REPRESENTAÇÃO NÓ-PROFUNDIDADE PARA ALGORITMOS EVOLUTIVOS APLICADOS A MINIMIZAÇÃO DE PERDAS RESISTIVAS EM SISTEMAS DE DISTRIBUIÇÃO

AUGUSTO C. SANTOS[1], ALEXANDRE C. B. DELBEM[2], NEWTON G. BRETAS[3]

Departamento de Engenharia Elétrica - EESC-USP / Escola Técnica Federal de Palmas[1]
Instituto de Ciências Matemática e de Computação – USP[2]
Departamento de Engenharia Elétrica – EESC-USP[3]
Avenida Trabalhador São-carlense, 400, Cep. 13566-590 - São Carlos/SP
E-mails: augusto@etfto.gov.br, acbd@icmc.usp.br, ngbretas@sel.eesc.usp.br

Abstract — Reduction of power losses with some constraints such as radial network, maximum voltage drop and network loading, usually are problems that involve network reconfiguration procedures. When it is used a multi-objective functions, usually this objectives are conflicting, increasing the problems complexity. Several traditional Evolutionary Algorithms (EAs) have been proposed to deal with network design problems, but usually it has low performance, mainly for large-scale networks. In order to improve the EA performance for network designs, researches have proposed alternatives encoding. This paper adapts a new data structure to manipulate graphs. An EA using this representation is evaluated for reconfigure large-scale systems in order to decrease power losses, voltage drop, and loading currents in Distribution Systems. The performance achieved suggests that the proposed methodology can provide an efficient alternative for reconfiguration problems. It improves the solutions quality and largely reduces the running time for great size networks. Thus, it can be used in problems that require online solutions.

Keywords — Evolutionary Algorithms, Data Structures, Networks Design, Loss Reduction, Node-Depth Representation.

Resumo — Problemas de minimização de perdas resistivas com restrições, geralmente envolvem reconfiguração de redes. As restrições compreendem: rede radial, limites de queda de tensão, carregamento da rede, carregamento das subestações, além da garantia que todos os consumidores sejam atendidos. Quando se tem múltiplos objetivos, geralmente tais objetivos são conflitantes, aumentando o grau de complexidade do problema. Metodologias utilizando Algoritmos Evolutivos (AEs) em sua forma convencional têm sido aplicadas para reconfiguração de redes, porém geralmente com baixo desempenho, principalmente em redes de grande porte. A fim de melhorar o desempenho dos AEs para reconfiguração de redes, pesquisas têm proposto codificações alternativas. Este artigo adapta uma nova estrutura de dados para a manipulação de grafos. Um AE utilizando esta codificação é aplicado para minimização de perdas resistivas em sistemas de distribuição de energia elétrica de grande porte. Os resultados obtidos mostram que a metodologia empregada é uma alternativa eficiente para problemas de reconfiguração sendo capaz de lidar com problemas de tamanho real. Devido seu reduzido tempo de processamento, o mesmo pode ser utilizado em problemas que requerem soluções em tempo real.

Palavras-chave — AEs, Estrutura de Dados, Reconfiguração de Redes, Redução de Perdas, Representação Nó-Profundidade.

1. Introdução

Reconfiguração de Redes de Distribuição pode ser visto como um problema multiobjetivo de natureza combinatorial. Geralmente compreende minimização de perdas ôhmicas com algumas restrições: nível de tensão dentro de limites permitidos, capacidade dos cabos e equipamentos não violada, número mínimo de chaves manobradas, sistema radial, atendimento a todos os consumidores. A necessidade de gerar configurações factíveis[1] e o elevado número de chaves num sistema de distribuição aumenta a complexidade do problema. A fim de resolver estes problemas, vários Algoritmos Evolutivos (AEs) tem sido propostos (Augugliaro et al., 2000), (Carvalho et al., 2001), (Chou et al., 2001), (Delbem et al., 1998) e (Li et al., 2000). Os resultados obtidos têm encorajado pesquisas nesta área, principalmente quando comparados com Programação Matemática (PM) e técnicas de Inteligência Artificial (Merlin e Back, 1975) em sua forma convencional de representação. Porém o problema de explosão combinatorial restringe a aplicação de PM para sistemas de pequeno porte. AEs em sua representação convencional tem tido um bom desempenho para redes de pequeno e médio portes, porém quando aplicados a sistemas de grande porte tem encontrado muitas soluções infactíveis e o tempo de processamento muito elevado.

A codificação utilizada na representação de uma árvore de grafo para AEs aplicados a grandes redes se tornou um fator crítico. Um AE convencional é representado por um único cromossomo contendo o estado das chaves de seccionamento do sistema através de uma cadeia de bits. Quando a chave é Normalmente Fechada (NF) o gene que representa seu estado é 1 e, quando a chave é Normalmente Aberta (NA), seu gene é 0. Na tentativa de gerar novas configurações para o sistema, é alterado o estado de duas ou mais chaves aleatoriamente. Isso produz muitas configurações infactíveis, tornando sua convergência, quando ocorre, muito lenta.

Para superar esses problemas e aumentar o desempenho dos AEs, principalmente em redes de grande porte, este artigo propõe uma nova codificação baseada na representação do nó e de sua profundidade na árvore de grafo. Juntamente com a Representação Nó-Profundidade (RNP), é desenvolvido um AE e dois operadores eficientes, que evitam a geração de soluções infactíveis.

Com essa nova codificação é possível representar florestas (grafos compostos de um conjunto de árvores), agilizar os cálculos utilizados no fluxo de carga e, resolver Problemas de

[1] Configurações factíveis nesse contexto são radiais, todos os consumidores atendidos e sem a interligação entre alimentadores.

Reconfiguração de Sistemas de Distribuição (PRSD) de tamanho real em um tempo relativamente pequeno. O ótimo tempo de processamento garante que o mesmo pode ser utilizado para resolver problemas que requerem soluções em tempo real.

A seção 2 apresenta a formulação matemática para PRSD; a seção 3 apresenta a estrutura de dados utilizada na RNP para árvores de grafos; a seção 4 descreve os operadores; a seção 5 demonstra como localizar um nó em uma floresta *F* e como determinar, eficientemente, o conjunto de nós requerido pelos operadores; a seção 6 apresenta testes experimentais para PRSD; a seção 7 discute os principais aspectos da metodologia proposta.

2. Formulação do Problema

Para obter a formulação geral de um problema de Sistema de Distribuição Radial (SDR), consideram-se juntos os objetivos e restrições.

Um problema de minimização de perdas consiste basicamente em determinar uma configuração radial para o sistema que atenda todos os consumidores e minimize as perdas resistivas na rede sem violar as restrições de tensão nas barras e carregamento nas linhas e no sistema.

O problema pode ser formulado como segue:

$$\begin{aligned} Min.\quad & F(G) \\ & H(G) = 0 \\ s.a.\quad & I(G) \le 0 \\ & G \text{ ser uma } floresta, \end{aligned} \tag{1}$$

onde:

- G – grafo correspondente a configuração do sistema;
- $F(G)$ – função objetivo;
- $H(G)$ – restrições de igualdade representando as equações de fluxo de carga;
- $I(G)$ – restrições de desigualdade representando as restrições operacionais do sistema.

Para minimização de perdas, a função $F(G)$ contém, em geral, os seguintes componentes:

- $\varphi(G)$ – perdas resistivas no sistema para G.
- $\psi(G,G^0)$ – número de operações de chaveamento para obter uma dada configuração G, a partir de uma configuração original G^0.

As restrições de igualdade correspondem as equações de fluxo de carga e representa um sistema linear do tipo Ax = b, onde:

- A – matriz incidência de G;
- x – vetor corrente de linha;
- b – vetor com as correntes nas barras ($b_i \le 0$), ou injeções de correntes nas subestações ($b_i > 0$).

As restrições operacionais $I(G)$ em PRSD geralmente incluem:

- um limitante superior de corrente $\overline{x}_j$ para cada corrente de linha x_j. A maior taxa $x_j / \overline{x}_j$ é denominada carregamento da rede;
- a máxima injeção de corrente $\overline{b}_i$ possível para cada subestação i, onde a maior taxa $b_i / \overline{b}_i$ é denominada carregamento da subestação;
- um limitante inferior para tensão nos nós $\underline{v}$.

AEs são empregados para resolução desse tipo de problema, porém algumas modificações são realizadas na formulação apresentada em (1). A fim de penalizar as configurações que violarem as restrições operacionais $I(G)$, são inseridos fatores de penalidades (Goldberg, 1989). Dessa forma, o PRSD pode ser reformulado como segue:

$$\begin{aligned} Min.\quad & F(G) + |\Omega I(G)| \\ & H(G) = 0 \\ s.a.\quad & G \text{ ser uma } floresta, \end{aligned} \tag{2}$$

onde Ω é um vetor com os pesos (w_x, w_s e w_v) para cada termo das restrições de fluxo de carga inserido na função objetivo. Os pesos w_x, w_s e w_v são valores positivos e $|\bullet|$ é a norma infinita usual e correspondem respectivamente aos fatores multiplicativos para carregamento da rede, carregamento do sistema e queda de tensão.

Como são produzidas somente configurações factíveis, não há a necessidade de utilizar uma rotina específica para verificar e nem para corrigir infactibilidades. Há uma redução significativa no tempo de processamento.

A formulação do problema anterior pode ser simplificada usando uma nova representação de redes de distribuição, denominada Representação Nó-Profundidade (RNP) e seus operadores correspondentes. O AE proposto utilizando a RNP produz somente configurações factíveis. Dessa forma, a formulação do problema de (2) pode ser reescrito como segue:

$$\begin{aligned} Min.\quad & F(G) + |\Omega I(G)| \\ & H(G) = 0 \\ s.a.\quad & G \text{ ser dado pelos operadores da RNP.} \end{aligned} \tag{3}$$

3. Representação de Florestas pelo nó e profundidade de nó

Um SDR pode ser representado por grafos e deve ser factível. Portanto, um grafo para representar SDRs deve ser conexo e acíclico (Ahuja, 1993).

A codificação proposta é baseada no conceito de nó e profundidade de nó em uma árvore de grafo. A representação consiste basicamente de uma lista linear contendo os nós e suas profundidades na árvore. A ordem em que os pares (nó, profundidade) são dispostos na lista é importante. Uma busca em profundidade (Cormen, 1990) em uma árvore de grafo produz uma ordem apropriada para inserção dos nós e sua profundidade na lista quando cada nó n_x do grafo é visitado pela busca.

A Figura 1 apresenta uma floresta com duas árvores. Os nós 1 e 20 são os nós raízes das árvores 1 e 2 respectivamente. O Grafo da Figura 1 pode ser visto como um SDR com 2 alimentadores em que os

nós são pontos de carga e as arestas são chaves seccionadoras. As arestas em linha cheia representam as chaves NF e, as arestas em linha pontilhada representam as chaves NA. Os nós 1 e 20 estão em uma subestação. A Figura 2 apresenta a RNP para as duas árvores do grafo da Figura 1.

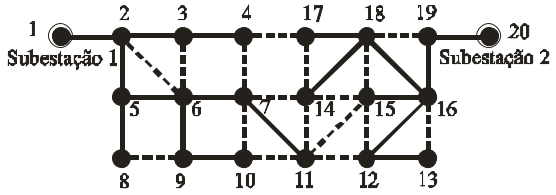


Figura 1: Grafo contendo duas árvores.

A codificação para uma floresta é composta pela união das codificações de todas as árvores da mesma. Assim, a estrutura de dados da floresta pode ser facilmente implementada utilizando *arrays* de ponteiros, onde cada ponteiro indica a codificação nó-profundidade de uma árvore na floresta.

T_1	profundidade	0 1 2 3 2 3 3 4 5 4 5
	nó	1 2 3 4 5 8 6 7 11 9 10
T_2	profundidade	0 1 2 3 3 4 3 4 4
	nó	20 19 16 15 12 13 18 17 14

Figura 2: Representação nó-profundidade para as Árvores T_1 e T_2.

4. Operadores

Esta seção apresenta dois operadores com RNP para gerar uma floresta *F'* de um grafo *G* quando eles são aplicados a outra floresta *F* de *G*.

Os resultados produzidos pela aplicação de ambos operadores são similares. Dado uma floresta com duas árvores ou mais, a aplicação do operador 1 (ou 2) é equivalente a transferir uma sub árvore de uma árvore T_1 para uma outra árvore T_2 da floresta. A diferença básica entre os operadores 1 e 2 está no fato que, ao aplicar o operador 1, a raiz da sub árvore cortada de T_1 será também a raiz dessa sub árvore na nova árvore T_2. Ao aplicar o operador 2, a sub árvore cortada terá um novo nó raiz que poderá ser qualquer nó da sub árvore cortada diferente da raiz original.

Como resultados, o operador 1 produz simples e pequenas mudanças nas árvores da floresta, enquanto o operador 2 gera grandes e complexas alterações. O operador 1 requer um conjunto de dois nós determinados previamente: o nó de poda (n_p) que indica a raiz da sub árvore a ser transferida e, o nó adjacente (n_a), que é um nó de uma árvore diferente da árvore cortada (T_1) e também adjacente a n_p em *G*. Os nós adjacentes a n_p podem ser armazenados em uma lista de adjacência[2] (Ahuja, 1993). O operador 2 requer um conjunto de três nós: o nó de poda (n_p), o nó adjacente (n_a) e o novo nó raiz (n_r) da sub árvore a ser transferida.

A seguir será demonstrado como os operadores atuam considerando que o conjunto de nós requerido foi determinado previamente.

[2] A lista de adjacência considera todas as conexões possíveis de cada nó, isto é, conexões referentes tanto a chaves abertas quanto fechadas. Desta forma, não altera de uma configuração para outra.

4.1. Operador 1

Considerando que a RNP foi desenvolvida utilizando *arrays* e, que os nós n_p e n_a foram previamente determinados, podemos assumir que os índices de n_p (i_p) e n_a (i_a) nos *arrays* T_1 e T_2 respectivamente também são conhecidos.

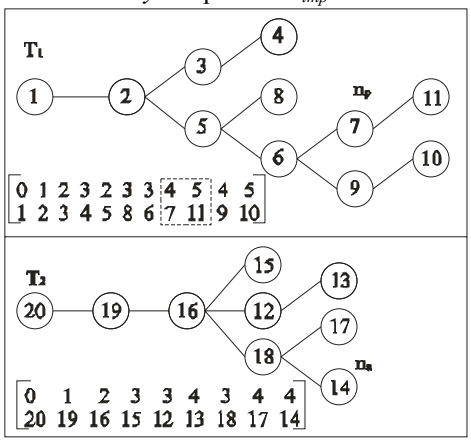
Os passos a seguir descrevem o operador 1:

1. Determine as posições (i_p – i_u) dos índices em T_1 correspondente a sub árvore que contém o nó n_p como raiz. Como i_p é conhecido, é necessário encontrar apenas i_u. O conjunto (i_p – i_u) corresponde ao nó n_p em i_p e os consecutivos nós n_x no *array* T_1 de forma que $i_x > i_p$ e $p_x > p_p$ (as linhas pontilhadas na Figura 3(a)), onde p_x é a profundidade do nó n_x e p_p é a profundidade do nó n_p;
2. Copie os dados do conjunto (i_p – i_u) de T_1 em um *array* temporário T_{tmp} (contendo os dados da sub árvore que está sendo transferida), ver Figura 3(b). A profundidade de cada nó n_x do conjunto i_p–i_u é atualizado da seguinte maneira: $p_x = p_x - p_p + p_a + 1$;
3. Crie um *array* T'_2 contendo os dados de T_2 e T_{tmp}, isto é, uma nova árvore deverá ser gerada conectando a sub árvore cortada de T_1 à T_2;
4. Construa um *array* T'_1 compreendendo os nós de T_1 sem os nós de T_{tmp};
5. Copie a estrutura de dados da floresta *F* para *F'* trocando os ponteiros dos *arrays* T_1 e T_2 para os *arrays* T'_1 e T'_2, respectivamente.

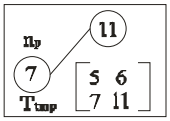
4.2. Operador 2

A descrição do operador 2 assume que um conjunto de nós seja previamente determinado: n_p, n_r, e n_a. Os nós n_p e n_r pertencem à árvore T_1 e, o nó n_a à T_2.

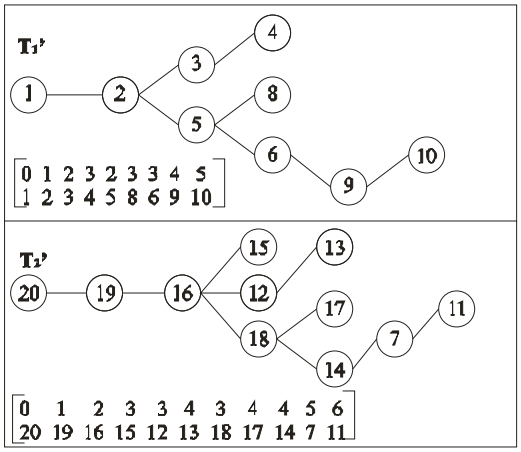
As diferenças entre os operadores 1 e 2 estão nos passos 2 e 3 do procedimento do operador 1, isto é, a formação da sub árvore cortada e o armazenamento da mesma no *array* temporário T_{tmp} são diferentes.



(a) T_1, T_2 na representação nó-profundidade.



(b) T_{tmp} *na representação nó-profundidade.*



(c) T_1', T_2' na representação nó-profundidade.

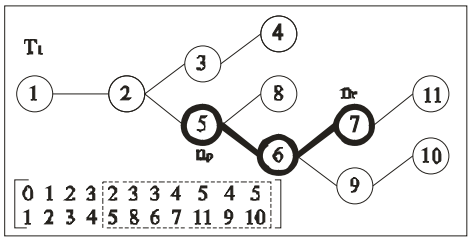
Figura 3: Exemplo de aplicação do operador 1.

A Figura 4 ilustra os passos 2 e 3 para o operador 2.

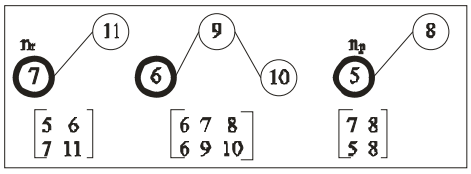
Para o operador 2 o procedimento de copiar a árvore cortada pode ser dividido em dois passos: o primeiro passo é similar ao passo 2 do operador 1, diferindo apenas na troca de i_p por i_r. O *array* retornado por este procedimento é chamado de T_{tmp1}.

O segundo passo considera os nós na cadeia de n_r a n_p (isto é, r_0, r_1, r_2, ..., r_n, onde $r_0 = n_r$ e $r_n = n_p$) como raízes das sub árvores (linhas destacadas na Figura 4(a)). O algoritmo para o segundo passo deverá copiar a sub árvore com a raiz r_i ($i = 1, ..., n$) sem a sub árvore com raiz r_{i-1} (Figura 4(b)) e armazenar a sub árvore resultante em um *array* temporário T_{tmp2} (ver Figura 4(c)).

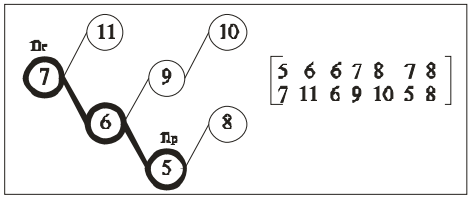
O operador 2 utilizará ambos *arrays* temporários, T_{tmp1} e T_{tmp2} para construir a T'_2.



(a) T_1 e a representação nó-profundidade.



(b) Sub árvores com raízes nos nós da cadeia de n_r a n_p.



(c) Representação nó-profundidade para a árvore cortada.

Figura 4: Exemplo de determinação de T_{tmp2}. As linhas em destaque realçam os nós na cadeia de n_r a n_p. As profundidades em (b) e (c) são relativas ao sorteio do nó 14 como adjacente.

5. Determinação dos nós n_p, n_r e n_a

Nesta seção será apresentado como obter o conjunto de nós requerido pelos operadores 1 e 2 e suas respectivas posições em F.

5.1. A Posição do Nó em F

A posição de um nó em F pode ser eficientemente determinada usando matrizes Π_x e um vetor π. Cada nó n_x de G possui uma matriz Π_x correspondente. Para a floresta original F_0 de G, Π_x é a matriz coluna:

$$\Pi x = \begin{bmatrix} 0 \\ i_0 \\ j_0 \\ k_0 \end{bmatrix}$$

onde, i_0 é o índice da árvore que contém n_x (T_{i0}), j_0 é o índice correspondente a n_x no *array* T_{i0} e k_0 é a profundidade de n_x na árvore.

Suponha que uma floresta F_h está sendo gerada de outra floresta F_g ($g < h$) e que n_x está na sub árvore que será transferida para uma nova árvore gerada em F_h. Dessa maneira, n_x terá uma nova posição em F_h diferente da posição em F_g. Deve-se inserir uma nova coluna em Π_x com os índices correspondentes a essa nova posição, resultando em:

$$\Pi x = \begin{bmatrix} 0 & h \\ i_0 & i_h \\ j_0 & j_h \\ k_0 & k_h \end{bmatrix}.$$

A atualização da posição é executada para todos os nós da sub árvore transferida (T_{tmp}), para os nós com índices maiores que i_u na árvore T_1 e, para os nós de T_2 que se localizarem em posições posteriores às posições dos nós da árvore transferida T_{tmp} em T'_2.

O vetor π armazena a floresta g, da qual a floresta F_h foi gerada, na posição h de π, isto é, $\pi(h)=g$. O pai de g é $\pi(g)$, o pai de $\pi(g)$ é $\pi(\pi(g))$ e assim por diante. Constitui assim, uma lista encadeada com todos os predecessores de F_h. Obviamente que a última mudança de posição de n_x ocorreu em um dos predecessores de h. Pode-se então procurar pelo predecessor de h nas colunas de Π_x. A busca se inicia por $\pi(h)$. Se esta coluna não é encontrada em Π_x então se tenta a coluna $\pi(\pi(h))$, e assim por diante. O processo de procurar por tais colunas em Π_x pode ser eficientemente implementado através de uma busca binária (Goodaire, 1998) na lista dada por Π_x (0, .) na primeira linha em Π_x.

Uma vez identificada a coluna com o predecessor de h, é necessário apenas ler os índices de n_x armazenados na mesma coluna.

5.2. Determinação dos nós n_p, n_r e n_a

Os operadores propostos requerem um conjunto especial de nós para gerar uma floresta F' de G baseada em outra floresta F de G.

Para o operador 1 esse conjunto pode ser eficientemente obtido pela seguinte estratégia:

1. Selecione aleatoriamente um índice de uma árvore em F e, para esta árvore, selecione aleatoriamente o índice de um nó que não seja uma raiz. Chame esse nó de nó de poda (n_p);
2. Selecione aleatoriamente, um nó adjacente a n_p (utilize para isso a lista de adjacência de G). Chame esse nó de nó adjacente (n_a). Se $n_a \notin T_1$, determine a posição em F utilizando o vetor π e a matriz Π_a, caso contrário, selecione aleatoriamente, outro n_a ou retorne ao passo 1.

A estratégia para a determinação de n_p, n_r e n_a para o operador 2 é desenvolvida como segue:

1. Selecione aleatoriamente um índice de uma árvore em F e, para esta árvore, selecione aleatoriamente um índice de um nó que não seja uma raiz. Chame-o de nó de poda (n_p);
2. Determine o conjunto de nós na sub árvore que tenha o nó n_p como raiz, como no passo 1 do operador 1. Escolha aleatoriamente um índice desse conjunto para ser o novo nó raiz (n_r);
3. Selecione aleatoriamente um nó adjacente a n_r (utilize os nós da lista de adjacência de G). Chame esse nó de nó adjacente n_a. Se $n_a \notin T_1$, determine a posição em F utilizando o vetor π e a matriz Π_a, senão, selecione outro n_a ou retorne ao passo 1.

6. Testes com o Algoritmo Evolutivo Proposto

Esta seção apresenta os resultados de testes realizados em SDRs. Todos os testes foram realizados em um computador com processador core 2 duo de 1.86GHz, 2GRAM de memória, Sistema Operacional *Ubuntu 6.06* e o compilador de linguagem C *gcc*.

A função objetivo utilizada nos testes foi: $f(x) = \delta_1 x_1 + \delta_2 x_2 + \delta_3 x_3 + \delta_4 x_4$, onde x_1, x_2, x_3 e x_4 são respectivamente as perdas em kW, número de manobras de chaves, carregamento máximo da rede em pu e, maior queda de tensão na geração em pu e, δ_1, δ_2, δ_3, δ_4 são os pesos relativos a cada termo da função objetivo respectivamente.

Os pesos utilizados na função de avaliação variam para cada teste e foram determinados empiricamente até se obter resultados satisfatórios.

6.1. Sistema de médio porte

O sistema em análise foi utilizado em outros trabalhos (Kagan, 1999), (Delbem, 2002) e consiste de 25 setores, 93 barras, 31 chaves, sendo 17 chaves NF e 14 chaves NA. Os testes foram realizados para minimização de perdas com múltiplos objetivos.

A Tabela 1 apresenta as características originais do sistema e seus valores médios otimizados.

Os pesos utilizados na função ponderada foram:

$$\delta_1 = \delta_2 = \delta_3 = \delta_4 = 1.$$

Tabela 1: Sistema de Médio Porte - Múltiplos Objetivos.

	Original	Otimizado
Perdas Totais na Rede	1453kW	1252 kW
Queda de Tensão Máxima	9,4%	5,6%
Carregamento Máximo da Rede	61%	67,10%
Número de Manobras	0	02 pares
Tempo médio de processamento	0	0,009s

Os resultados encontrados são exatamente os encontrados por outros trabalhos, diferenciando apenas no tempo de processamento. Em (Kagan, 1999) o tempo de processamento para gerar 100 configurações de 150 indivíduos foi de 30s utilizando um computador Pentium III, 450MHz. Em (Delbem, 2002), o tempo de processamento para gerar 1500 indivíduos foi de 0,24s utilizando um Pentium III, 450MHz. Segundo (Kagan, 1999) este é um ótimo global. Foram realizadas 30 execuções com diferentes sementes para o gerador de números aleatórios para gerar 1000 configurações, onde a média do tempo de processamento foi de 0,009s e o desvio padrão, 0,003. A média das perdas encontradas foi de 1251,54kW e o desvio padrão, 0,000.

6.2. Sistema Reduzido de São Carlos/SP

Testes foram realizados em um sistema de grande porte, sistema reduzido da cidade de São Carlos/SP. Esta rede possui 204 setores, 1473 barras, 249 chaves, sendo 181 chaves NF e 68 NA, divididos em 3 subestações e 23 alimentadores.

Os testes foram realizados com 20 indivíduos na população e executados até 5000 gerações. Para cada caso foram realizadas 30 execuções com diferentes sementes para o gerador de números aleatórios. Mais detalhes pode ser visto em (Santos, 2004).

A Tabela 2 apresenta os resultados obtidos considerando múltiplos objetivos. Os pesos utilizados para este teste foram:

$$\delta_1 = \delta_3 = \delta_4 = 1$$

$$\delta_2 = 4.$$

Tabela 2: Sistema de Grande Porte - Múltiplos Objetivos

	Original	Otimizado
Perdas Totais na Rede	674 kW	626 kW
Queda de Tensão Máxima	1,32%	1,32%
Carregamento Máximo da Rede	65,35%	61,43%
Número de Manobras	0	04 pares
Tempo médio de processamento	0	0,21s

Em média o melhor indivíduo foi encontrado na geração 1748. O desempenho do algoritmo é mostrado na Figura 5. As médias do tempo de processamento e perdas foram de 0,205s e 626,11kW respectivamente. O desvio padrão foi de 0,006 para o tempo de processamento e 0,102 para as perdas.

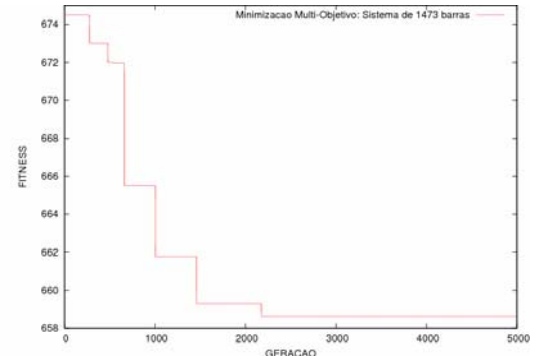


Figura 5: Desempenho do Algoritmo no SDR de Grande Porte com a Função Ponderada.

A Figura 6 mostra os indivíduos na população ao longo das gerações.

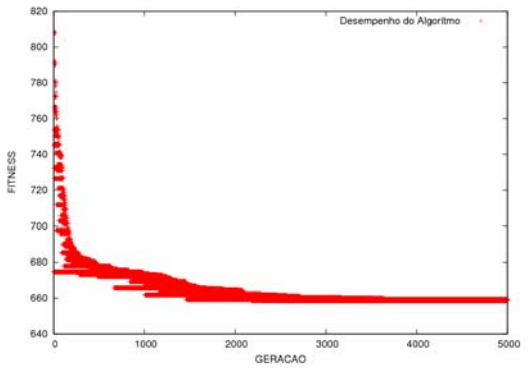


Figura 6: Indivíduos pertencentes à população.

7. Conclusões

Problemas de Reconfiguração de Sistema de Distribuição Radial são altamente complexos e de natureza combinatorial. Para resolver esse tipo de problemas vários métodos baseados em Inteligência Artificial têm sido propostos. Algoritmos Evolutivos pode ser uma eficiente alternativa para resolver problemas de reconfiguração de SDRs, porém requerem um alto tempo de processamento quando aplicados a sistemas de grande porte. A codificação do AE tem uma importância crítica no desempenho do algoritmo para esses sistemas.

Para resolver esse problema, é proposta uma codificação para AEs baseada em nós e na profundidade de nós na árvore de grafo, juntamente com dois operadores que geram apenas configurações factíveis. Tal codificação é capaz de representar florestas. A estrutura de dados desenvolvida acelerou os cálculos de fluxo de carga e aumentou o desempenho da técnica, possibilitando trabalhar com sistemas reais de grande porte.

Testes foram realizados em um sistema de médio porte utilizado em outros trabalhos e, em um sistema de grande porte. Os resultados obtidos demonstram que a técnica proposta é capaz de lidar com grandes redes de grafos utilizando tempo de processamento relativamente pequeno.

Agradecimentos

Os autores agradecem cordialmente ao CNPq o suporte financeiro para o desenvolvimento deste trabalho.

Referências Bibliográficas

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993). Network Flows: Theory, Algorithms, and Applications, *Printce Hall, Englewood Cliffs*.

Augugliaro, A., Dusonchet, L., Sanseverino, E.D. (2000). Multiobjective service restoration in distribution networks using an evolutionary approach and fuzzy sets. *Elect. Power Energy Syst*., vol. 22, pp. 103-110;

Carvalho, P.M.S., Ferreira, L.A.F.M., Barruncho, L.M.F. (2001). On spanning tree recombination in evolutionary large-scale network problems – Application to electrical distribution planning. *IEEE Trans. Evol. Comput*., vol. 5, no 6, pp. 623-630, Dec.;

Chou, H.H., Premkumar, G. Chu, C.H. (2001). Genetic algorithms for communications network design – An empirical study of the factors that influence performance. *IEEE Trans. Evol. Comput*., vol. 5, no. 3, pp. 236-249, Jun.;

Delbem, A.C.B., "Restabelecimento de Energia em Sistemas de Distribuição por Algoritmo Evolucionário Associado a Cadeias de Grafos", *EESC-USP/ Tese de Doutorado*, 2002;

Delbem, A.C.B., Carvalho, A., Bretas, N.G. (1998). Energy restoration in distribution systems using search with fuzzy heuristics. *Int. J. Eng. Intell. Syst. Elect. Eng. Commun.* Vol. 6, pp. 201-205;

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley;

Goodaire, E.G., Parmenter, M.M. (1998). Discrete Mathematics with Graph Theory. *Englewood Cliffs*, NJ: Prentice-Hall;

Kagan, N. (1999). Configuração de Redes de Distribuição através de Algoritmos Genéticos e Tomada de Decisão Fuzzy, *Tese de livre docência/EP-USP*, São Paulo.

Li, Y., Bouchebaba, Y. (2000). A new genetic algorithms for the optimal communication spanning tree problem. *Artificial Evol.*, vol. 1829, no. 1, pp. 162-173(1);

Merlin, Back, H. (1975). Search for a minimal-loss operating spanning tree configuration in na urban power distribution system, *In: Power System Computation Conference, 5*, *Cambridge*, 1975. Proceedings. v.1, p.1-18.

Santos, A. C. (2004). Restabelecimento de Energia Considerando todas Barras e Chaves de um Sistema de Distribuição Real, *EESC-USP/Dissertação* (Mestrado).