03



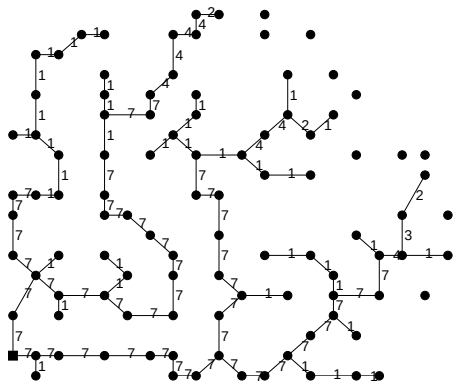
(d) Estágio 04



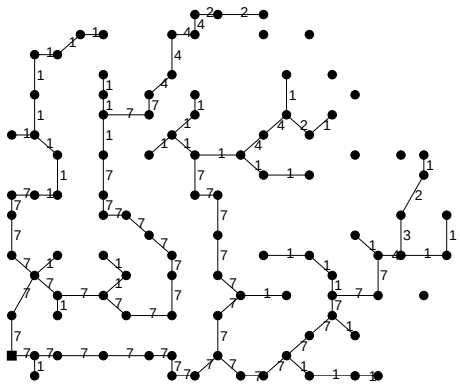
(e) Estágio 05



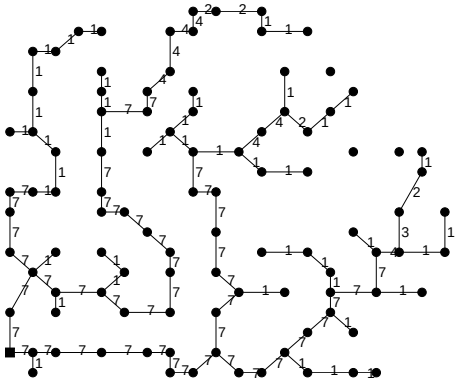
(f) Estágio 06



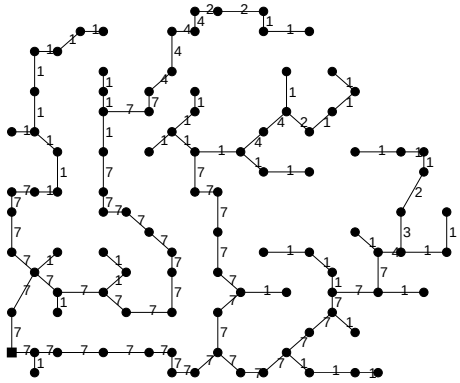
(g) Estágio 07



(h) Estágio 08



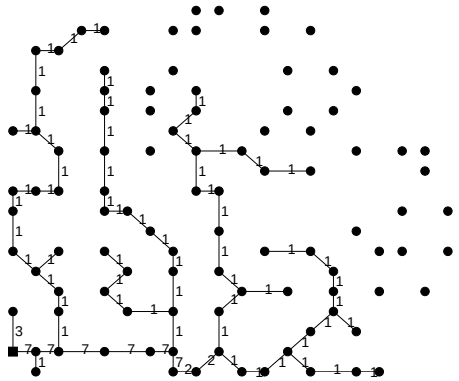
(i) Estágio 09



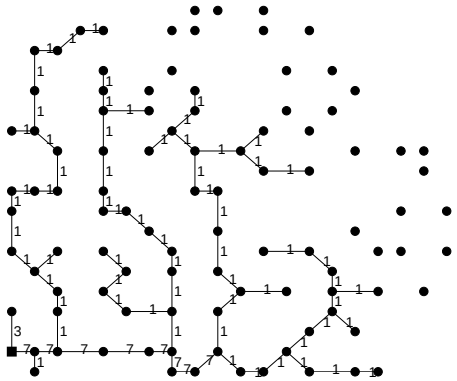
(j) Estágio 10

Figura C.1: *Scheduling* obtido pelo *DP-GA*

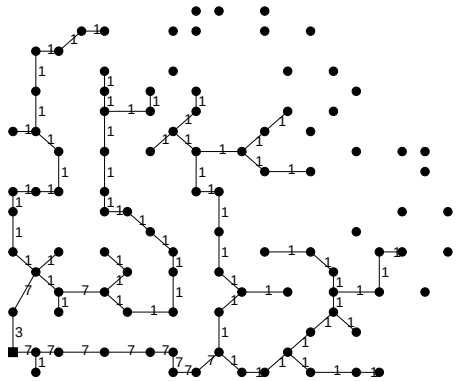
C.2 *Scheduling* Obtido pelo Algoritmo Genético Não-Dinâmico



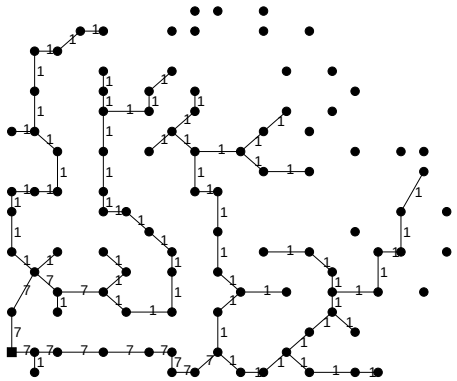
(a) Estágio 01



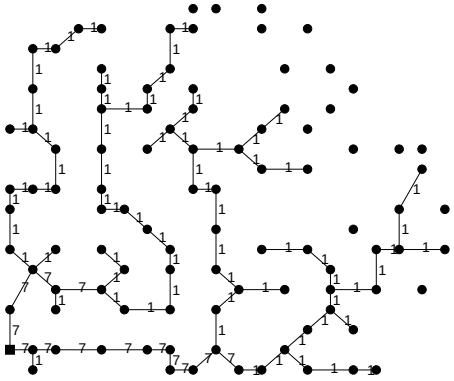
(b) Estágio 02



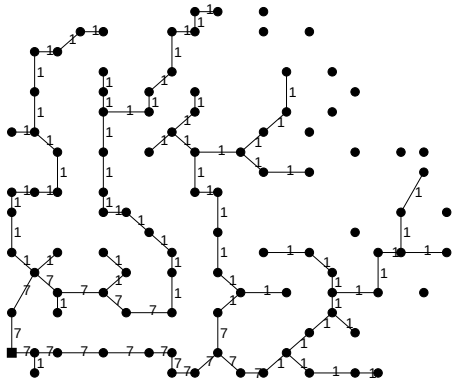
(c) Estágio 03



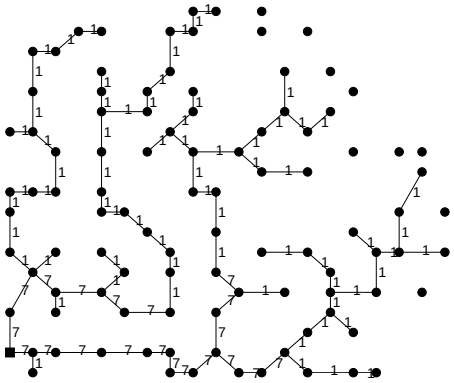
(d) Estágio 04



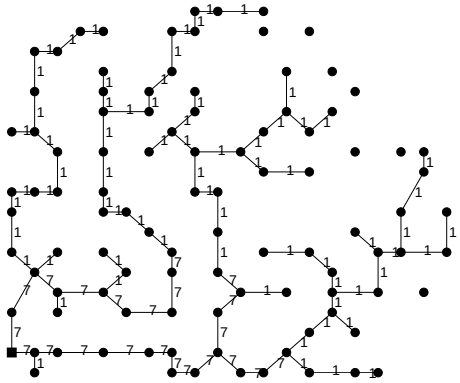
(e) Estágio 05



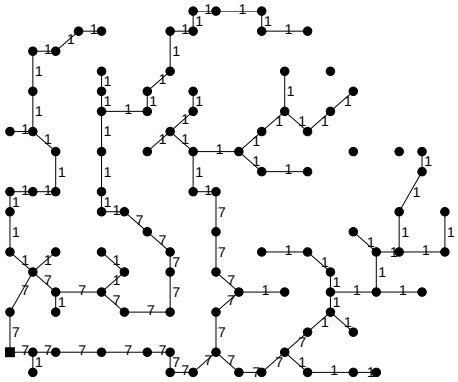
(f) Estágio 06



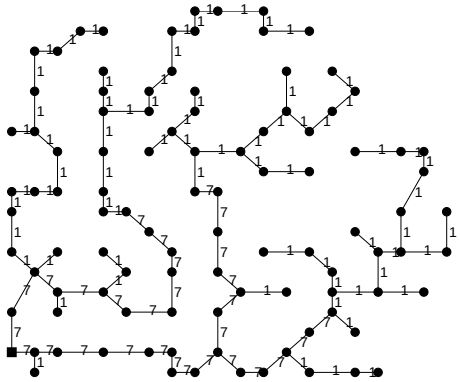
(g) Estágio 07



(h) Estágio 08



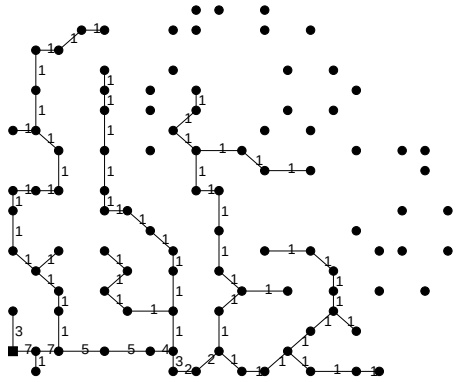
(i) Estágio 09



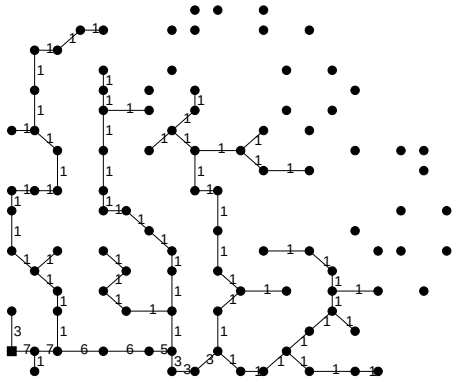
(j) Estágio 10

Figura C.2: *Scheduling* obtido pelo *ndGA*

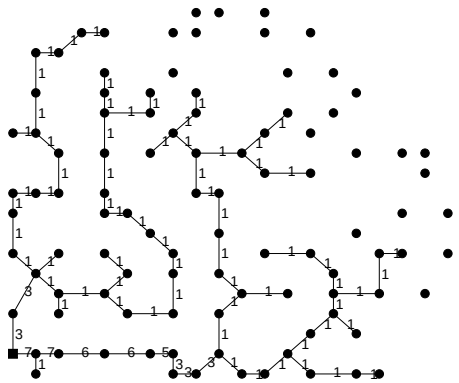
C.3 *Scheduling* Obtido pela Abordagem Incremental



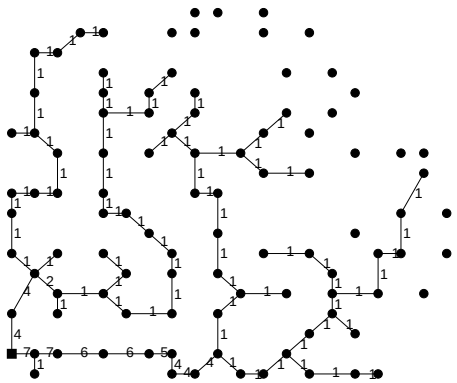
(a) Estágio 01



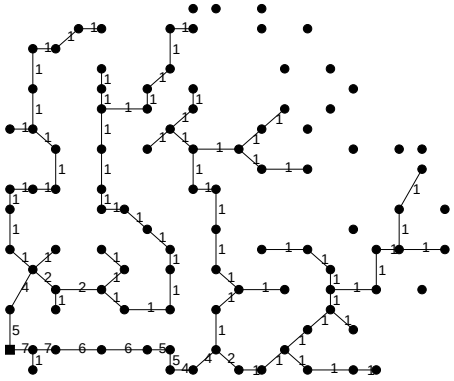
(b) Estágio 02



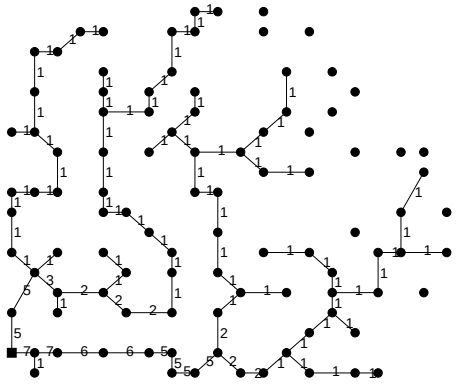
(c) Estágio 03



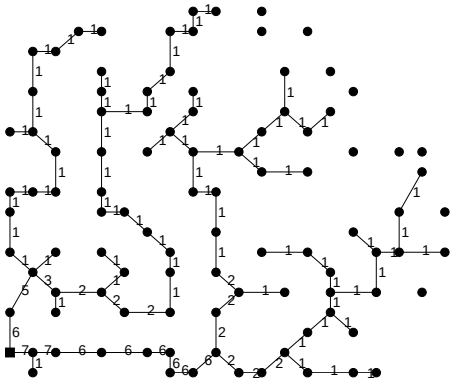
(d) Estágio 04



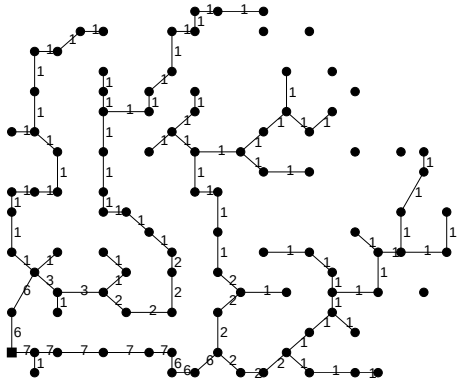
(e) Estágio 05



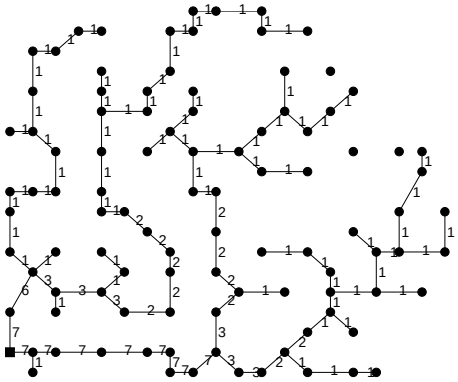
(f) Estágio 06



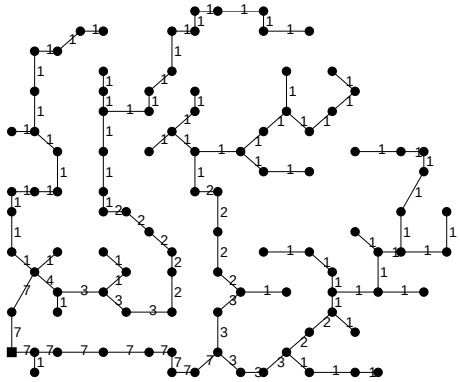
(g) Estágio 07



(h) Estágio 08



(i) Estágio 09



(j) Estágio 10

Figura C.3: *Scheduling* obtido pela *INC*